AGVC SDK

0.8.0

Generated by Doxygen 1.16.1

Chapter 1

AGVC SDK

Overview

AGVC SDK provides a complete set of interfaces for controlling and managing AGV systems. Applications can communicate with AGV controllers through the AGVC API entry points to perform map management, path planning, motion control, charging, system status monitoring, and other operations.

Contents

Quick Start

Basic Workflow

```
// 1. Initialize the AGV SDK client
auto rpc = std::make_shared<agvc_interface::RpcClient>();

// 2. Configure AGV communication
// 2.1 Set request timeout
rpc->setRequestTimeout(100);
// 2.2 Connect to the AGV
rpc->connect("192.168.192.100", 30104);
// 2.3 Log in
rpc->login("aa", "bb");

// 3. Use AGV interfaces
// 3.1 Query AGV details
rpc->getAgvDetails();
// 3.2 Get laser point cloud
rpc->getLaserPointCloud();
```

SDK Architecture

Module Structure

AGVC_API (main entry)
- Map
- Station
- Path
- Navigation
- Charging
- AgvInfo
- System

Usage Examples

Get Robot Details

```
// Establish communication
auto rpc = std::make_shared<agvc_interface::RpcClient>();
rpc->setRequestTimeout(100);
rpc->connect("192.168.192.100", 30104);
rpc->login("aa", "bb");

// Get AGV details
auto agv_details = rpc->getAgvDetails();
```

Create a Map

```
// Acquire control priority
rpc->setPriority("aa", "");

// Switch to mapping mode
agvc_interface::Header header;
header.id = "123sdaw-asdafas-zff";
agvc_interface::RunningMode mode = agvc_interface::RunningMode::MAPPING;
auto ret = rpc->changeRunningMode(mode, header);

// Query the execution status of the asynchronous changeRunningMode interface
agvc_interface::AsyncInterfaceResultStatus async_result = rpc->getAsyncInterfaceResultStatus();
agvc_interface::ChangeRunningModeStatus change_running_mode_status = async_result.change_running_mode_status.status;

// If the asynchronous interface is still running, keep querying until it finishes
while (change_running_mode_status == agvc_interface::ChangeRunningModeStatus::RUNNING)
{
    change_running_mode_status = rpc->getAsyncInterfaceResultStatus().change_running_mode_status.status;
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}

// Query the final result after the asynchronous operation finishes
ret = rpc->changeRunningMode(mode, header);

// Check the result
if (ret == 10100000) {
    std::cout « "Mapping mode switched successfully";
} else {
    std::cout « "Failed to switch mapping mode, error code: " « ret;
}
```

Save a Map

```
// Acquire control priority
rpc->setPriority("aa", "");

// Configure map header
agvc_interface::Header map_header;
map_header.id     = "test_save_map";
map_header.map_id = map_id;
map_header.name   = "test1";
map_header.stamp  = 1;

auto ret = rpc->saveMap(map_header);

// Query the execution status of the asynchronous saveMap interface
```

```
agvc_interface::AsyncInterfaceResultStatus async_result = rpc->getAsyncInterfaceResultStatus();
auto save_map_status = ret.save_map_status.status;

// If the asynchronous interface is still running, keep querying until it finishes
while (save_map_status == agvc_interface::ChangeRunningModeStatus::RUNNING)
{
    save_map_status = rpc->getAsyncInterfaceResultStatus().save_map_status.status;
    std::this_thread::sleep_for(std::chrono::milliseconds(10));
}

// Query the final result after the asynchronous operation finishes
ret = rpc->saveMap(map_header);

// Check the result
if (ret == 10100000) {
    std::cout « "Map saved successfully";
} else {
    std::cout « "Failed to save map, error code: " « ret;
}
```

Namespace

All interfaces are defined in the following namespace:

```
namespace agvc_interface {
    // Interface definitions
}
```

Error Handling

Most interface methods return integer error codes. 10100000 indicates success, while other values indicate specific errors that can be queried through the error-code interface. Some interfaces may throw Aubo↩ Exception exceptions.

Deprecated List

For deprecated APIs and fields, see Deprecated List.

Supported Programming Languages

- C++
- Python
- JSON-RPC

Related Files

- agvc_interface.h - Interface definitions
- type.h - Type definitions

Chapter 2

Deprecated List

Deprecated List

This page summarizes deprecated C++ APIs and fields in the current SDK. Prefer the replacement APIs or fields.

Deprecated item	Since	Replacement	Description
agvc_interface::AgvcInterface::checkPowerAndAutoCharge()	0.7.0	sendAutoChargingCommand(const AutoChargingCommand&)	Checks the current battery level and automatically docks for charging when the battery is low.
agvc_interface::AgvcInterface::setLeaveBoardTargetStation()	0.7.0	sendAutoChargingCommand(const AutoChargingCommand&)	Sets the target station for automatic undocking.
agvc_interface::AgvcInterface::getLeaveBoardTargetStation()	0.7.0	getNavInfo()	Gets the target station for automatic undocking.
agvc_interface::AgvcInterface::forcedAgvToChargingBoard()	0.7.0	sendAutoChargingCommand(const AutoChargingCommand&)	Forces the AGV to dock for charging.
agvc_interface::AgvcInterface::forcedAgvLeaveChargingBoard()	0.7.0	sendAutoChargingCommand(const AutoChargingCommand&)	Forces the AGV to leave the charging board.
agvc_interface::NavInfo::type	0.7.0	nav_type	The navigation type field is replaced by nav_type.

Chapter 3

Topic Index

3.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 4

Directory Hierarchy

4.1 Directories

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Topic Documentation

9.1 Map

Functions

- std::vector< Header > agvc_interface::AgvcInterface::getMapList ()

  Gets the headers of all maps on the AGV.
- Header agvc_interface::AgvcInterface::getCurrentMapHeader ()

  Queries the header of the current AGV map.
- OccupancyGridMap agvc_interface::AgvcInterface::getGridMapFromAgv (const Header &map_header)

  Gets the specified occupancy grid map information.
- Base64PngMap agvc_interface::AgvcInterface::getBase64PngMapFromAgv (const Header &map_header)

  Gets Base64-encoded PNG map information.
- Base64PngMap agvc_interface::AgvcInterface::previewPngMapFromAgv (const Header &map_header, const int &image_width_px=0, const int &image_height_px=0)

  Gets a preview image with optional width and height.
- int agvc_interface::AgvcInterface::sendGridMapToAgv (const OccupancyGridMap &map)

  Sends occupancy grid map information to the AGV.
- int agvc_interface::AgvcInterface::sendBase64PngMapToAgv (const Base64PngMap &map)

  Sends a Base64-encoded PNG map to the AGV.
- int agvc_interface::AgvcInterface::saveMap (const Header &map_header)

  Saves the map after mapping is complete.
- int agvc_interface::AgvcInterface::switchMap (const Header &map_header)

  Switches to the specified map.
- int agvc_interface::AgvcInterface::deleteMap (const Header &map_header)

  Deletes one specified map.
- int agvc_interface::AgvcInterface::deleteMaps (const std::vector< Header > &map_headers)

  Deletes multiple specified maps.
- std::vector< MapVirtualArea > agvc_interface::AgvcInterface::getAllMapVirtualArea ()

  Queries virtual areas on the current map.
- std::vector< MapVirtualArea > agvc_interface::AgvcInterface::getAllMapVirtualAreaOfTargetMap (const Header &map_header)

  Queries virtual areas on the target map.

- int agvc_interface::AgvcInterface::addMapVirtualArea (const MapVirtualArea &map_virtual_area)

  Adds or modifies one virtual area.
- int agvc_interface::AgvcInterface::addMapVirtualAreas (const std::vector< MapVirtualArea > &map_virtual_areas)

  Adds or modifies multiple virtual areas.
- int agvc_interface::AgvcInterface::deleteMapVirtualArea (const Header &virtual_area_header)

  Deletes the specified virtual area.
- int agvc_interface::AgvcInterface::deleteMapVirtualAreas (const std::vector< Header > &virtual_areas_header)

  Deletes multiple specified virtual areas.
- MapAllInfo agvc_interface::AgvcInterface::getGridMapAllInfo (const Header &map_header)

  Gets occupancy grid map data, paths, stations, and virtual areas for the specified map.
- MapAllInfo agvc_interface::AgvcInterface::getPngMapAllInfo (const Header &map_header)

  Gets Base64 map data, paths, stations, and virtual areas for the specified map.
- int agvc_interface::AgvcInterface::setGridMapAllInfo (const MapAllInfo &map_all_info, const Header &command_header={ "99999", "99999", 1, "99999" })

  Sets occupancy grid map, path, station, and virtual area information.
- int agvc_interface::AgvcInterface::setPngMapAllInfo (const MapAllInfo &map_all_info, const Header &command_header={ "99999", "99999", 1, "99999" })

  Sets Base64 map, path, station, and virtual area information.
- int agvc_interface::AgvcInterface::deleteMapAllInfo (const Header &map_header)

  Deletes all information of the specified map, including map data, stations, paths, and virtual areas.

9.1.1 Detailed Description

Core functions for creating, switching, deleting, reading, and writing AGVC custom maps.

9.1.2 Function Documentation

9.1.2.1 addMapVirtualArea()

int agvc_interface::AgvcInterface::addMapVirtualArea (
const MapVirtualArea & map_virtual_area)

Adds or modifies one virtual area.

Parameters

in	map_virtual_area	Virtual area information to add or modify.

Returns

10100000: virtual area added or modified successfully; else: failed to add or modify the virtual area.

9.1.2.2 addMapVirtualAreas()

int agvc_interface::AgvcInterface::addMapVirtualAreas (
const std::vector< MapVirtualArea > & map_virtual_areas)

Parameters

in	map_virtual_areas	Virtual area information to add or modify.

Returns

10100000: virtual areas added or modified successfully; else: failed to add or modify virtual areas.

9.1.2.3 deleteMap()

int agvc_interface::AgvcInterface::deleteMap (
const Header & map_header)

Deletes one specified map.

Parameters

in	map_header	Header of the specified map (command ID, name, time, map ID).

Returns

10100000: map deleted successfully; else: failed to delete the map.

9.1.2.4 deleteMapAllInfo()

int agvc_interface::AgvcInterface::deleteMapAllInfo (
const Header & map_header)

Deletes all information of the specified map, including map data, stations, paths, and virtual areas.

Parameters

in	map_header	Header of the specified map (command ID, map name, time, map ID).

Returns

10100000: deleted successfully; else: deletion failed.

9.1.2.5 deleteMaps()

int agvc_interface::AgvcInterface::deleteMaps (
const std::vector< Header > & map_headers)

Parameters

in	map_headers	Headers of the specified maps (command ID, name, time, map ID).

Returns

10100000: maps deleted successfully; else: failed to delete maps.

9.1.2.6 deleteMapVirtualArea()

int agvc_interface::AgvcInterface::deleteMapVirtualArea (
const Header & virtual_area_header)

Deletes the specified virtual area.

Parameters

in	virtual_area_header	Header of the specified virtual area (command ID, virtual area name, time, map ID).

Returns

10100000: virtual area deleted successfully; else: failed to delete the virtual area.

9.1.2.7 deleteMapVirtualAreas()

int agvc_interface::AgvcInterface::deleteMapVirtualAreas (
const std::vector< Header > & virtual_areas_header)

Deletes multiple specified virtual areas.

Parameters

in	virtual_areas_header	Headers of the specified virtual areas (command ID, virtual area name, time, map ID).

Returns

10100000: virtual areas deleted successfully; else: failed to delete virtual areas.

9.1.2.8 getAllMapVirtualArea()

std::vector< MapVirtualArea > agvc_interface::AgvcInterface::getAllMapVirtualArea ()

Queries virtual areas on the current map.

Returns

Virtual area information on the current map.

9.1.2.9 getAllMapVirtualAreaOfTargetMap()

std::vector< MapVirtualArea > agvc_interface::AgvcInterface::getAllMapVirtualAreaOfTargetMap (
const Header & map_header)

Queries virtual areas on the target map.

Parameters

in	map_header	Header of the specified map (command ID, name, time, map ID).

Returns

All virtual areas on the specified map.

9.1.2.10 getBase64PngMapFromAgv()

Base64PngMap agvc_interface::AgvcInterface::getBase64PngMapFromAgv (
const Header & map_header)

Gets Base64-encoded PNG map information.

Attention

1. Asynchronous API.

Parameters

in	map_header	Header of the specified map (this asynchronous command ID, name, time, map ID).

Note

When map_id is "current_map", it refers to the current map.

Returns

Base64-encoded PNG map information.

9.1.2.11 getCurrentMapHeader()

Header agvc_interface::AgvcInterface::getCurrentMapHeader ()

Queries the header of the current AGV map.

Returns

Header of the current map (command ID, name, time, map ID).

9.1.2.12 getGridMapAllInfo()

MapAllInfo agvc_interface::AgvcInterface::getGridMapAllInfo (
const Header & map_header)

Gets occupancy grid map data, paths, stations, and virtual areas for the specified map.

Attention

1. Asynchronous API.

Parameters

in	map_header	Header of the specified map (this asynchronous command ID, map name, time, map ID).

Note

When map_id is "current_map", it refers to the current map.

Returns

All information on the occupancy grid map.

9.1.2.13 getGridMapFromAgv()

OccupancyGridMap agvc_interface::AgvcInterface::getGridMapFromAgv (
const Header & map_header)

Gets the specified occupancy grid map information.

Parameters

in	map_header	Header of the specified map (this asynchronous command ID, name, time, map ID).

Note

When map_id is "current_map", it refers to the current map.

Returns

Specified occupancy grid map information.

9.1.2.14 getMapList()

std::vector< Header > agvc_interface::AgvcInterface::getMapList ()

Gets the headers of all maps on the AGV.

Returns

Headers of all maps (command ID, name, time, map ID).

9.1.2.15 getPngMapAllInfo()

MapAllInfo agvc_interface::AgvcInterface::getPngMapAllInfo (
const Header & map_header)

Gets Base64 map data, paths, stations, and virtual areas for the specified map.

Attention

1. Asynchronous API.

Parameters

in	map_header	Header of the specified map (this asynchronous command ID, map name, time, map ID).

Note

When map_id is "current_map", it refers to the current map.

Returns

All information on the Base64 map.

9.1.2.16 previewPngMapFromAgv()

```
Base64PngMap agvc_interface::AgvcInterface::previewPngMapFromAgv (
        const Header & map_header,
        const int & image_width_px = 0,
        const int & image_height_px = 0)
```

Gets a preview image with optional width and height.

Attention

1. Asynchronous API.

Parameters

in	map_header	Header of the specified map (this asynchronous command ID, name, time, map ID).
in	image_width_px	Map width in pixels. If the default value 0 is used, returned thumbnail data is empty.
in	image_height_px	Map height in pixels. If the default value 0 is used, returned thumbnail data is empty.

Note

When map_id is "current_map", it refers to the current map.

Returns

Base64-encoded PNG thumbnail.

9.1.2.17 saveMap()

```
int agvc_interface::AgvcInterface::saveMap (
        const Header & map_header)
```

Saves the map after mapping is complete.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	map_header	Map header (this asynchronous command ID, name, time, map ID).

Returns

10100000: map saved successfully; 10100201: asynchronous API is running; 10120202: no control priority; else: failed to save the map.

9.1.2.18 sendBase64PngMapToAgv()

int agvc_interface::AgvcInterface::sendBase64PngMapToAgv (
const Base64PngMap & map)

Sends a Base64-encoded PNG map to the AGV.

Attention

1. Asynchronous API.

Parameters

in	map	Base64-encoded PNG map.

Returns

10100000: map sent successfully; 10100201: map is being sent; else: failed to send the map.

9.1.2.19 sendGridMapToAgv()

int agvc_interface::AgvcInterface::sendGridMapToAgv (
const OccupancyGridMap & map)

Sends occupancy grid map information to the AGV.

Attention

1. Asynchronous API.

Parameters

in	map	Occupancy grid map information.

Returns

10100000: map sent successfully; 10100201: map is being sent; else: failed to send the map.

9.1.2.20 setGridMapAllInfo()

```
int agvc_interface::AgvcInterface::setGridMapAllInfo (
        const MapAllInfo & map_all_info,
        const Header & command_header = { "99999", "99999", 1, "99999" })
```

Sets occupancy grid map, path, station, and virtual area information.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	map_all_info	All information on the occupancy grid map.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Note

Ensure that the data can be applied; the AGV does not validate the data.

Returns

10100000: all map information uploaded successfully; 10100201: asynchronous API is running; 10120202: no control priority; else: failed to upload all map information.

9.1.2.21 setPngMapAllInfo()

```
int agvc_interface::AgvcInterface::setPngMapAllInfo (
        const MapAllInfo & map_all_info,
        const Header & command_header = { "99999", "99999", 1, "99999" })
```

Sets Base64 map, path, station, and virtual area information.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	map_all_info	All information on the Base64 map.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Note

Ensure that the data can be applied; the AGV does not validate the data.

Returns

10100000: all map information uploaded successfully; 10100201: asynchronous API is running; 10120202: no control priority; else: failed to upload all map information.

9.1.2.22 switchMap()

int agvc_interface::AgvcInterface::switchMap (
const Header & map_header)

Switches to the specified map.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	map_header	Header of the specified map (this asynchronous command ID, name, time, map ID).

Returns

10100000: map switched successfully; 10100201: asynchronous API is running; 10120202: no control priority; else: failed to switch the map.

9.2 Station

Functions

- std::vector< StationMark > agvc_interface::AgvcInterface::getAllStations ()
  Queries all station information on the current map.
- std::vector< StationMark > agvc_interface::AgvcInterface::getAllStationsOfTargetMap (const Header &map_header)
  Queries all station information on the specified map.
- int agvc_interface::AgvcInterface::addStation (const StationMark &station)
  Adds or modifies one station.
- int agvc_interface::AgvcInterface::addStations (const std::vector< StationMark > &stations)
  Adds or modifies multiple stations.
- int agvc_interface::AgvcInterface::addCPStationUseChargingBoard (const Header &station_↩ header, const double &dis_station_board=1.5)
  Adds a charging station by automatically recognizing the charging board pose.
- int agvc_interface::AgvcInterface::deleteStation (const Header &station_header)
  Deletes the specified station and its related paths.
- int agvc_interface::AgvcInterface::deleteStations (const std::vector< Header > &stations_header)
  Deletes multiple stations and their related paths.

9.2.1 Detailed Description

Functions for adding, deleting, modifying, querying AGVC map stations, and auto-detecting charging stations.

9.2.2 Function Documentation

9.2.2.1 addCPStationUseChargingBoard()

int agvc_interface::AgvcInterface::addCPStationUseChargingBoard (
const Header & station_header,
const double & dis_station_board = 1.5)

Adds a charging station by automatically recognizing the charging board pose.

Since

0.7.0

Parameters

in	station_header	Charging station information to add or modify (station ID, name, time, map ID).
in	dis_station_board	Distance between the charging station and charging board. The charging station is at the AGV center. Unit: m; range: [1.0, 2.0].

Returns

10100000: charging station added successfully; else: failed to add the charging station.

9.2.2.2 addStation()

int agvc_interface::AgvcInterface::addStation (
const StationMark & station)

Adds or modifies one station.

Parameters

in	station	Station information to add or modify.

Returns

10100000: station added or modified successfully; else: failed to add or modify the station.

9.2.2.3 addStations()

int agvc_interface::AgvcInterface::addStations (
const std::vector< StationMark > & stations)

Adds or modifies multiple stations.

Parameters

in	stations	Station information to add or modify.

Returns

10100000: stations added or modified successfully; else: failed to add or modify stations.

9.2.2.4 deleteStation()

int agvc_interface::AgvcInterface::deleteStation (
const Header & station_header)

Deletes the specified station and its related paths.

Parameters

in	station_header	Station information to delete (station ID, name, time, map ID).

Returns

Parameters

in	stations_header	Station information to delete (station ID, name, time, map ID).

Returns

10100000: stations deleted successfully; else: failed to delete stations.

9.2.2.6 getAllStations()

std::vector< StationMark > agvc_interface::AgvcInterface::getAllStations ()

Queries all station information on the current map.

Returns

All station information on the current map.

9.2.2.7 getAllStationsOfTargetMap()

std::vector< StationMark > agvc_interface::AgvcInterface::getAllStationsOfTargetMap (
const Header & map_header)

Queries all station information on the specified map.

Parameters

in	map_header	Header of the specified map (command ID, name, time, map ID).

Returns

All station information on the specified map.

9.3 Path

Functions

- std::vector< PathStation > agvc_interface::AgvcInterface::getAllPaths ()
  Queries all path information on the current map.
- std::vector< PathStation > agvc_interface::AgvcInterface::getAllPathsOfTargetMap (const Header &map_header)
  Queries all path information on the specified map.
- PathStation agvc_interface::AgvcInterface::getCurrentPath ()
  Queries the path currently being tracked by the AGV.
- int agvc_interface::AgvcInterface::generatePath (const PathStation &path_station)
  Generates or modifies one path.
- int agvc_interface::AgvcInterface::generatePaths (const std::vector< PathStation > &paths_station)
  Generates or modifies multiple paths.
- int agvc_interface::AgvcInterface::deletePath (const Header &path_header)
  Deletes one path.
- int agvc_interface::AgvcInterface::deletePaths (const std::vector< Header > &paths_header)
  Deletes multiple paths.

9.3.1 Detailed Description

Functions for generating, modifying, deleting, and querying AGVC map paths.

9.3.2 Function Documentation

9.3.2.1 deletePath()

int agvc_interface::AgvcInterface::deletePath (
const Header & path_header)

Deletes one path.

Parameters

in	path_header	Path information to delete (path ID, name, time, map ID).

Returns

10100000: path deleted successfully; else: failed to delete the path.

9.3.2.2 deletePaths()

int agvc_interface::AgvcInterface::deletePaths (
const std::vector< Header > & paths_header)

Deletes multiple paths.

Parameters

in	paths_header	Path information to delete (path ID, name, time, map ID).

Returns

10100000: paths deleted successfully; else: failed to delete paths.

9.3.2.3 generatePath()

int agvc_interface::AgvcInterface::generatePath (
const PathStation & path_station)

Generates or modifies one path.

Parameters

in	path_station	Path information to generate or modify.

Returns

10100000: path modified or generated successfully; else: failed to modify or generate the path.

9.3.2.4 generatePaths()

int agvc_interface::AgvcInterface::generatePaths (
const std::vector< PathStation > & paths_station)

Generates or modifies multiple paths.

Parameters

in	paths_station	Path information to generate or modify.

Returns

10100000: paths modified or generated successfully; else: failed to modify or generate paths.

9.3.2.5 getAllPaths()

std::vector< PathStation > agvc_interface::AgvcInterface::getAllPaths ()

Queries all path information on the current map.

Returns

All path information on the current map.

9.3.2.6 getAllPathsOfTargetMap()

std::vector< PathStation > agvc_interface::AgvcInterface::getAllPathsOfTargetMap (
const Header & map_header)

Queries all path information on the specified map.

Parameters

in	map_header	Header of the specified map (command ID, name, time, map ID).

Returns

All path information on the specified map.

9.3.2.7 getCurrentPath()

PathStation agvc_interface::AgvcInterface::getCurrentPath ()

Queries the path currently being tracked by the AGV.

Returns

Information about the path currently being tracked.

9.4 Navigation

Functions

- int agvc_interface::AgvcInterface::changeRunningMode (const RunningMode &running_mode, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })
  Switches the AGV running mode.
- int agvc_interface::AgvcInterface::setControlSpeed (const Speed &speed)
  Controls AGV motion.
- int agvc_interface::AgvcInterface::setNavGoal (const NavGoalType &target)
  Sends a navigation goal to the AGV.
- int agvc_interface::AgvcInterface::pauseAgvSpeed ()
  Pauses AGV speed.
- int agvc_interface::AgvcInterface::resumeAgvSpeed ()
  Resumes AGV speed after a pause.
- int agvc_interface::AgvcInterface::cancelNavigation ()
  Cancels the navigation task.
- int agvc_interface::AgvcInterface::relocation (const Pose2d &init_pose, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })
  Relocates the AGV.

9.4.1 Detailed Description

Functions for AGVC motion control, navigation goals, relocation, and mode switching.

9.4.2 Function Documentation

9.4.2.1 cancelNavigation()

int agvc_interface::AgvcInterface::cancelNavigation ()

Cancels the navigation task.

Attention

Requires control priority.

Returns

10100000: navigation task canceled successfully; 10120202: no control priority; else: failed to cancel the navigation task.

9.4.2.2 changeRunningMode()

int agvc_interface::AgvcInterface::changeRunningMode (
const RunningMode & running_mode,
const Header & command_header = { ”99999”, ”99999”, 1, ”99999” })

Switches the AGV running mode.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	running_mode	Target mode to switch to, such as navigation mode or mapping mode.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Returns

10100000: mode switched successfully; 10100201: asynchronous API is running; 10120202: no control priority; else: failed to switch mode.

9.4.2.3 pauseAgvSpeed()

int agvc_interface::AgvcInterface::pauseAgvSpeed ()

Pauses AGV speed.

Attention

Requires control priority.

Returns

10100000: paused successfully; 10120202: no control priority; else: pause failed.

9.4.2.4 relocation()

int agvc_interface::AgvcInterface::relocation (
const Pose2d & init_pose,
const Header & command_header = { ”99999”, ”99999”, 1, ”99999” })

Relocates the AGV.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	init_pose	Current reference pose of the AGV.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Returns

10100000: relocation succeeded; 10100201: asynchronous API is running; 10120202: no control priority; else: relocation failed.

9.4.2.5 resumeAgvSpeed()

int agvc_interface::AgvcInterface::resumeAgvSpeed ()

Resumes AGV speed after a pause.

Attention

Requires control priority.

Returns

10100000: resumed successfully; 10120202: no control priority; else: resume failed.

9.4.2.6 setControlSpeed()

int agvc_interface::AgvcInterface::setControlSpeed (
const Speed & speed)

Controls AGV motion.

Attention

Requires control priority.

Parameters

in	speed	Speed information.

Returns

10100000: speed command sent to the AGV successfully; 10120202: no control priority; else: failed to send the speed command.

9.4.2.7 setNavGoal()

int agvc_interface::AgvcInterface::setNavGoal (
const NavGoalType & target)

Sends a navigation goal to the AGV.

Attention

Requires control priority.

Parameters

in	target	Target pose information and navigation parameters.

Returns

10100000: navigation goal set successfully; 10120202: no control priority; else: failed to set the navigation goal.

9.5 Charging

Functions

- int agvc_interface::AgvcInterface::checkPowerAndAutoCharge (const Header &check_power_↩ header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &nav_board_charge_station↩ _id=”99999”)

  Checks the current battery level and automatically docks for charging when the battery is low.
- int agvc_interface::AgvcInterface::setLeaveBoardTargetStation (const Header &leave_board_↩ target_station_header)

  Sets the target station for automatic undocking.
- Header agvc_interface::AgvcInterface::getLeaveBoardTargetStation ()

  Gets the target station for automatic undocking.
- int agvc_interface::AgvcInterface::forcedAgvToChargingBoard (const Header &forced_charge_↩ header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &nav_board_charge_station_↩ id=”99999”)

  Forces the AGV to dock for charging.
- int agvc_interface::AgvcInterface::forcedAgvLeaveChargingBoard (const Header &forced_leave_↩ header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &leave_board_target_station_↩ id=”99999”)

  Forces the AGV to leave the charging board.
- int agvc_interface::AgvcInterface::sendAutoChargingCommand (const AutoChargingCommand &auto_charging_command)

  Sends an automatic charging command.

9.5.1 Detailed Description

AGVC charging functions, including auto charging, forced docking or undocking, and battery checks.

9.5.2 Function Documentation

9.5.2.1 checkPowerAndAutoCharge()

```
int agvc_interface::AgvcInterface::checkPowerAndAutoCharge (
        const Header & check_power_header = { "99999", "99999", 99999, "99999" },
        const std::string & nav_board_charge_station_id = "99999")
```

Checks the current battery level and automatically docks for charging when the battery is low.

Attention

Requires control priority.

Note

After charging completes, the auto_leave_board parameter decides whether the AGV automatically leaves the board and moves to a target station or waiting area.

Configure the low battery threshold [low_battery_threshold] or high battery threshold [high_↩ battery_threshold] in the configuration file.

Parameters

in	check_power_header	ID of this command, name of this command, current time, and map ID.
in	nav_board_charge_station_id	Drives to the specified charging station for docking and charging. If unset, drives to the nearest charging station.

Returns

10100000: API call succeeded and the AGV starts the charging command; 10120202: no control priority; else: API call failed.

9.5.2.2 forcedAgvLeaveChargingBoard()

```
int agvc_interface::AgvcInterface::forcedAgvLeaveChargingBoard (
        const Header & forced_leave_header = { "99999", "99999", 99999, "99999" },
        const std::string & leave_board_target_station_id = "99999")
```

Forces the AGV to leave the charging board.

Attention

Requires control priority.

Parameters

in	forced_leave_header	ID of this command, name of this command, current time, and map ID.
in	leave_board_target_station_id	Station to move to after leaving the board. If unset, the AGV leaves the board to the charging station.

Returns

10100000: API call succeeded and the AGV starts leaving the board; 10120202: no control priority; else: API call failed.

9.5.2.3 forcedAgvToChargingBoard()

int agvc_interface::AgvcInterface::forcedAgvToChargingBoard (
const Header & forced_charge_header = { ”99999”, ”99999”, 99999, ”99999” },
const std::string & nav_board_charge_station_id = ”99999”)

Forces the AGV to dock for charging.

Attention

Requires control priority.

Parameters

in	forced_charge_header	ID of this command, name of this command, current time, and map ID.
in	nav_board_charge_station_id	Drives to the specified charging station for docking and charging. If unset, drives to the nearest charging station.

Returns

10100000: API call succeeded and the AGV starts docking for charging; 10120202: no control priority; else: API call failed.

9.5.2.4 getLeaveBoardTargetStation()

Header agvc_interface::AgvcInterface::getLeaveBoardTargetStation ()

Gets the target station for automatic undocking.

Returns

Header information of the target station for automatic undocking.

9.5.2.5 sendAutoChargingCommand()

int agvc_interface::AgvcInterface::sendAutoChargingCommand (
const AutoChargingCommand & auto_charging_command)

Sends an automatic charging command.

Attention

Requires control priority.

Since

0.7.0

Parameters

in	auto_charging_command	Automatic charging command.

Returns

10100000: API call succeeded and the AGV starts the charging command; 10120202: no control priority; else: API call failed.

9.5.2.6 setLeaveBoardTargetStation()

int agvc_interface::AgvcInterface::setLeaveBoardTargetStation (
const Header & leave_board_target_station_header)

Sets the target station for automatic undocking.

Parameters

in	leave_board_target_station_header	Header information of the station used for automatic undocking.

Returns

10100000: automatic undocking station set successfully; else: failed to set the station.

9.6 AgvInfo

Functions

- AgvDetails agvc_interface::AgvcInterface::getAgvDetails ()
  Queries current AGV information.
- RunningInfo agvc_interface::AgvcInterface::getRunningInfo ()
  Queries current AGV running information.
- Pose2d agvc_interface::AgvcInterface::getAgvCurrentPose ()
  Queries the current AGV pose.
- std::vector< Point2d > agvc_interface::AgvcInterface::getLaserPointCloud ()
  Queries the current laser point cloud data.
- NavInfo agvc_interface::AgvcInterface::getNavInfo ()
  Gets current navigation information.
- std::string agvc_interface::AgvcInterface::getAgvLogMessage ()
  Gets AGV runtime log messages.
- ScriptRuntimeState agvc_interface::AgvcInterface::getScriptStatus ()
  Gets the script runtime state.
- std::string agvc_interface::AgvcInterface::errorCodeDecoder (const int &error_code)
  Queries an error code.
- std::vector< Header > agvc_interface::AgvcInterface::getCurrentErrorCodes ()
  Gets current error codes.
- int agvc_interface::AgvcInterface::soundLightPrompt ()
  Locates the AGV by automatically playing audio and flashing special lights.
- bool agvc_interface::AgvcInterface::isObstacleAhead (const double &detect_distance=1.0)
  Checks whether there is an obstacle in the AGV travel direction.
- StationMark agvc_interface::AgvcInterface::getNearestStation (const double &detect_distance=1.↩ 0)
  Gets the nearest station within the specified detection distance.

9.6.1 Detailed Description

Query functions for AGVC status, pose, logs, error codes, and related information.

9.6.2 Function Documentation

9.6.2.1 errorCodeDecoder()

std::string agvc_interface::AgvcInterface::errorCodeDecoder (
const int & error_code)

Queries an error code.

Parameters

in	error_code	Error code.

Returns

Meaning of the error code. Empty means the error code is unknown.

9.6.2.2 getAgvCurrentPose()

Pose2d agvc_interface::AgvcInterface::getAgvCurrentPose ()

Queries the current AGV pose.

Attention

RTDE pushed data.

Returns

Current pose.

9.6.2.3 getAgvDetails()

AgvDetails agvc_interface::AgvcInterface::getAgvDetails ()

Queries current AGV information.

Attention

RTDE pushed data.

Returns

Detailed AGV information.

9.6.2.4 getAgvLogMessage()

std::string agvc_interface::AgvcInterface::getAgvLogMessage ()

Gets AGV runtime log messages.

Returns

AGV runtime log messages.

9.6.2.5 getCurrentErrorCodes()

std::vector< Header > agvc_interface::AgvcInterface::getCurrentErrorCodes ()

Gets current error codes.

Returns

Error code collection.

9.6.2.6 getLaserPointCloud()

std::vector< Point2d > agvc_interface::AgvcInterface::getLaserPointCloud ()

Queries the current laser point cloud data.

Attention

RTDE pushed data.

Returns

2D laser point coordinates.

9.6.2.7 getNavInfo()

NavInfo agvc_interface::AgvcInterface::getNavInfo ()

Gets current navigation information.

Attention

RTDE pushed data.

Returns

Current navigation information.

9.6.2.8 getNearestStation()

StationMark agvc_interface::AgvcInterface::getNearestStation (
const double & detect_distance = 1.0)

Gets the nearest station within the specified detection distance.

Parameters

in	detect_distance	Detection distance. Default: 1.0 m.

Returns

Nearest station information. id NONE means no station is within the detection distance.

9.6.2.9 getRunningInfo()

RunningInfo agvc_interface::AgvcInterface::getRunningInfo ()

Queries current AGV running information.

Attention

RTDE pushed data.

Returns

AGV running information.

9.6.2.10 getScriptStatus()

ScriptRuntimeState agvc_interface::AgvcInterface::getScriptStatus ()

Gets the script runtime state.

Returns

Script runtime state.

9.6.2.11 isObstacleAhead()

bool agvc_interface::AgvcInterface::isObstacleAhead (
const double & detect_distance = 1.0)

Checks whether there is an obstacle in the AGV travel direction.

Parameters

in	detect_distance	Detection distance. Default: 1.0 m.

Returns

true: obstacle exists in the travel direction; false: no obstacle in the travel direction.

9.6.2.12 soundLightPrompt()

int agvc_interface::AgvcInterface::soundLightPrompt ()

Locates the AGV by automatically playing audio and flashing special lights.

Returns

10100000: locate command sent successfully; else: failed to send the command.

9.7 System

Functions

- int agvc_interface::AgvcInterface::setSystemClock (const std::string &stamp)

  Sets the AGV system time.
- AsyncInterfaceResultStatus agvc_interface::AgvcInterface::getAsyncInterfaceResultStatus ()

  Queries the running status of asynchronous APIs; this does not represent each API's own execution result.
- std::vector< std::string > agvc_interface::AgvcInterface::getWifiList ()

  Gets the currently scanned WiFi list.
- int agvc_interface::AgvcInterface::connectWifi (const std::string &ssid, const std::string &password)

  Connects to the specified WiFi and automatically switches to WiFi mode.
- int agvc_interface::AgvcInterface::enableHotspot (const std::string &password)

  Enables hotspot mode and automatically switches to hotspot mode.
- std::vector< IpAddressInfo > agvc_interface::AgvcInterface::getIpAddressList ()

  Gets all local IP addresses.
- std::string agvc_interface::AgvcInterface::getAgvControllerParametersFile ()

  Gets the AGV controller parameter file.
- int agvc_interface::AgvcInterface::setAgvControllerParametersFile (const std::string &agv_↩ parameters)

  Sets the AGV controller parameter file.
- int agvc_interface::AgvcInterface::refreshAgvControllerParametersFile ()

  Refreshes parameters on AGV nodes after the AGV controller parameter file is modified.
- int agvc_interface::AgvcInterface::resetAgvControllerParametersFile ()

  Restores the AGV controller parameter file to factory defaults.
- int agvc_interface::AgvcInterface::updateSoftware (const std::string &upgrade_pack_path)

  Upgrades AGV software.
- std::vector< std::string > agvc_interface::AgvcInterface::getSoftwareVersionList ()

  Gets the AGV software version list.
- int agvc_interface::AgvcInterface::switchSoftwareVersion (const std::string &software_version)

  Switches the AGV software version.
- int agvc_interface::AgvcInterface::uninstallSoftware (const std::string &software_version)

  Uninstalls an AGV software version.
- int agvc_interface::AgvcInterface::updateFirmware (const FirmwareUpdateParam &update_↩ firmware)

  Updates AGV firmware.
- FirmwareUpdateProcessInfo agvc_interface::AgvcInterface::getUpdateFirmwareProcess ()

  Gets firmware update process information.
- int agvc_interface::AgvcInterface::startCollectCalibrationData (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Starts collecting calibration data.
- int agvc_interface::AgvcInterface::cancelCollectCalibrationData (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Cancels calibration data collection.
- int agvc_interface::AgvcInterface::startCalibration (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Starts calibration.
- CalibrationProcessInfo agvc_interface::AgvcInterface::getCalibrationProcessInfo (const CalibrationType &type)

  Gets calibration information.

- int agvc_interface::AgvcInterface::restartAgv ()
  Restarts the AGV.
- int agvc_interface::AgvcInterface::setAgvName (const std::string &agv_name)
  Sets the AGV name.
- int agvc_interface::AgvcInterface::setPriority (const std::string &name, const std::string &ip="")
  Sets control priority.
- int agvc_interface::AgvcInterface::releasePriority ()
  Releases control priority.
- int agvc_interface::AgvcInterface::scriptPaused ()
  Pauses the script.
- int agvc_interface::AgvcInterface::scriptResume ()
  Resumes the script.
- int agvc_interface::AgvcInterface::scriptStop ()
  Stops the script.
- int agvc_interface::AgvcInterface::setScriptStatus (ScriptRuntimeState status)
  Sets the script runtime state.
- int agvc_interface::AgvcInterface::autoAlignRailway (const Header &command_header={ "99999", "99999", 1, "99999" })
  Automatically aligns with the rail and boards the rail.

9.7.1 Detailed Description

System-level functions for AGVC configuration, control priority, software upgrades, and firmware updates.

9.7.2 Function Documentation

9.7.2.1 autoAlignRailway()

int agvc_interface::AgvcInterface::autoAlignRailway (
const Header & command_header = { "99999", "99999", 1, "99999" })

Automatically aligns with the rail and boards the rail.

Attention

1. Asynchronous API; 2. Requires control priority.

Parameters

in	command_header	The id field is the ID of this command; other fields have no special meaning.

Returns

10100000: rail boarding succeeded; 10100201: rail boarding is in progress; 10120202: no control priority; else: rail boarding failed.

9.7.2.2 cancelCollectCalibrationData()

Parameters

in	type	Calibration type.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Returns

10100000: API call succeeded; 10120202: no control priority; else: API call failed.

9.7.2.3 connectWifi()

```
int agvc_interface::AgvcInterface::connectWifi (
        const std::string & ssid,
        const std::string & password)
```

Connects to the specified WiFi and automatically switches to WiFi mode.

Note

Must be called over a wired connection and may take a long time.

Parameters

in	ssid	WiFi SSID.
in	password	WiFi password.

Returns

10100000: connected to WiFi successfully; else: failed to connect to WiFi.

9.7.2.4 enableHotspot()

```
int agvc_interface::AgvcInterface::enableHotspot (
        const std::string & password)
```

Enables hotspot mode and automatically switches to hotspot mode.

Note

Must be called over a wired connection and may take a long time.

Parameters

in	password	Hotspot password. If empty, ”administrator” is used as the default password, and the SSID defaults to the SN.

Returns

10100000: hotspot enabled successfully; else: failed to enable hotspot.

9.7.2.5 getAgvControllerParametersFile()

std::string agvc_interface::AgvcInterface::getAgvControllerParametersFile ()

Gets the AGV controller parameter file.

Returns

Parameter information in the current AGV controller.

9.7.2.6 getAsyncInterfaceResultStatus()

AsyncInterfaceResultStatus agvc_interface::AgvcInterface::getAsyncInterfaceResultStatus ()

Queries the running status of asynchronous APIs; this does not represent each API's own execution result.

Note

Asynchronous APIs: saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMap↩ ToAgv.

Asynchronous APIs: getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMap↩ FromAgv.

Asynchronous APIs: getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, preview↩ PngMapFromAgv.

Asynchronous API: autoAlignRailway.

After the client disconnects from the AGV, call this API to obtain asynchronous API status.

Each field in the return value contains a header, where header.id is the ID used when issuing the asynchronous task.

Returns

NONE: asynchronous API is not running; RUNNING: asynchronous API is running; FINISH:↩ asynchronous API has finished.

9.7.2.7 getCalibrationProcessInfo()

CalibrationProcessInfo agvc_interface::AgvcInterface::getCalibrationProcessInfo (
const CalibrationType & type)

Gets calibration information

Parameters

in	type	Calibration type.

Returns

Calibration information.

9.7.2.8 getIpAddressList()

std::vector< IpAddressInfo > agvc_interface::AgvcInterface::getIpAddressList ()

Gets all local IP addresses.

Returns

All local IP addresses.

9.7.2.9 getSoftwareVersionList()

std::vector< std::string > agvc_interface::AgvcInterface::getSoftwareVersionList ()

Gets the AGV software version list.

Returns

Software version list, for example {0.3.2+3b807d1-Linux_x86_64,0.6.3-patch.3+b8c6aa4-Linux_↩ x86_64}.

9.7.2.10 getUpdateFirmwareProcess()

FirmwareUpdateProcessInfo agvc_interface::AgvcInterface::getUpdateFirmwareProcess ()

Gets firmware update process information.

Returns

Firmware update step information (failed means the update failed) and update progress (1.0 means the update succeeded).

9.7.2.11 getWifiList()

std::vector< std::string > agvc_interface::AgvcInterface::getWifiList ()

Gets the currently scanned WiFi list.

Attention

Asynchronous API.

Returns

{RUNNING}: asynchronous API is running; {}: empty, no WiFi found; {SSID}: non-empty list of scanned WiFi SSIDs.

9.7.2.12 refreshAgvControllerParametersFile()

int agvc_interface::AgvcInterface::refreshAgvControllerParametersFile ()

Refreshes parameters on AGV nodes after the AGV controller parameter file is modified.

Attention

Requires control priority.

Note

Do not call frequently. Keep the call rate below 1 Hz.

Returns

10100000: API executed successfully; else: API execution failed.

9.7.2.13 releasePriority()

int agvc_interface::AgvcInterface::releasePriority ()

Releases control priority.

Returns

10100000: control priority released successfully; else: failed to release control priority.

9.7.2.14 resetAgvControllerParametersFile()

int agvc_interface::AgvcInterface::resetAgvControllerParametersFile ()

Restores the AGV controller parameter file to factory defaults.

Attention

Requires control priority.

Note

Do not call frequently. Keep the call rate below 1 Hz.

Returns

10100000: API executed successfully; else: API execution failed.

9.7.2.15 restartAgv()

int agvc_interface::AgvcInterface::restartAgv ()

Restarts the AGV.

Returns

10100000: restart command executed successfully; else: restart command failed.

9.7.2.16 scriptPaused()

int agvc_interface::AgvcInterface::scriptPaused ()

Pauses the script.

Returns

10100000: script paused successfully; else: failed to pause the script.

9.7.2.17 scriptResume()

int agvc_interface::AgvcInterface::scriptResume ()

Resumes the script.

Returns

10100000: script resumed successfully; else: failed to resume the script.

9.7.2.18 scriptStop()

int agvc_interface::AgvcInterface::scriptStop ()

Stops the script.

Returns

10100000: script stopped successfully; else: failed to stop the script.

9.7.2.19 setAgvControllerParametersFile()

int agvc_interface::AgvcInterface::setAgvControllerParametersFile (
const std::string & agv_parameters)

Sets the AGV controller parameter file.

Attention

Requires control priority.

Note

Do not call frequently. Keep the call rate below 1 Hz. This API refreshes parameters by default.

Parameters

in	agv_parameters	Parameter information for the AGV controller.

Returns

10100000: parameters set successfully; else: failed to set parameters.

9.7.2.20 setAgvName()

int agvc_interface::AgvcInterface::setAgvName (
const std::string & agv_name)

Sets the AGV name.

Parameters

in	agv_name	AGV name to set.

Returns

10100000: name set successfully; else: failed to set the name.

9.7.2.21 setPriority()

```
int agvc_interface::AgvcInterface::setPriority (
        const std::string & name,
        const std::string & ip = "")
```

Sets control priority.

Note

1. Control-priority APIs can be used only after this API succeeds; 2. Release another client's control priority with releasePriority before setting priority; 3. To preempt priority from the web side, set name="web".

Parameters

in	name	User name registered during RPC login.
in	ip	IP address.

Returns

10100000: control priority set successfully; else: failed to set control priority.

9.7.2.22 setScriptStatus()

```
int agvc_interface::AgvcInterface::setScriptStatus (
        ScriptRuntimeState status)
```

Sets the script runtime state.

Returns

10100000: script runtime state set successfully; else: failed to set script runtime state.

9.7.2.23 setSystemClock()


```
int agvc_interface::AgvcInterface::setSystemClock (
        const std::string & stamp)
```

Parameters

in	stamp	Timestamp in YYYY-MM-DDTHH:mm:ss.ssZ format, for example "2017-04-15T11:40:03.12Z".

9.7.2.24 startCalibration()

```
int agvc_interface::AgvcInterface::startCalibration (
        const CalibrationType & type,
        const Header & command_header = { "99999", "99999", 1, "99999" })
```

Starts calibration.

Parameters

in	type	Calibration type.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Returns

10100000: API call succeeded; 10120202: no control priority; else: API call failed.

9.7.2.25 startCollectCalibrationData()

```
int agvc_interface::AgvcInterface::startCollectCalibrationData (
        const CalibrationType & type,
        const Header & command_header = { "99999", "99999", 1, "99999" })
```

Starts collecting calibration data.

Note

LASER_ODOM calibration requires more than 1000 data samples. The exact amount can be changed in the configuration file.

Parameters

in	type	Calibration type.
in	command_header	The id field is the ID of this command; other fields have no special meaning.

Returns

10100000: API call succeeded; 10120202: no control priority; else: API call failed.

Parameters

in	software_version	Software version, for example "0.3.2+3b807d1-Linux_x86_64".

Returns

10100000: software version switched successfully; else: failed to switch software version.

9.7.2.27 uninstallSoftware()

```
int agvc_interface::AgvcInterface::uninstallSoftware (
        const std::string & software_version)
```

Uninstalls an AGV software version.

Parameters

in	software_version	Software version, for example "0.3.2+3b807d1-Linux_x86_64".

Returns

10100000: software version uninstalled successfully; else: failed to uninstall software version.

9.7.2.28 updateFirmware()

```
int agvc_interface::AgvcInterface::updateFirmware (
        const FirmwareUpdateParam & update_firmware)
```

Updates AGV firmware.

Parameters

in	update_firmware	Configures the firmware modules to update and the firmware storage path.

Returns

10100000: firmware update information sent successfully; else: firmware update is in progress.

9.7.2.29 updateSoftware()


```
int agvc_interface::AgvcInterface::updateSoftware (
        const std::string & upgrade_pack_path)
```

Parameters

in	upgrade_pack_path	Upgrade package path, for example "/root/agvc_release-0.1.1+398d1f7-Linux_x86_64.tar.gz".

Returns

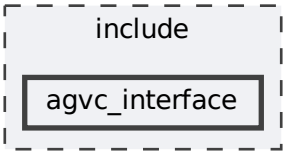
10100000: target version upgraded successfully; 10100201: upgrade is in progress; else: upgrade failed.

Chapter 10

Directory Documentation

10.1 include/agvc_interface Directory Reference

Directory dependency graph for agvc_interface:



Files

- file agvc_interface.h
- file rpc.h
  RPC
- file rtde.h
  RPC
- file script.h
  SCRIPT
- file type.h

10.2 doc Directory Reference

10.3 include Directory Reference

Directories

- directory agvc_interface

Chapter 11

Namespace Documentation

11.1 agvc_interface Namespace Reference

Classes

- class AgvcInterface
- class RpcClient
  RPC
- class OutputBuilder
- class InputParser
- class RtdeClient
  RTDE
- class ScriptWriter
- class ScriptClient
  SCRIPT
- struct Point2d
- struct Pose2d
- struct Speed
- struct Header
- struct AgvDetails
  agv
- struct RunningInfo
  agv
- struct SaveMapWorkingStatus
  saveMap
- struct SwitchMapWorkingStatus
  switchMap
- struct ChangeModeWorkingStatus
  changeRunningMode
- struct RelocationWorkingStatus

Relocation

- struct SetPngMapAllInfoWorkingStatus

  setPngMapAllInfo
- struct GetPngMapAllInfoWorkingStatus

  getPngMapAllInfo
- struct GetGridMapWorkingStatus

  getGridMapFromAgv
- struct SendGridMapWorkingStatus

  sendGridMapToAgv
- struct GetPngMapWorkingStatus
- struct SendPngMapWorkingStatus

  sendBase64PngMapToAgv
- struct SetGridMapAllInfoWorkingStatus

  setGridMapAllInfo
- struct GetGridMapAllInfoWorkingStatus

  getGridMapAllInfo
- struct PreviewPngMapWorkingStatus

  previewPngMapFromAgv
- struct AutoAlignRailwayWorkingStatus

  autoAlignRailway
- struct AsyncInterfaceResultStatus

  14 saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, previewPngMapFromAgv, autoAlignRailway
- struct StationMark
- struct PathStation
- struct MapInfo
- struct MapVirtualArea
- struct MapAllInfo
- struct IpAddressInfo

  IP
- struct AdjustParam

  agv
- struct NavGoalType

  agv
- struct NavInfo

  /
- struct AutoChargingCommand
- struct FirmwareUpdateParam
- struct FirmwareUpdateProcessInfo
- struct CalibrationProcessInfo
- struct RtdeRecipe

  RTDE
- class AgvcException

Typedefs

- using AgvcInterfacePtr = std::shared_ptr<AgvcInterface>
- using RpcClientPtr = std::shared_ptr<RpcClient>
- using RtdeClientPtr = std::shared_ptr<RtdeClient>
- using ScriptClientPtr = std::shared_ptr<ScriptClient>
- typedef FeedbackStatus NavStatus
  agv
- typedef FeedbackStatus SaveMapStatus
  saveMap
- typedef FeedbackStatus SwitchMapStatus
  switchMap
- typedef FeedbackStatus ChangeRunningModeStatus
  changeRunningMode
- typedef FeedbackStatus RelocationStatus
  relocation
- typedef FeedbackStatus SetPngMapAllInfoStatus
  Png setPngMapAllInfo
- typedef FeedbackStatus GetPngMapAllInfoStatus
  Png getPngMapAllInfo
- typedef FeedbackStatus GetGridMapStatus
  getGridMapFromAgv
- typedef FeedbackStatus SendGridMapStatus
  sendGridMapToAgv
- typedef FeedbackStatus GetPngMapStatus
  png getBase64PngMapFromAgv
- typedef FeedbackStatus SendPngMapStatus
  png sendBase64PngMapToAgv
- typedef FeedbackStatus SetGridMapAllInfoStatus
  setGridMapAllInfo
- typedef FeedbackStatus GetGridMapAllInfoStatus
  getGridMapAllInfo
- typedef FeedbackStatus PreviewPngMapStatus
  previewPngMapFromAgv
- typedef FeedbackStatus AutoAlignRailwayStatus
  autoAlignRailway
- typedef FeedbackStatus CalibrationStatus
- typedef MapInfo OccupancyGridMap
- typedef MapInfo Base64PngMap
  Base64 png

Enumerations

- enum AgvcErrorCodes : int { ENUM_AgvcErrorCodes_DECLARES }
- enum class RunningMode {
  NONE = 0 , START = 1 , MAPPING = 2 , NAVIGATE = 3 ,
  CHARGING = 4 , MAINTAIN = 5 , STOP = 6 }
  agv
- enum class PathShape { NONE = 0 , LINE = 1 , ARC = 2 , BEZIER = 3 }

- enum class MapFormat { NONE = 0 , OCCUPANCY_GRID = 1 , BASE64_PNG = 2 }
- enum class MapVirtualAreaType { NONE = 0 , OBSTACLE = 1 , SLOW_DOWN = 2 , INFLATE = 3 }
- enum class MapVirtualAreaShape {
  NONE = 0 , LINE = 1 , ARC = 2 , CIRCLE = 3 ,
  POLYGON = 4 }
- enum class NavType { NONE = 0 , FREE_TO_POSE = 1 , FREE_TO_STATION = 2 , PATH_TO_STATION = 3 }

  agv
- enum class AutoChargingType {
  NONE = 0 , LOW_POWER_TO_BOARD = 1 , HIGH_POWER_LEAVE_BOARD = 2 ,
  FORCE_TO_BOARD = 3 ,
  FORCE_LEAVE_BOARD = 4 }
- enum class FeedbackStatus {
  NONE = 0 , FINISHED = 1 , FAILED = 2 , RUNNING = 3 ,
  PAUSED = 4 , CANCELED = 5 }
- enum class FirmwareUpdateMode { NONE = 0 , UPDATE = 1 , FORCED_UPDATE = 2 , NOT_UPDATE = 3 }
- enum class ScriptRuntimeState { RUNNING = 0 , PAUSED = 1 , STOPPED = 2 , ABORTING = 3 }
- enum class CalibrationType { NONE = 0 , LASER_ODOM = 1 }
- enum error_type {
  parse_error = -32700 , invalid_request = -32600 , method_not_found = -32601 , invalid_params = -32602 ,
  internal_error = -32603 , server_error , invalid }
- enum ExceptionCode {
  EC_DISCONNECTED = -1 , EC_NOT_LOGINED = -2 , EC_INVAL_SOCKET = -3 , EC_REQUEST_BUSY = -4 ,
  EC_SEND_FAILED = -5 , EC_RECV_TIMEOUT = -6 , EC_RECV_ERROR = -7 , EC_PARSE_ERROR = -8 ,
  EC_INVALID_REQUEST = -9 , EC_METHOD_NOT_FOUND = -10 , EC_INVALID_PARAMS = -11 , EC_INTERNAL_ERROR = -12 ,
  EC_SERVER_ERROR = -13 , EC_INVALID = -14 }

Functions

- const char * returnValue2Str (int retval)

11.1.1 Typedef Documentation

11.1.1.1 AgvcInterfacePtr

using agvc_interface::AgvcInterfacePtr = std::shared_ptr<AgvcInterface>

Definition at line 1492 of file agvc_interface.h.

11.1.1.2 AutoAlignRailwayStatus

typedef FeedbackStatus agvc_interface::AutoAlignRailwayStatus

autoAlignRailway

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 380 of file type.h.

11.1.1.3 Base64PngMap

typedef MapInfo agvc_interface::Base64PngMap

Base64 png

Header header; // id map_id id
double resolution; // m
uint32_t width; //
uint32_t height; //
Pose2d origin; // {x(m) y(m) theta(rad)} // //
MapFormat format; // =MapFormat::BASE64_PNG
std::vector<int8_t> data; // Base64 png data string

Definition at line 706 of file type.h.

11.1.1.4 CalibrationStatus

typedef FeedbackStatus agvc_interface::CalibrationStatus

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 392 of file type.h.

11.1.1.5 ChangeRunningModeStatus

typedef FeedbackStatus agvc_interface::ChangeRunningModeStatus

changeRunningMode

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 248 of file type.h.

11.1.1.6 GetGridMapAllInfoStatus

typedef FeedbackStatus agvc_interface::GetGridMapAllInfoStatus

getGridMapAllInfo

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 356 of file type.h.

11.1.1.7 GetGridMapStatus

typedef FeedbackStatus agvc_interface::GetGridMapStatus

getGridMapFromAgv

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 296 of file type.h.

11.1.1.8 GetPngMapAllInfoStatus

typedef FeedbackStatus agvc_interface::GetPngMapAllInfoStatus

Png getPngMapAllInfo

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 284 of file type.h.

11.1.1.9 GetPngMapStatus

typedef FeedbackStatus agvc_interface::GetPngMapStatus

png getBase64PngMapFromAgv

NONE = 0, // png
FINISHED = 1, // png
FAILED = 2, //
RUNNING = 3, // png
PAUSED = 4, //
CANCELED = 5, //

Definition at line 320 of file type.h.

11.1.1.10 NavStatus

typedef FeedbackStatus agvc_interface::NavStatus

agv

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 212 of file type.h.

11.1.1.11 OccupancyGridMap

typedef MapInfo agvc_interface::OccupancyGridMap

Header header; // id map_id id
double resolution; // m
uint32_t width; //
uint32_t height; //
Pose2d origin; // {x(m) y(m) theta(rad)} // //
MapFormat format; // =MapFormat::OCCUPANCY_GRID
std::vector<int8_t> data; // 0 100 -1

Definition at line 691 of file type.h.

11.1.1.12 PreviewPngMapStatus

typedef FeedbackStatus agvc_interface::PreviewPngMapStatus

previewPngMapFromAgv

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 368 of file type.h.

11.1.1.13 RelocationStatus

typedef FeedbackStatus agvc_interface::RelocationStatus

relocation

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 260 of file type.h.

11.1.1.14 RpcClientPtr

using agvc_interface::RpcClientPtr = std::shared_ptr<RpcClient>

Definition at line 147 of file rpc.h.

11.1.1.15 RtdeClientPtr

using agvc_interface::RtdeClientPtr = std::shared_ptr<RtdeClient>

Definition at line 284 of file rtde.h.

11.1.1.16 SaveMapStatus

typedef FeedbackStatus agvc_interface::SaveMapStatus

saveMap

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 224 of file type.h.

11.1.1.17 ScriptClientPtr

using agvc_interface::ScriptClientPtr = std::shared_ptr<ScriptClient>

Definition at line 188 of file script.h.

11.1.1.18 SendGridMapStatus

typedef FeedbackStatus agvc_interface::SendGridMapStatus

sendGridMapToAgv

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 308 of file type.h.

11.1.1.19 SendPngMapStatus

typedef FeedbackStatus agvc_interface::SendPngMapStatus

png sendBase64PngMapToAgv

NONE = 0, // png
FINISHED = 1, // png
FAILED = 2, //
RUNNING = 3, // png
PAUSED = 4, //
CANCELED = 5, //

Definition at line 332 of file type.h.

11.1.1.20 SetGridMapAllInfoStatus

typedef FeedbackStatus agvc_interface::SetGridMapAllInfoStatus

setGridMapAllInfo

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 344 of file type.h.

11.1.1.21 SetPngMapAllInfoStatus

typedef FeedbackStatus agvc_interface::SetPngMapAllInfoStatus

Png setPngMapAllInfo

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Definition at line 272 of file type.h.

11.1.1.22 SwitchMapStatus

typedef FeedbackStatus agvc_interface::SwitchMapStatus

switchMap

NONE = 0, //
FINISHED = 1, //
FAILED = 2, //
RUNNING = 3, //
PAUSED = 4, //
CANCELED = 5, //

Enumerator

ENUM_AgvcErrorCodes_DECLARES	

Definition at line 71 of file type.h.

11.1.2.2 AutoChargingType

enum class agvc_interface::AutoChargingType [strong]

Since

0.7.0

Note

Enumerator

NONE	
LOW_POWER_TO_BOARD	
HIGH_POWER_LEAVE_BOARD	
FORCE_TO_BOARD	
FORCE_LEAVE_BOARD	

Definition at line 151 of file type.h.

11.1.2.3 CalibrationType

enum class agvc_interface::CalibrationType [strong]

Enumerator

NONE	
LASER_ODOM	-

Definition at line 196 of file type.h.

Enumerator

parse_error	
invalid_request	
method_not_found	
invalid_params	
internal_error	
server_error	
invalid	

Definition at line 850 of file type.h.

11.1.2.5 ExceptionCode

enum agvc_interface::ExceptionCode

Enumerator

EC_DISCONNECTED	
EC_NOT_LOGINED	
EC_INVAL_SOCKET	
EC_REQUEST_BUSY	
EC_SEND_FAILED	
EC_RECV_TIMEOUT	
EC_RECV_ERROR	
EC_PARSE_ERROR	
EC_INVALID_REQUEST	
EC_METHOD_NOT_FOUND	
EC_INVALID_PARAMS	
EC_INTERNAL_ERROR	
EC_SERVER_ERROR	
EC_INVALID	

Definition at line 861 of file type.h.

11.1.2.6 FeedbackStatus

enum class agvc_interface::FeedbackStatus [strong]

Enumerator

NONE	
FINISHED	
FAILED	
RUNNING	
PAUSED	
CANCELED	

Definition at line 163 of file type.h.

11.1.2.7 FirmwareUpdateMode

enum class agvc_interface::FirmwareUpdateMode [strong]

Enumerator

NONE	
UPDATE	
FORCED_UPDATE	
NOT_UPDATE	

Definition at line 176 of file type.h.

11.1.2.8 MapFormat

enum class agvc_interface::MapFormat [strong]

Enumerator

NONE	
OCCUPANCY_GRID	
BASE64_PNG	base64 png

Definition at line 105 of file type.h.

11.1.2.9 MapVirtualAreaShape

enum class agvc_interface::MapVirtualAreaShape [strong]

Enumerator

NONE	
LINE	
ARC	
CIRCLE	
POLYGON	

Definition at line 126 of file type.h.

11.1.2.10 MapVirtualAreaType

enum class agvc_interface::MapVirtualAreaType [strong]

Enumerator

NONE	
OBSTACLE	
SLOW_DOWN	
INFLATE	

Definition at line 115 of file type.h.

11.1.2.11 NavType

enum class agvc_interface::NavType [strong]

agv

Enumerator

NONE	
FREE_TO_POSE	
FREE_TO_STATION	
PATH_TO_STATION	

Definition at line 138 of file type.h.

11.1.2.12 PathShape

enum class agvc_interface::PathShape [strong]

Enumerator

NONE	
LINE	
ARC	
BEZIER	B_SPLINE = 4 // B

Definition at line 93 of file type.h.

11.1.2.13 RunningMode

enum class agvc_interface::RunningMode [strong]

agv

Enumerator

NONE	
START	
MAPPING	
NAVIGATE	
CHARGING	
MAINTAIN	
STOP	

Definition at line 79 of file type.h.

11.1.2.14 ScriptRuntimeState

enum class agvc_interface::ScriptRuntimeState [strong]

Enumerator

RUNNING	
PAUSED	
STOPPED	
ABORTING	

Definition at line 185 of file type.h.

11.1.3 Function Documentation

11.1.3.1 returnValue2Str()

const char * agvc_interface::returnValue2Str (
int retval) [inline]

Definition at line 909 of file type.h.

References ENUM_AgvcErrorCodes_DECLARES.

Chapter 12

Class Documentation

12.1 agvc_interface::AdjustParam Struct Reference

agv

#include <type.h>

Public Attributes

- bool enable
  agv true- false
- bool forward_flag
  agv true false
- double max_distance
  agv
- double max_angle
  agv

12.1.1 Detailed Description

agv

Definition at line 752 of file type.h.

12.1.2 Member Data Documentation

12.1.2.1 enable

bool agvc_interface::AdjustParam::enable

agv true- false

Definition at line 754 of file type.h.

12.1.2.2 forward_flag

bool agvc_interface::AdjustParam::forward_flag

agv true false

Definition at line 755 of file type.h.

12.1.2.3 max_angle

double agvc_interface::AdjustParam::max_angle

agv

Definition at line 757 of file type.h.

12.1.2.4 max_distance

double agvc_interface::AdjustParam::max_distance

agv

Definition at line 756 of file type.h.

The documentation for this struct was generated from the following file:

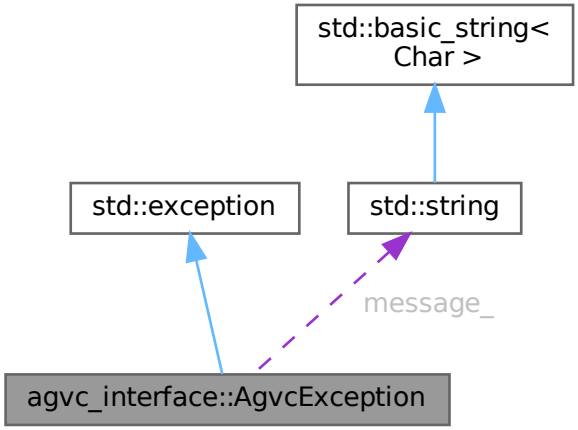
- include/agvc_interface/type.h

12.2 agvc_interface::AgvcException Class Reference

#include <type.h>

Inheritance diagram for agvc_interface::AgvcException:

Collaboration diagram for agvc_interface::AgvcException:



Public Member Functions

- AgvcException (int code, const std::string &prefix, const std::string &message) noexcept
- AgvcException (int code, const std::string &message) noexcept
- error_type type () const
- int code () const
- const char * what () const noexcept override

Private Attributes

- int code_
- std::string message_

12.2.1 Detailed Description

Definition at line 879 of file type.h.

12.2.2 Constructor & Destructor Documentation

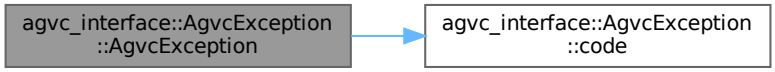
12.2.2.1 AgvcException() [1/2]

agvc_interface::AgvcException::AgvcException (
int code,
const std::string & prefix,
const std::string & message) [inline], [noexcept]

Definition at line 882 of file type.h.

References code().

Here is the call graph for this function:



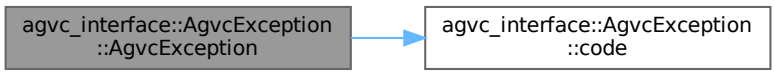
12.2.2.2 AgvcException() [2/2]

agvc_interface::AgvcException::AgvcException (
int code,
const std::string & message) [inline], [noexcept]

Definition at line 887 of file type.h.

References code().

Here is the call graph for this function:



12.2.3 Member Function Documentation

12.2.3.1 code()

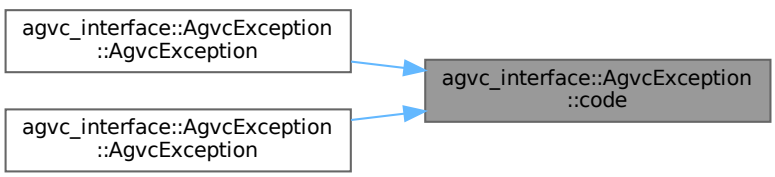
int agvc_interface::AgvcException::code () const [inline]

Definition at line 901 of file type.h.

References code_.

Referenced by AgvcException(), and AgvcException().

Here is the caller graph for this function:



12.2.3.2 type()

error_type agvc_interface::AgvcException::type () const [inline]

Definition at line 889 of file type.h.

References code_, agvc_interface::invalid, agvc_interface::parse_error, and agvc_interface::server_error.

12.2.3.3 what()

const char * agvc_interface::AgvcException::what () const [inline], [override], [noexcept]

Definition at line 902 of file type.h.

References message_.

12.2.4 Member Data Documentation

12.2.4.1 code_

int agvc_interface::AgvcException::code_ [private]

Definition at line 905 of file type.h.

Referenced by code(), and type().

12.2.4.2 message_

std::string agvc_interface::AgvcException::message_ [private]

Definition at line 906 of file type.h.

Referenced by what().

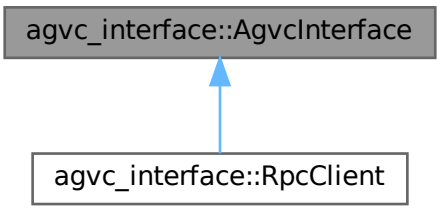
The documentation for this class was generated from the following file:

- include/agvc_interface/type.h

12.3 agvc_interface::AgvcInterface Class Reference

#include <agvc_interface.h>

Inheritance diagram for agvc_interface::AgvcInterface:



Public Member Functions

- AgvcInterface ()
- virtual ~AgvcInterface ()
- int setSystemClock (const std::string &stamp)

  Sets the AGV system time.
- AgvDetails getAgvDetails ()

  Queries current AGV information.
- RunningInfo getRunningInfo ()

  Queries current AGV running information.
- Pose2d getAgvCurrentPose ()

  Queries the current AGV pose.
- std::vector< Point2d > getLaserPointCloud ()

  Queries the current laser point cloud data.
- AsyncInterfaceResultStatus getAsyncInterfaceResultStatus ()

  Queries the running status of asynchronous APIs; this does not represent each API's own execution result.
- std::vector< StationMark > getAllStations ()

  Queries all station information on the current map.
- std::vector< StationMark > getAllStationsOfTargetMap (const Header &map_header)

  Queries all station information on the specified map.
- int addStation (const StationMark &station)

  Adds or modifies one station.
- int addStations (const std::vector< StationMark > &stations)

  Adds or modifies multiple stations.
- int addCPStationUseChargingBoard (const Header &station_header, const double &dis_station↩ _board=1.5)

  Adds a charging station by automatically recognizing the charging board pose.
- int deleteStation (const Header &station_header)

  Deletes the specified station and its related paths.
- int deleteStations (const std::vector< Header > &stations_header)

  Deletes multiple stations and their related paths.
- std::vector< PathStation > getAllPaths ()

  Queries all path information on the current map.
- std::vector< PathStation > getAllPathsOfTargetMap (const Header &map_header)

  Queries all path information on the specified map.
- PathStation getCurrentPath ()

  Queries the path currently being tracked by the AGV.
- int generatePath (const PathStation &path_station)

  Generates or modifies one path.
- int generatePaths (const std::vector< PathStation > &paths_station)

  Generates or modifies multiple paths.
- int deletePath (const Header &path_header)

  Deletes one path.
- int deletePaths (const std::vector< Header > &paths_header)

  Deletes multiple paths.
- std::vector< Header > getMapList ()

  Gets the headers of all maps on the AGV.
- Header getCurrentMapHeader ()

  Queries the header of the current AGV map.
- OccupancyGridMap getGridMapFromAgv (const Header &map_header)

Gets the specified occupancy grid map information.

- Base64PngMap getBase64PngMapFromAgv (const Header &map_header)

  Gets Base64-encoded PNG map information.
- Base64PngMap previewPngMapFromAgv (const Header &map_header, const int &image_width_px=0, const int &image_height_px=0)

  Gets a preview image with optional width and height.
- int sendGridMapToAgv (const OccupancyGridMap &map)

  Sends occupancy grid map information to the AGV.
- int sendBase64PngMapToAgv (const Base64PngMap &map)

  Sends a Base64-encoded PNG map to the AGV.
- int saveMap (const Header &map_header)

  Saves the map after mapping is complete.
- int switchMap (const Header &map_header)

  Switches to the specified map.
- int deleteMap (const Header &map_header)

  Deletes one specified map.
- int deleteMaps (const std::vector< Header > &map_headers)

  Deletes multiple specified maps.
- std::vector< MapVirtualArea > getAllMapVirtualArea ()

  Queries virtual areas on the current map.
- std::vector< MapVirtualArea > getAllMapVirtualAreaOfTargetMap (const Header &map_header)

  Queries virtual areas on the target map.
- int addMapVirtualArea (const MapVirtualArea &map_virtual_area)

  Adds or modifies one virtual area.
- int addMapVirtualAreas (const std::vector< MapVirtualArea > &map_virtual_areas)

  Adds or modifies multiple virtual areas.
- int deleteMapVirtualArea (const Header &virtual_area_header)

  Deletes the specified virtual area.
- int deleteMapVirtualAreas (const std::vector< Header > &virtual_areas_header)

  Deletes multiple specified virtual areas.
- MapAllInfo getGridMapAllInfo (const Header &map_header)

  Gets occupancy grid map data, paths, stations, and virtual areas for the specified map.
- MapAllInfo getPngMapAllInfo (const Header &map_header)

  Gets Base64 map data, paths, stations, and virtual areas for the specified map.
- int setGridMapAllInfo (const MapAllInfo &map_all_info, const Header &command_header={ "99999", "99999", 1, "99999" })

  Sets occupancy grid map, path, station, and virtual area information.
- int setPngMapAllInfo (const MapAllInfo &map_all_info, const Header &command_header={ "99999", "99999", 1, "99999" })

  Sets Base64 map, path, station, and virtual area information.
- int deleteMapAllInfo (const Header &map_header)

  Deletes all information of the specified map, including map data, stations, paths, and virtual areas.
- std::vector< std::string > getWifiList ()

  Gets the currently scanned WiFi list.
- int connectWifi (const std::string &ssid, const std::string &password)

  Connects to the specified WiFi and automatically switches to WiFi mode.
- int enableHotspot (const std::string &password)

  Enables hotspot mode and automatically switches to hotspot mode.
- std::vector< IpAddressInfo > getIpAddressList ()

  Gets all local IP addresses.

- int changeRunningMode (const RunningMode &running_mode, const Header &command_↩ header={ ”99999”, ”99999”, 1, ”99999” })

  Switches the AGV running mode.
- int setControlSpeed (const Speed &speed)

  Controls AGV motion.
- int setNavGoal (const NavGoalType &target)

  Sends a navigation goal to the AGV.
- NavInfo getNavInfo ()

  Gets current navigation information.
- int pauseAgvSpeed ()

  Pauses AGV speed.
- int resumeAgvSpeed ()

  Resumes AGV speed after a pause.
- int cancelNavigation ()

  Cancels the navigation task.
- int relocation (const Pose2d &init_pose, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Relocates the AGV.
- std::string getAgvControllerParametersFile ()

  Gets the AGV controller parameter file.
- int setAgvControllerParametersFile (const std::string &agv_parameters)

  Sets the AGV controller parameter file.
- int refreshAgvControllerParametersFile ()

  Refreshes parameters on AGV nodes after the AGV controller parameter file is modified.
- int resetAgvControllerParametersFile ()

  Restores the AGV controller parameter file to factory defaults.
- int checkPowerAndAutoCharge (const Header &check_power_header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &nav_board_charge_station_id=”99999”)

  Checks the current battery level and automatically docks for charging when the battery is low.
- int setLeaveBoardTargetStation (const Header &leave_board_target_station_header)

  Sets the target station for automatic undocking.
- Header getLeaveBoardTargetStation ()

  Gets the target station for automatic undocking.
- int forcedAgvToChargingBoard (const Header &forced_charge_header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &nav_board_charge_station_id=”99999”)

  Forces the AGV to dock for charging.
- int forcedAgvLeaveChargingBoard (const Header &forced_leave_header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &leave_board_target_station_id=”99999”)

  Forces the AGV to leave the charging board.
- int sendAutoChargingCommand (const AutoChargingCommand &auto_charging_command)

  Sends an automatic charging command.
- std::string getAgvLogMessage ()

  Gets AGV runtime log messages.
- int updateSoftware (const std::string &upgrade_pack_path)

  Upgrades AGV software.
- std::vector< std::string > getSoftwareVersionList ()

  Gets the AGV software version list.
- int switchSoftwareVersion (const std::string &software_version)

  Switches the AGV software version.
- int uninstallSoftware (const std::string &software_version)

  Uninstalls an AGV software version.

- int updateFirmware (const FirmwareUpdateParam &update_firmware)

  Updates AGV firmware.
- FirmwareUpdateProcessInfo getUpdateFirmwareProcess ()

  Gets firmware update process information.
- int startCollectCalibrationData (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Starts collecting calibration data.
- int cancelCollectCalibrationData (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Cancels calibration data collection.
- int startCalibration (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Starts calibration.
- CalibrationProcessInfo getCalibrationProcessInfo (const CalibrationType &type)

  Gets calibration information.
- int restartAgv ()

  Restarts the AGV.
- int setAgvName (const std::string &agv_name)

  Sets the AGV name.
- int setPriority (const std::string &name, const std::string &ip=””)

  Sets control priority.
- int releasePriority ()

  Releases control priority.
- int scriptPaused ()

  Pauses the script.
- int scriptResume ()

  Resumes the script.
- int scriptStop ()

  Stops the script.
- ScriptRuntimeState getScriptStatus ()

  Gets the script runtime state.
- int setScriptStatus (ScriptRuntimeState status)

  Sets the script runtime state.
- std::string errorCodeDecoder (const int &error_code)

  Queries an error code.
- std::vector< Header > getCurrentErrorCodes ()

  Gets current error codes.
- int soundLightPrompt ()

  Locates the AGV by automatically playing audio and flashing special lights.
- bool isObstacleAhead (const double &detect_distance=1.0)

  Checks whether there is an obstacle in the AGV travel direction.
- StationMark getNearestStation (const double &detect_distance=1.0)

  Gets the nearest station within the specified detection distance.
- int autoAlignRailway (const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Automatically aligns with the rail and boards the rail.

12.3.1 Detailed Description

Definition at line 98 of file agvc_interface.h.

12.3.2 Constructor & Destructor Documentation

12.3.2.1 AgvcInterface()

agvc_interface::AgvcInterface::AgvcInterface ()

12.3.2.2 ~AgvcInterface()

virtual agvc_interface::AgvcInterface::~AgvcInterface () [virtual]

The documentation for this class was generated from the following file:

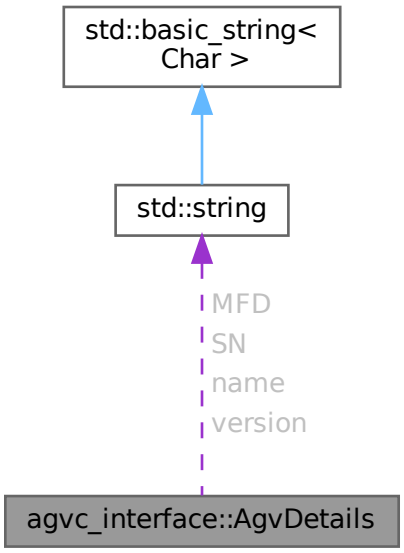
- include/agvc_interface/agvc_interface.h

12.4 agvc_interface::AgvDetails Struct Reference

agv

#include <type.h>

Collaboration diagram for agvc_interface::AgvDetails:



Public Attributes

- std::string name
  agv
- std::string version
- std::string MFD
- std::string SN
  S/N
- int communication_version
  bridge
- int battery_version
- int loop_times
- float surplus_capacity
- int left_motor_firmware_version
- int right_motor_firmware_version
- int master_board_firmware_version
- int hardware_abstraction_version
- int error_code_version
- float max_speed
  agv
- float max_angular
  agv

12.4.1 Detailed Description

agv

Definition at line 438 of file type.h.

12.4.2 Member Data Documentation

12.4.2.1 battery_version

int agvc_interface::AgvDetails::battery_version

Definition at line 445 of file type.h.

12.4.2.2 communication_version

int agvc_interface::AgvDetails::communication_version

bridge

Definition at line 444 of file type.h.

12.4.2.3 error_code_version

int agvc_interface::AgvDetails::error_code_version

Definition at line 452 of file type.h.

12.4.2.4 hardware_abstraction_version

int agvc_interface::AgvDetails::hardware_abstraction_version

Definition at line 451 of file type.h.

12.4.2.5 left_motor_firmware_version

int agvc_interface::AgvDetails::left_motor_firmware_version

Definition at line 448 of file type.h.

12.4.2.6 loop_times

int agvc_interface::AgvDetails::loop_times

Definition at line 446 of file type.h.

12.4.2.7 master_board_firmware_version

int agvc_interface::AgvDetails::master_board_firmware_version

Definition at line 450 of file type.h.

12.4.2.8 max_angular

float agvc_interface::AgvDetails::max_angular

agv

Definition at line 454 of file type.h.

12.4.2.9 max_speed

float agvc_interface::AgvDetails::max_speed

agv

Definition at line 453 of file type.h.

12.4.2.10 MFD

std::string agvc_interface::AgvDetails::MFD

Definition at line 442 of file type.h.

12.4.2.11 name

std::string agvc_interface::AgvDetails::name

agv

Definition at line 440 of file type.h.

12.4.2.12 right_motor_firmware_version

int agvc_interface::AgvDetails::right_motor_firmware_version

Definition at line 449 of file type.h.

12.4.2.13 SN

std::string agvc_interface::AgvDetails::SN

S/N

Definition at line 443 of file type.h.

12.4.2.14 surplus_capacity

float agvc_interface::AgvDetails::surplus_capacity

Definition at line 447 of file type.h.

12.4.2.15 version

std::string agvc_interface::AgvDetails::version

Definition at line 441 of file type.h.

The documentation for this struct was generated from the following file:

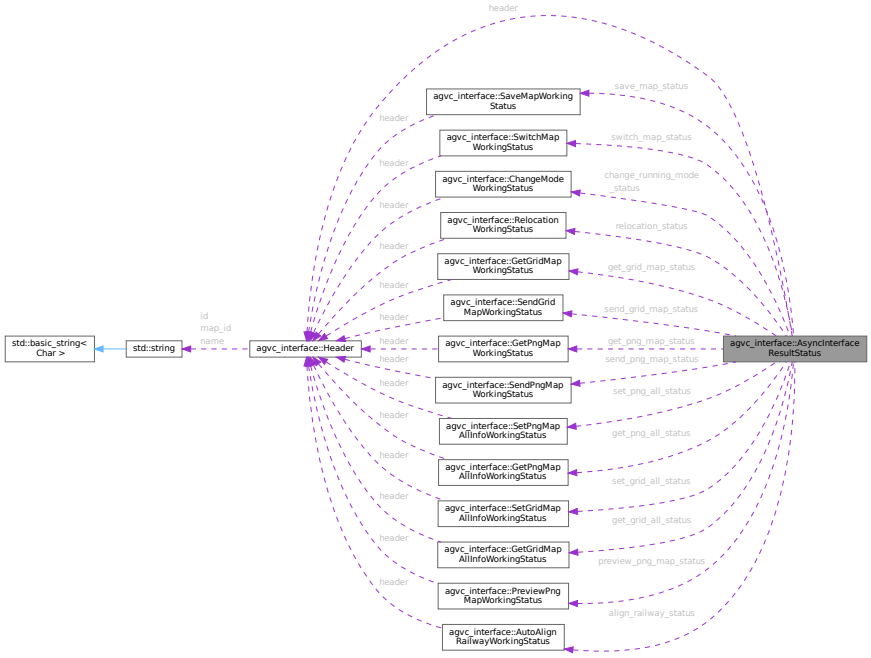
- include/agvc_interface/type.h

12.5 agvc_interface::AsyncInterfaceResultStatus Struct Reference

14 saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, previewPngMapFromAgv, autoAlignRailway

#include <type.h>

Collaboration diagram for agvc_interface::AsyncInterfaceResultStatus:



Public Attributes

- Header header
  agv SN agv id
- SaveMapWorkingStatus save_map_status
  (saveMap)
- SwitchMapWorkingStatus switch_map_status
  (switchMap)
- ChangeModeWorkingStatus change_running_mode_status
  (changeRunningMode)
- RelocationWorkingStatus relocation_status
  (relocation)
- GetGridMapWorkingStatus get_grid_map_status
  (getGridMapFromAgv)
- SendGridMapWorkingStatus send_grid_map_status
  (sendGridMapToAgv)
- GetPngMapWorkingStatus get_png_map_status
  png (getBase64PngMapFromAgv)
- SendPngMapWorkingStatus send_png_map_status
  png (sendBase64PngMapToAgv)
- SetPngMapAllInfoWorkingStatus set_png_all_status
  png (setPngMapAllInfo)
- GetPngMapAllInfoWorkingStatus get_png_all_status
  png (getPngMapAllInfo)
- SetGridMapAllInfoWorkingStatus set_grid_all_status
  (setGridMapAllInfo)
- GetGridMapAllInfoWorkingStatus get_grid_all_status
  (getGridMapAllInfo)
- PreviewPngMapWorkingStatus preview_png_map_status
  png (previewPngMapFromAgv)
- AutoAlignRailwayWorkingStatus align_railway_status
  (autoAlignRailway)

12.5.1 Detailed Description

14 saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, previewPngMapFromAgv, autoAlignRailway

Definition at line 619 of file type.h.

12.5.2 Member Data Documentation

12.5.2.1 align_railway_status

AutoAlignRailwayWorkingStatus agvc_interface::AsyncInterfaceResultStatus::align_railway_status

(autoAlignRailway)

Definition at line 635 of file type.h.

12.5.2.2 change_running_mode_status

ChangeModeWorkingStatus agvc_interface::AsyncInterfaceResultStatus::change_running_mode_status

(changeRunningMode)

Definition at line 624 of file type.h.

12.5.2.3 get_grid_all_status

GetGridMapAllInfoWorkingStatus agvc_interface::AsyncInterfaceResultStatus::get_grid_all_status

(getGridMapAllInfo)

Definition at line 633 of file type.h.

12.5.2.4 get_grid_map_status

GetGridMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::get_grid_map_status

(getGridMapFromAgv)

Definition at line 626 of file type.h.

12.5.2.5 get_png_all_status

GetPngMapAllInfoWorkingStatus agvc_interface::AsyncInterfaceResultStatus::get_png_all_status

png (getPngMapAllInfo)

Definition at line 631 of file type.h.

12.5.2.6 get_png_map_status

GetPngMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::get_png_map_status

png (getBase64PngMapFromAgv)

Definition at line 628 of file type.h.

12.5.2.7 header

Header agvc_interface::AsyncInterfaceResultStatus::header

agv SN agv id

Definition at line 621 of file type.h.

12.5.2.8 preview_png_map_status

PreviewPngMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::preview_png_map_status

png (previewPngMapFromAgv)

Definition at line 634 of file type.h.

12.5.2.9 relocation_status

RelocationWorkingStatus agvc_interface::AsyncInterfaceResultStatus::relocation_status

(relocation)

Definition at line 625 of file type.h.

12.5.2.10 save_map_status

SaveMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::save_map_status

(saveMap)

Definition at line 622 of file type.h.

12.5.2.11 send_grid_map_status

SendGridMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::send_grid_map_status

(sendGridMapToAgv)

Definition at line 627 of file type.h.

12.5.2.12 send_png_map_status

SendPngMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::send_png_map_status

png (sendBase64PngMapToAgv)

Definition at line 629 of file type.h.

12.5.2.13 set_grid_all_status

SetGridMapAllInfoWorkingStatus agvc_interface::AsyncInterfaceResultStatus::set_grid_all_status

(setGridMapAllInfo)

Definition at line 632 of file type.h.

12.5.2.14 set_png_all_status

SetPngMapAllInfoWorkingStatus agvc_interface::AsyncInterfaceResultStatus::set_png_all_status

png (setPngMapAllInfo)

Definition at line 630 of file type.h.

12.5.2.15 switch_map_status

SwitchMapWorkingStatus agvc_interface::AsyncInterfaceResultStatus::switch_map_status

(switchMap)

Definition at line 623 of file type.h.

The documentation for this struct was generated from the following file:

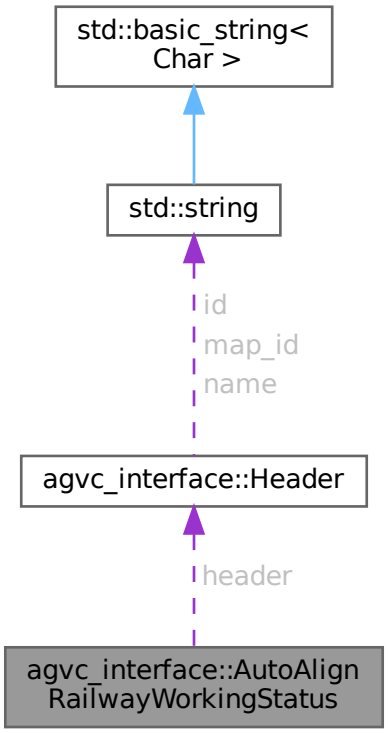
- include/agvc_interface/type.h

12.6 agvc_interface::AutoAlignRailwayWorkingStatus Struct Reference

autoAlignRailway

#include <type.h>

Collaboration diagram for agvc_interface::AutoAlignRailwayWorkingStatus:



Public Attributes

- Header header
  header.id autoAlignRailway id
- AutoAlignRailwayStatus status
  autoAlignRailway NONE - RUNNING- FINISH-

12.6.1 Detailed Description

autoAlignRailway

Definition at line 606 of file type.h.

12.6.2 Member Data Documentation

12.6.2.1 header

Header agvc_interface::AutoAlignRailwayWorkingStatus::header

header.id autoAlignRailway id

Definition at line 608 of file type.h.

12.6.2.2 status

AutoAlignRailwayStatus agvc_interface::AutoAlignRailwayWorkingStatus::status

autoAlignRailway NONE - RUNNING- FINISH-

Definition at line 609 of file type.h.

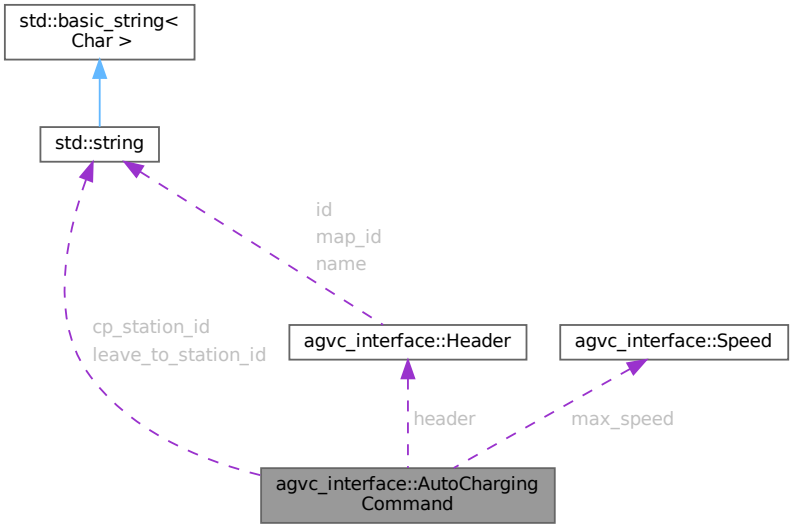
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.7 agvc_interface::AutoChargingCommand Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::AutoChargingCommand:



Public Attributes

- Header header
  id id
- AutoChargingType auto_charging_type

- std::string cp_station_id
  id
- std::string leave_to_station_id
  id
- Speed max_speed

12.7.1 Detailed Description

Since

0.7.0

Definition at line 800 of file type.h.

12.7.2 Member Data Documentation

12.7.2.1 auto_charging_type

AutoChargingType agvc_interface::AutoChargingCommand::auto_charging_type

Definition at line 803 of file type.h.

12.7.2.2 cp_station_id

std::string agvc_interface::AutoChargingCommand::cp_station_id

id

Definition at line 804 of file type.h.

12.7.2.3 header

Header agvc_interface::AutoChargingCommand::header

id id

Definition at line 802 of file type.h.

12.7.2.4 leave_to_station_id

std::string agvc_interface::AutoChargingCommand::leave_to_station_id

id

Definition at line 805 of file type.h.

12.7.2.5 max_speed

Speed agvc_interface::AutoChargingCommand::max_speed

Definition at line 806 of file type.h.

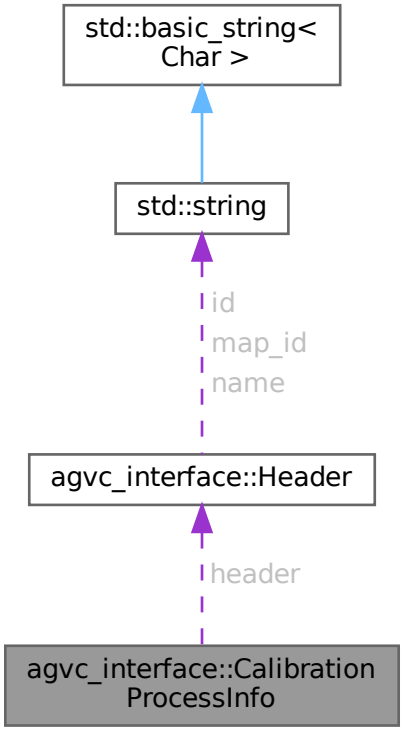
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.8 agvc_interface::CalibrationProcessInfo Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::CalibrationProcessInfo:



Public Attributes

- Header header
  id id
- CalibrationType type
- CalibrationStatus status
- uint32_t collected_data_count

12.8.1 Detailed Description

Definition at line 830 of file type.h.

12.8.2 Member Data Documentation

12.8.2.1 collected_data_count

uint32_t agvc_interface::CalibrationProcessInfo::collected_data_count

Definition at line 835 of file type.h.

12.8.2.2 header

Header agvc_interface::CalibrationProcessInfo::header

id id

Definition at line 832 of file type.h.

12.8.2.3 status

CalibrationStatus agvc_interface::CalibrationProcessInfo::status

Definition at line 834 of file type.h.

12.8.2.4 type

CalibrationType agvc_interface::CalibrationProcessInfo::type

Definition at line 833 of file type.h.

The documentation for this struct was generated from the following file:

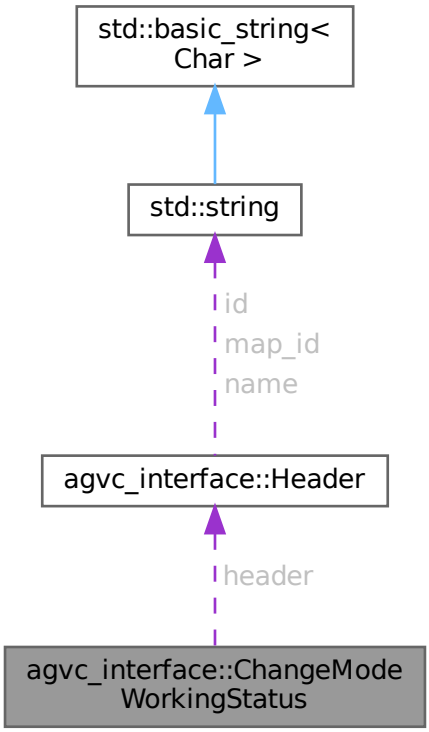
- include/agvc_interface/type.h

12.9 agvc_interface::ChangeModeWorkingStatus Struct Reference

changeRunningMode

#include <type.h>

Collaboration diagram for agvc_interface::ChangeModeWorkingStatus:



Public Attributes

- Header header
  header.id changeRunningMode id
- ChangeRunningModeStatus status
  changeRunningMode NONE - RUNNING- FINISH-

12.9.1 Detailed Description

changeRunningMode

Definition at line 506 of file type.h.

12.9.2 Member Data Documentation

12.9.2.1 header

Header agvc_interface::ChangeModeWorkingStatus::header

header.id changeRunningMode id

Definition at line 508 of file type.h.

12.9.2.2 status

ChangeRunningModeStatus agvc_interface::ChangeModeWorkingStatus::status

changeRunningMode NONE - RUNNING- FINISH-

Definition at line 509 of file type.h.

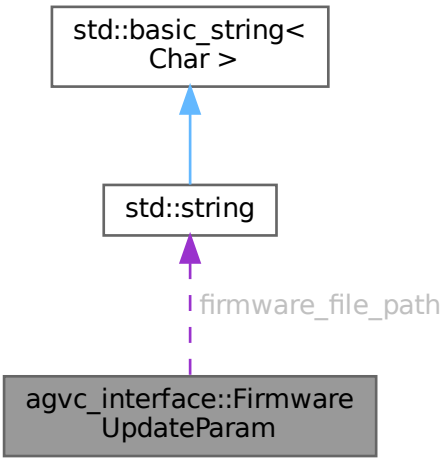
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.10 agvc_interface::FirmwareUpdateParam Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::FirmwareUpdateParam:



Public Attributes

- FirmwareUpdateMode left_motor
- FirmwareUpdateMode right_motor
- FirmwareUpdateMode master_board
- std::string firmware_file_path
  "/root/update.firm"

12.10.1 Detailed Description

Definition at line 812 of file type.h.

12.10.2 Member Data Documentation

12.10.2.1 firmware_file_path

std::string agvc_interface::FirmwareUpdateParam::firmware_file_path

"/root/update.firm"

Definition at line 817 of file type.h.

12.10.2.2 left_motor

FirmwareUpdateMode agvc_interface::FirmwareUpdateParam::left_motor

Definition at line 814 of file type.h.

12.10.2.3 master_board

FirmwareUpdateMode agvc_interface::FirmwareUpdateParam::master_board

Definition at line 816 of file type.h.

12.10.2.4 right_motor

FirmwareUpdateMode agvc_interface::FirmwareUpdateParam::right_motor

Definition at line 815 of file type.h.

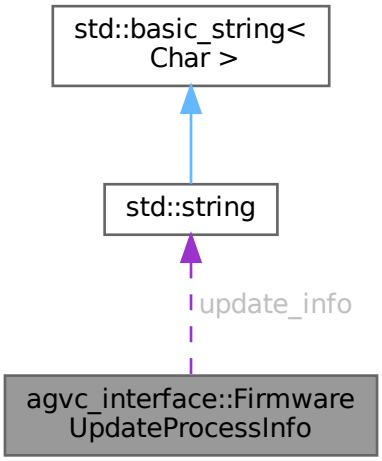
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.11 agvc_interface::FirmwareUpdateProcessInfo Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::FirmwareUpdateProcessInfo:



Public Attributes

- std::string update_info
  failed , none
- double update_progress
  1.0

12.11.1 Detailed Description

Definition at line 823 of file type.h.

12.11.2 Member Data Documentation

12.11.2.1 update_info

std::string agvc_interface::FirmwareUpdateProcessInfo::update_info

failed , none

Definition at line 825 of file type.h.

12.11.2.2 update_progress

double agvc_interface::FirmwareUpdateProcessInfo::update_progress

1.0

Definition at line 827 of file type.h.

The documentation for this struct was generated from the following file:

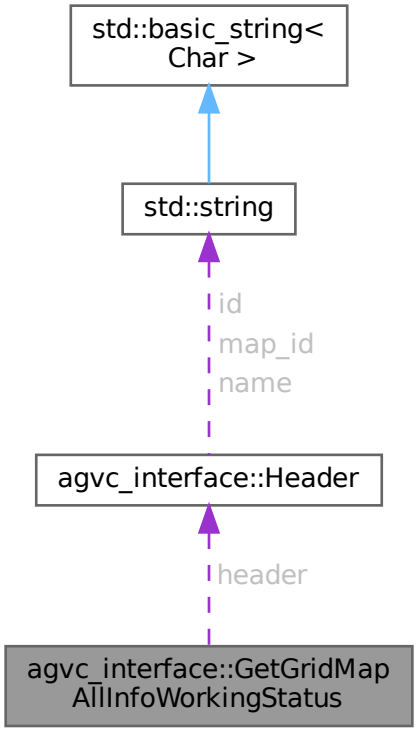
- include/agvc_interface/type.h

12.12 agvc_interface::GetGridMapAllInfoWorkingStatus Struct Reference

getGridMapAllInfo

#include <type.h>

Collaboration diagram for agvc_interface::GetGridMapAllInfoWorkingStatus:



Public Attributes

- Header header
  header.id getGridMapAllInfo id
- GetGridMapAllInfoStatus status
  getGridMapAllInfo NONE - RUNNING- FINISH-

12.12.1 Detailed Description

getGridMapAllInfo

Definition at line 586 of file type.h.

12.12.2 Member Data Documentation

12.12.2.1 header

Header agvc_interface::GetGridMapAllInfoWorkingStatus::header

header.id getGridMapAllInfo id

Definition at line 588 of file type.h.

12.12.2.2 status

GetGridMapAllInfoStatus agvc_interface::GetGridMapAllInfoWorkingStatus::status

getGridMapAllInfo NONE - RUNNING- FINISH-

Definition at line 589 of file type.h.

The documentation for this struct was generated from the following file:

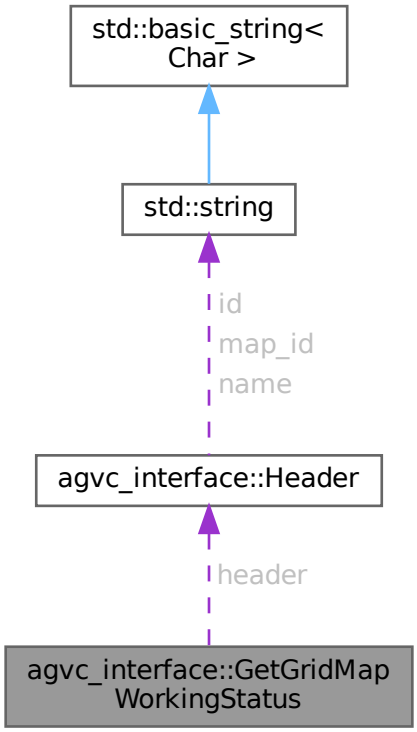
- include/agvc_interface/type.h

12.13 agvc_interface::GetGridMapWorkingStatus Struct Reference

getGridMapFromAgv

#include <type.h>

Collaboration diagram for agvc_interface::GetGridMapWorkingStatus:



Public Attributes

- Header header
  header.id getGridMapFromAgv id
- GetGridMapStatus status
  getGridMapFromAgv NONE - RUNNING- FINISH-

12.13.1 Detailed Description

getGridMapFromAgv

Definition at line 543 of file type.h.

12.13.2 Member Data Documentation

12.13.2.1 header

Header agvc_interface::GetGridMapWorkingStatus::header

header.id getGridMapFromAgv id

Definition at line 545 of file type.h.

12.13.2.2 status

GetGridMapStatus agvc_interface::GetGridMapWorkingStatus::status

getGridMapFromAgv NONE - RUNNING- FINISH-

Definition at line 546 of file type.h.

The documentation for this struct was generated from the following file:

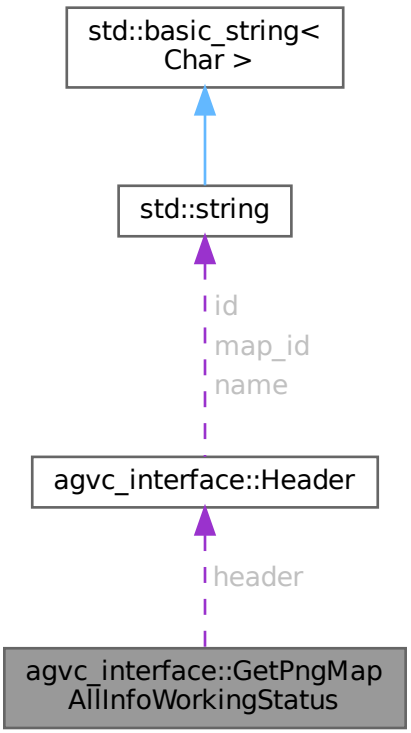
- include/agvc_interface/type.h

12.14 agvc_interface::GetPngMapAllInfoWorkingStatus Struct Reference

getPngMapAllInfo

#include <type.h>

Collaboration diagram for agvc_interface::GetPngMapAllInfoWorkingStatus:



Public Attributes

- Header header
  header.id getPngMapAllInfo id
- GetPngMapAllInfoStatus status
  getPngMapAllInfo NONE - RUNNING- FINISH-

12.14.1 Detailed Description

getPngMapAllInfo

Definition at line 534 of file type.h.

12.14.2 Member Data Documentation

12.14.2.1 header

Header agvc_interface::GetPngMapAllInfoWorkingStatus::header

header.id getPngMapAllInfo id

Definition at line 536 of file type.h.

12.14.2.2 status

GetPngMapAllInfoStatus agvc_interface::GetPngMapAllInfoWorkingStatus::status

getPngMapAllInfo NONE - RUNNING- FINISH-

Definition at line 537 of file type.h.

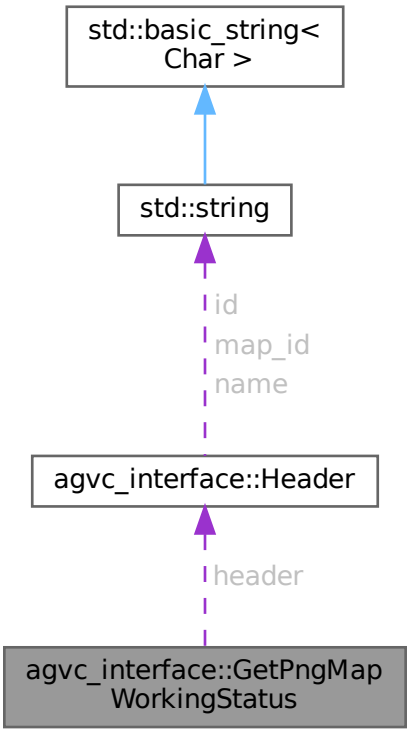
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.15 agvc_interface::GetPngMapWorkingStatus Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::GetPngMapWorkingStatus:



Public Attributes

- Header header
  header.id getBase64PngMapFromAgv id
- GetPngMapStatus status
  getBase64PngMapFromAgv NONE - RUNNING- FINISH-

12.15.1 Detailed Description

Definition at line 558 of file type.h.

12.15.2 Member Data Documentation

12.15.2.1 header

Header agvc_interface::GetPngMapWorkingStatus::header

header.id getBase64PngMapFromAgv id

Definition at line 560 of file type.h.

12.15.2.2 status

GetPngMapStatus agvc_interface::GetPngMapWorkingStatus::status

getBase64PngMapFromAgv NONE - RUNNING- FINISH-

Definition at line 561 of file type.h.

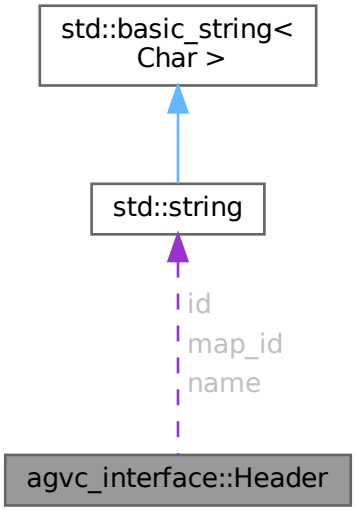
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.16 agvc_interface::Header Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::Header:



Public Attributes

- std::string id
  uuid
- std::string name
- int64_t stamp
  UTC (1970 01 01 00 00 00 ~)
- std::string map_id
  id, uuid a5e7c6de-4463-443e-8b25-ccbfd6a27aee,
- int error_code = -1

12.16.1 Detailed Description

Definition at line 425 of file type.h.

12.16.2 Member Data Documentation

12.16.2.1 error_code

int agvc_interface::Header::error_code = -1

Definition at line 432 of file type.h.

12.16.2.2 id

std::string agvc_interface::Header::id

uuid

Definition at line 427 of file type.h.

12.16.2.3 map_id

std::string agvc_interface::Header::map_id

id, uuid a5e7c6de-4463-443e-8b25-ccbfd6a27aee,

Definition at line 430 of file type.h.

12.16.2.4 name

std::string agvc_interface::Header::name

Definition at line 428 of file type.h.

12.16.2.5 stamp

int64_t agvc_interface::Header::stamp

UTC (1970 01 01 00 00 00 ~)

Definition at line 429 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.17 agvc_interface::InputParser Class Reference

#include <rtde.h>

Public Member Functions

- InputParser ()
- ~InputParser ()
- bool popBool ()
- int popInt32 ()
- int64_t popInt64 ()
- int16_t popInt16 ()
- double popDouble ()
- char popChar ()
- std::string popString ()
- std::vector< int > popVectorInt ()
- std::vector< int16_t > popVectorInt16 ()
- std::vector< double > popVectorDouble ()
- std::vector< std::vector< double > > popVectorVectorDouble ()
- agvc_interface::Pose2d popPose2d ()
- agvc_interface::RunningInfo popRunningInfo ()
- agvc_interface::AgvDetails popAgvDetails ()
- agvc_interface::NavInfo popNavInfo ()
- std::vector< agvc_interface::Point2d > popVectorPoint2d ()
- std::vector< agvc_interface::Header > popVectorHeader ()

Private Attributes

- friend RtdeClient
- Impl * impl

12.17.1 Detailed Description

Definition at line 48 of file rtde.h.

12.17.2 Constructor & Destructor Documentation

12.17.2.1 InputParser()

agvc_interface::InputParser::InputParser ()

12.17.2.2 ~InputParser()

agvc_interface::InputParser::~InputParser ()

12.17.3 Member Function Documentation

12.17.3.1 popAgvDetails()

agvc_interface::AgvDetails agvc_interface::InputParser::popAgvDetails ()

12.17.3.2 popBool()

bool agvc_interface::InputParser::popBool ()

12.17.3.3 popChar()

char agvc_interface::InputParser::popChar ()

12.17.3.4 popDouble()

double agvc_interface::InputParser::popDouble ()

12.17.3.5 popInt16()

int16_t agvc_interface::InputParser::popInt16 ()

12.17.3.6 popInt32()

int agvc_interface::InputParser::popInt32 ()

12.17.3.7 popInt64()

int64_t agvc_interface::InputParser::popInt64 ()

12.17.3.8 popNavInfo()

agvc_interface::NavInfo agvc_interface::InputParser::popNavInfo ()

12.17.3.9 popPose2d()

agvc_interface::Pose2d agvc_interface::InputParser::popPose2d ()

12.17.3.10 popRunningInfo()

agvc_interface::RunningInfo agvc_interface::InputParser::popRunningInfo ()

12.17.3.11 popString()

std::string agvc_interface::InputParser::popString ()

12.17.3.12 popVectorDouble()

std::vector< double > agvc_interface::InputParser::popVectorDouble ()

12.17.3.13 popVectorHeader()

std::vector< agvc_interface::Header > agvc_interface::InputParser::popVectorHeader ()

12.17.3.14 popVectorInt()

std::vector< int > agvc_interface::InputParser::popVectorInt ()

12.17.3.15 popVectorInt16()

std::vector< int16_t > agvc_interface::InputParser::popVectorInt16 ()

12.17.3.16 popVectorPoint2d()

std::vector< agvc_interface::Point2d > agvc_interface::InputParser::popVectorPoint2d ()

12.17.3.17 popVectorVectorDouble()

std::vector< std::vector< double > > agvc_interface::InputParser::popVectorVectorDouble ()

12.17.4 Member Data Documentation

12.17.4.1 impl

Impl* agvc_interface::InputParser::impl [private]

Definition at line 75 of file rtde.h.

12.17.4.2 RtdeClient

friend agvc_interface::InputParser::RtdeClient [private]

Definition at line 73 of file rtde.h.

The documentation for this class was generated from the following file:

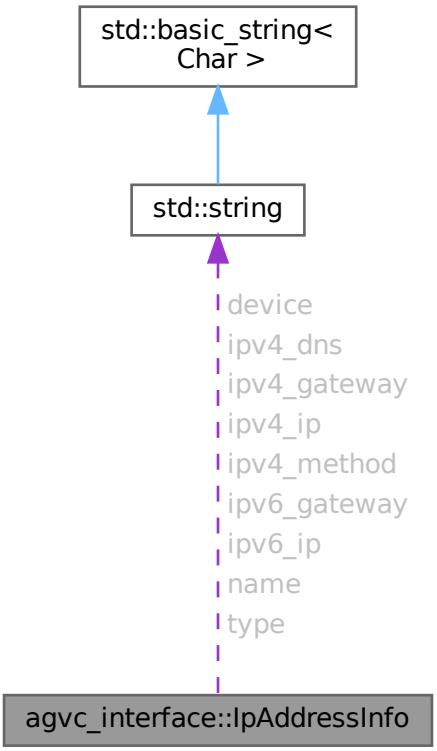
- include/agvc_interface/rtde.h

12.18 agvc_interface::IpAddressInfo Struct Reference

IP

#include <type.h>

Collaboration diagram for agvc_interface::IpAddressInfo:



Public Attributes

- std::string name
- std::string type
- std::string device
- std::string ipv4_ip
  IPv4 CIDR
- std::string ipv4_gateway
- std::string ipv4_method
  auto, manual
- std::string ipv4_dns

DNS
- std::string ipv6_ip
  IPv6 CIDR
- std::string ipv6_gateway

12.18.1 Detailed Description

IP

Definition at line 737 of file type.h.

12.18.2 Member Data Documentation

12.18.2.1 device

std::string agvc_interface::IpAddressInfo::device

Definition at line 741 of file type.h.

12.18.2.2 ipv4_dns

std::string agvc_interface::IpAddressInfo::ipv4_dns

DNS

Definition at line 745 of file type.h.

12.18.2.3 ipv4_gateway

std::string agvc_interface::IpAddressInfo::ipv4_gateway

Definition at line 743 of file type.h.

12.18.2.4 ipv4_ip

std::string agvc_interface::IpAddressInfo::ipv4_ip

IPv4 CIDR

Definition at line 742 of file type.h.

12.18.2.5 ipv4_method

std::string agvc_interface::IpAddressInfo::ipv4_method

auto, manual

Definition at line 744 of file type.h.

12.18.2.6 ipv6_gateway

std::string agvc_interface::IpAddressInfo::ipv6_gateway

Definition at line 747 of file type.h.

12.18.2.7 ipv6_ip

std::string agvc_interface::IpAddressInfo::ipv6_ip

IPv6 CIDR

Definition at line 746 of file type.h.

12.18.2.8 name

std::string agvc_interface::IpAddressInfo::name

Definition at line 739 of file type.h.

12.18.2.9 type

std::string agvc_interface::IpAddressInfo::type

Definition at line 740 of file type.h.

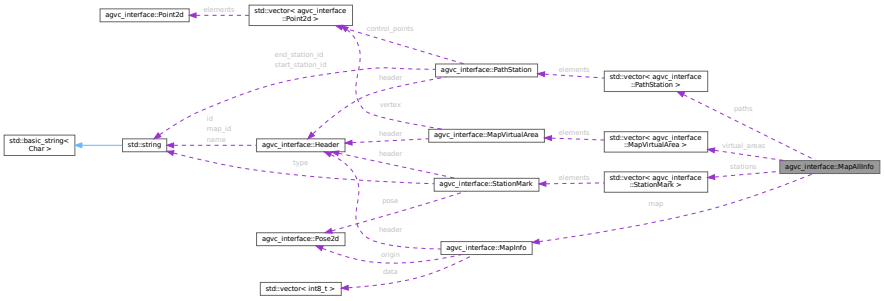
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.19 agvc_interface::MapAllInfo Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::MapAllInfo:



Public Attributes

- MapInfo map
- std::vector< StationMark > stations
- std::vector< PathStation > paths
- std::vector< MapVirtualArea > virtual_areas

12.19.1 Detailed Description

Definition at line 726 of file type.h.

12.19.2 Member Data Documentation

12.19.2.1 map

MapInfo agvc_interface::MapAllInfo::map

Definition at line 728 of file type.h.

12.19.2.2 paths

std::vector<PathStation> agvc_interface::MapAllInfo::paths

Definition at line 730 of file type.h.

12.19.2.3 stations

std::vector<StationMark> agvc_interface::MapAllInfo::stations

Definition at line 729 of file type.h.

12.19.2.4 virtual_areas

std::vector<MapVirtualArea> agvc_interface::MapAllInfo::virtual_areas

Definition at line 731 of file type.h.

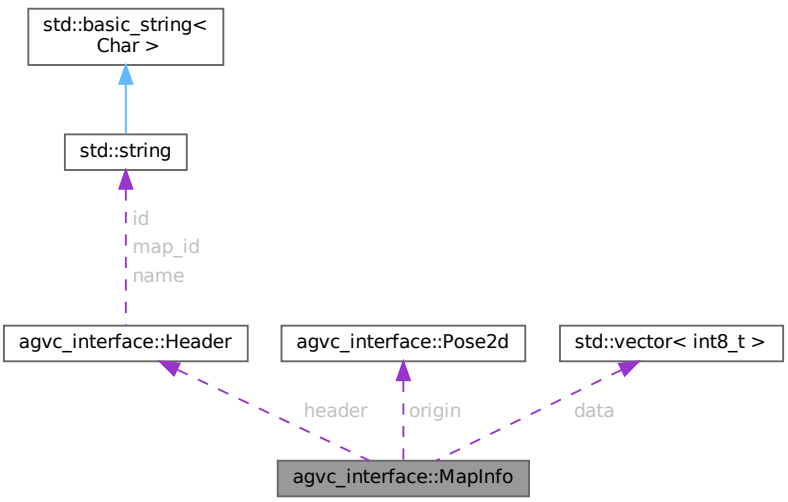
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.20 agvc_interface::MapInfo Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::MapInfo:



Public Attributes

- Header header
  id map_id id
- double resolution
  / m
- uint32_t width
  /
- uint32_t height
  /
- Pose2d origin
  {x(m) y(m) theta(rad)}
- MapFormat format
- std::vector< int8_t > data

12.20.1 Detailed Description

Definition at line 665 of file type.h.

12.20.2 Member Data Documentation

12.20.2.1 data

std::vector<int8_t> agvc_interface::MapInfo::data

Definition at line 675 of file type.h.

12.20.2.2 format

MapFormat agvc_interface::MapInfo::format

Definition at line 674 of file type.h.

12.20.2.3 header

Header agvc_interface::MapInfo::header

id map_id id

Definition at line 667 of file type.h.

12.20.2.4 height

uint32_t agvc_interface::MapInfo::height

/

Definition at line 670 of file type.h.

12.20.2.5 origin

Pose2d agvc_interface::MapInfo::origin

{x(m) y(m) theta(rad)}

Definition at line 671 of file type.h.

12.20.2.6 resolution

double agvc_interface::MapInfo::resolution

/ m

Definition at line 668 of file type.h.

12.20.2.7 width

uint32_t agvc_interface::MapInfo::width

/

Definition at line 669 of file type.h.

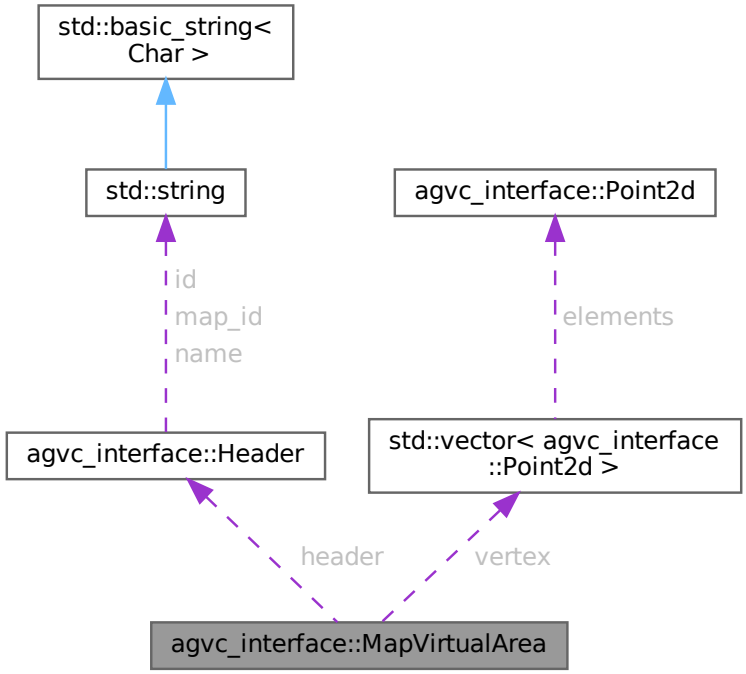
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.21 agvc_interface::MapVirtualArea Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::MapVirtualArea:



Public Attributes

- Header header
  id id
- MapVirtualAreaType type
- MapVirtualAreaShape shape
- double width_speed
  / /
- std::vector< Point2d > vertex
  {{x(m) y(m)},{}, ...} vertex (3)

12.21.1 Detailed Description

Definition at line 711 of file type.h.

12.21.2 Member Data Documentation

12.21.2.1 header

Header agvc_interface::MapVirtualArea::header

id id

Definition at line 713 of file type.h.

12.21.2.2 shape

MapVirtualAreaShape agvc_interface::MapVirtualArea::shape

Definition at line 715 of file type.h.

12.21.2.3 type

MapVirtualAreaType agvc_interface::MapVirtualArea::type

Definition at line 714 of file type.h.

12.21.2.4 vertex

std::vector<Point2d> agvc_interface::MapVirtualArea::vertex

{{x(m) y(m)},{}, ...} vertex (3)

Definition at line 717 of file type.h.

12.21.2.5 width_speed

double agvc_interface::MapVirtualArea::width_speed

/ /

Definition at line 716 of file type.h.

The documentation for this struct was generated from the following file:

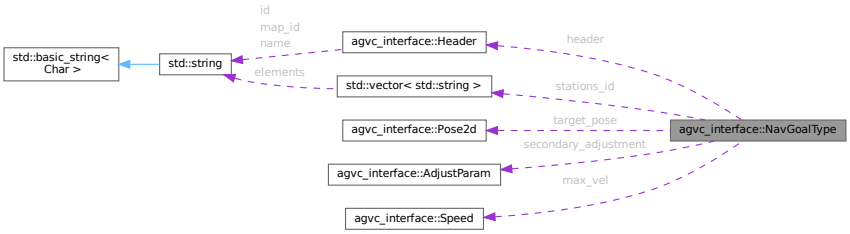
- include/agvc_interface/type.h

12.22 agvc_interface::NavGoalType Struct Reference

agv

#include <type.h>

Collaboration diagram for agvc_interface::NavGoalType:



Public Attributes

- Header header
  id id
- NavType type
- Pose2d target_pose
  {x(m) y(m) theta(rad)}
- std::vector< std::string > stations_id
  id,
- bool forward_flag
  agv
- bool goal_end_rotate
  true false
- AdjustParam secondary_adjustment
  , true false
- Speed max_vel
  v,w

12.22.1 Detailed Description

agv

Definition at line 762 of file type.h.

12.22.2 Member Data Documentation

12.22.2.1 forward_flag

bool agvc_interface::NavGoalType::forward_flag

agv

Definition at line 770 of file type.h.

12.22.2.2 goal_end_rotate

bool agvc_interface::NavGoalType::goal_end_rotate

true false

Definition at line 771 of file type.h.

12.22.2.3 header

Header agvc_interface::NavGoalType::header

id id

Definition at line 764 of file type.h.

12.22.2.4 max_vel

Speed agvc_interface::NavGoalType::max_vel

v,w

Definition at line 775 of file type.h.

12.22.2.5 secondary_adjustment

AdjustParam agvc_interface::NavGoalType::secondary_adjustment

, true false

Definition at line 773 of file type.h.

12.22.2.6 stations_id

std::vector<std::string> agvc_interface::NavGoalType::stations_id

id,

Definition at line 768 of file type.h.

12.22.2.7 target_pose

Pose2d agvc_interface::NavGoalType::target_pose

{x(m) y(m) theta(rad)}

Definition at line 766 of file type.h.

12.22.2.8 type

NavType agvc_interface::NavGoalType::type

Definition at line 765 of file type.h.

The documentation for this struct was generated from the following file:

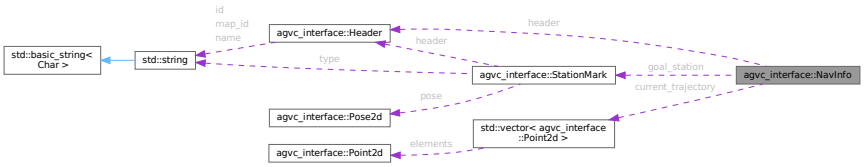
- include/agvc_interface/type.h

12.23 agvc_interface::NavInfo Struct Reference

/

#include <type.h>

Collaboration diagram for agvc_interface::NavInfo:



Public Attributes

- Header header
  id id
- NavStatus status

- NavType type
- NavType nav_type

- AutoChargingType auto_charging_type

- StationMark goal_station
  agv
- std::vector< Point2d > current_trajectory
  {{x(m) y(m)},{}, ...}

12.23.1 Detailed Description

/

type current_trajectory

Definition at line 784 of file type.h.

12.23.2 Member Data Documentation

12.23.2.1 auto_charging_type

AutoChargingType agvc_interface::NavInfo::auto_charging_type

Definition at line 791 of file type.h.

12.23.2.2 current_trajectory

std::vector<Point2d> agvc_interface::NavInfo::current_trajectory

{{x(m) y(m)},{}, ...}

Definition at line 793 of file type.h.

12.23.2.3 goal_station

StationMark agvc_interface::NavInfo::goal_station

agv

Definition at line 792 of file type.h.

12.23.2.4 header

Header agvc_interface::NavInfo::header

id id

Definition at line 786 of file type.h.

12.23.2.5 nav_type

NavType agvc_interface::NavInfo::nav_type

Definition at line 790 of file type.h.

12.23.2.6 status

NavStatus agvc_interface::NavInfo::status

Definition at line 787 of file type.h.

12.23.2.7 type

NavType agvc_interface::NavInfo::type

Definition at line 789 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.24 agvc_interface::OutputBuilder Class Reference

#include <rtde.h>

Public Member Functions

- OutputBuilder ()
- ~OutputBuilder ()
- OutputBuilder & push (int val)
- OutputBuilder & push (double val)
- OutputBuilder & push (const std::vector< double > &val)
- OutputBuilder & push (const std::tuple< int, bool > &val)
- OutputBuilder & push (int16_t &val)
- OutputBuilder & push (const std::vector< int16_t > &val)
- OutputBuilder & push (const std::vector< int > &val)
- OutputBuilder & push (const std::string &val)
- OutputBuilder & push (char val)
- OutputBuilder & push (const agvc_interface::RtdeRecipe &val)

Private Attributes

- friend RtdeClient
- Impl * impl

12.24.1 Detailed Description

Definition at line 23 of file rtde.h.

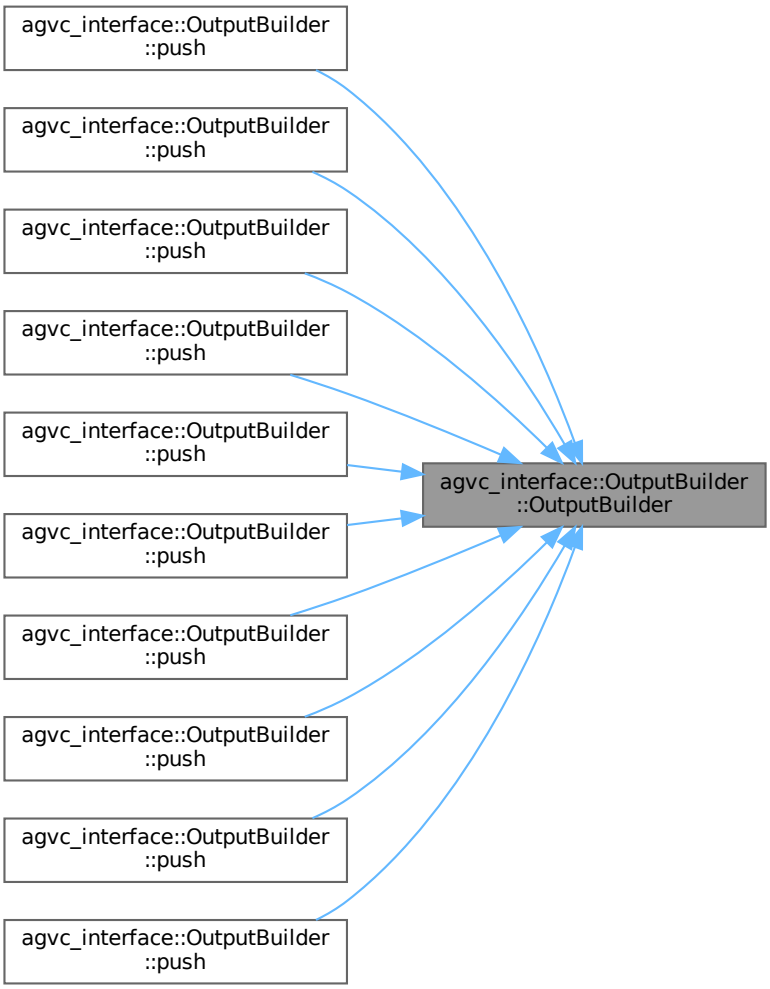
12.24.2 Constructor & Destructor Documentation

12.24.2.1 OutputBuilder()

agvc_interface::OutputBuilder::OutputBuilder ()

Referenced by push(), push(), push(), push(), push(), push(), push(), push(), push(), and push().

Here is the caller graph for this function:



12.24.2.2 ~OutputBuilder()

agvc_interface::OutputBuilder::~OutputBuilder ()

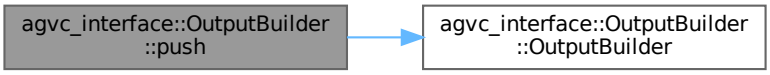
12.24.3 Member Function Documentation

12.24.3.1 push() [1/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
char val)

References OutputBuilder().

Here is the call graph for this function:

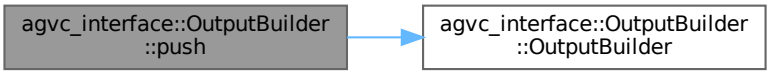


12.24.3.2 push() [2/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
const agvc_interface::RtdeRecipe & val)

References OutputBuilder().

Here is the call graph for this function:

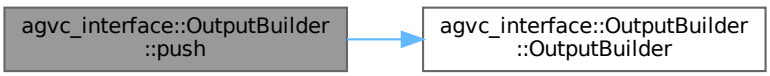


12.24.3.3 push() [3/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
const std::string & val)

References OutputBuilder().

Here is the call graph for this function:



12.24.3.4 push() [4/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
const std::tuple< int, bool > & val)

References OutputBuilder().

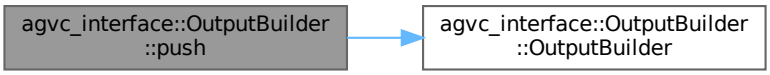
Here is the call graph for this function:

12.24.3.5 push() [5/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
const std::vector< double > & val)

References OutputBuilder().

Here is the call graph for this function:

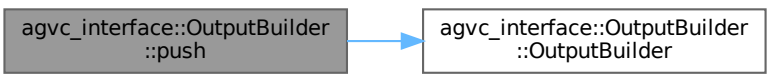


12.24.3.6 push() [6/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
const std::vector< int > & val)

References OutputBuilder().

Here is the call graph for this function:



12.24.3.7 push() [7/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
const std::vector< int16_t > & val)

References OutputBuilder().

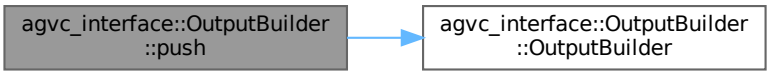
Here is the call graph for this function:

12.24.3.8 push() [8/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
double val)

References OutputBuilder().

Here is the call graph for this function:

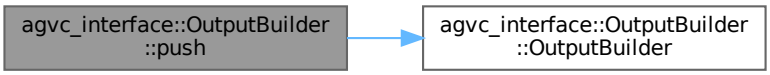


12.24.3.9 push() [9/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
int val)

References OutputBuilder().

Here is the call graph for this function:

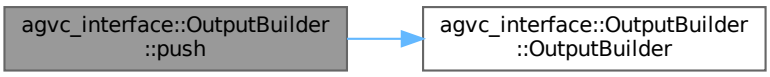


12.24.3.10 push() [10/10]

OutputBuilder & agvc_interface::OutputBuilder::push (
int16_t & val)

References OutputBuilder().

Here is the call graph for this function:



12.24.4 Member Data Documentation

12.24.4.1 impl

Impl* agvc_interface::OutputBuilder::impl [private]

Definition at line 44 of file rtde.h.

12.24.4.2 RtdeClient

friend agvc_interface::OutputBuilder::RtdeClient [private]

Definition at line 42 of file rtde.h.

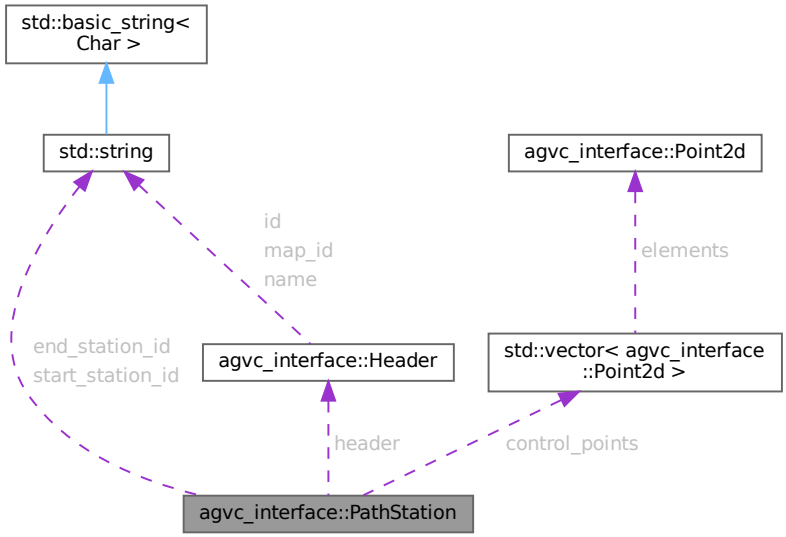
The documentation for this class was generated from the following file:

- include/agvc_interface/rtde.h

12.25 agvc_interface::PathStation Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::PathStation:



Public Attributes

- Header header
  id id
- PathShape shape
- bool use_direction
- double max_speed
  agv
- std::string start_station_id
  id
- std::string end_station_id
  id
- std::vector< Point2d > control_points
  {{x,y},{}, ...}

12.25.1 Detailed Description

Definition at line 651 of file type.h.

12.25.2 Member Data Documentation

12.25.2.1 control_points

std::vector<Point2d> agvc_interface::PathStation::control_points

{{x,y},{}, ...}

Definition at line 659 of file type.h.

12.25.2.2 end_station_id

std::string agvc_interface::PathStation::end_station_id

id

Definition at line 658 of file type.h.

12.25.2.3 header

Header agvc_interface::PathStation::header

id id

Definition at line 653 of file type.h.

12.25.2.4 max_speed

double agvc_interface::PathStation::max_speed

agv

Definition at line 656 of file type.h.

12.25.2.5 shape

PathShape agvc_interface::PathStation::shape

Definition at line 654 of file type.h.

12.25.2.6 start_station_id

std::string agvc_interface::PathStation::start_station_id

id

Definition at line 657 of file type.h.

12.25.2.7 use_direction

bool agvc_interface::PathStation::use_direction

Definition at line 655 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.26 agvc_interface::Point2d Struct Reference

#include <type.h>

Public Attributes

- float x
  m
- float y
  m

12.26.1 Detailed Description

Definition at line 397 of file type.h.

12.26.2 Member Data Documentation

12.26.2.1 x

float agvc_interface::Point2d::x

m

Definition at line 399 of file type.h.

12.26.2.2 y

float agvc_interface::Point2d::y

m

Definition at line 400 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.27 agvc_interface::Pose2d Struct Reference

#include <type.h>

Public Attributes

- double x
  m
- double y
  m
- double yaw
  rad

12.27.1 Detailed Description

Definition at line 406 of file type.h.

12.27.2 Member Data Documentation

12.27.2.1 x

double agvc_interface::Pose2d::x

m

Definition at line 408 of file type.h.

12.27.2.2 y

double agvc_interface::Pose2d::y

m

Definition at line 409 of file type.h.

12.27.2.3 yaw

double agvc_interface::Pose2d::yaw

rad

Definition at line 410 of file type.h.

The documentation for this struct was generated from the following file:

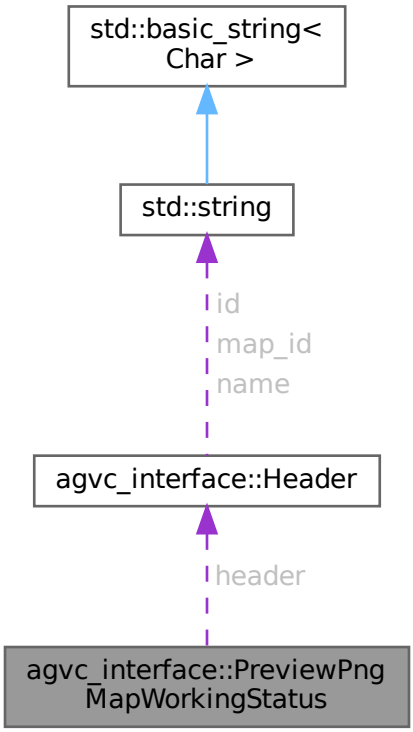
- include/agvc_interface/type.h

12.28 agvc_interface::PreviewPngMapWorkingStatus Struct Reference

previewPngMapFromAgv

#include <type.h>

Collaboration diagram for agvc_interface::PreviewPngMapWorkingStatus:



Public Attributes

- Header header
  header.id previewPngMapFromAgv id
- PreviewPngMapStatus status
  previewPngMapFromAgv NONE - RUNNING- FINISH-

12.28.1 Detailed Description

previewPngMapFromAgv

Definition at line 596 of file type.h.

12.28.2 Member Data Documentation

12.28.2.1 header

Header agvc_interface::PreviewPngMapWorkingStatus::header

header.id previewPngMapFromAgv id

Definition at line 598 of file type.h.

12.28.2.2 status

PreviewPngMapStatus agvc_interface::PreviewPngMapWorkingStatus::status

previewPngMapFromAgv NONE - RUNNING- FINISH-

Definition at line 599 of file type.h.

The documentation for this struct was generated from the following file:

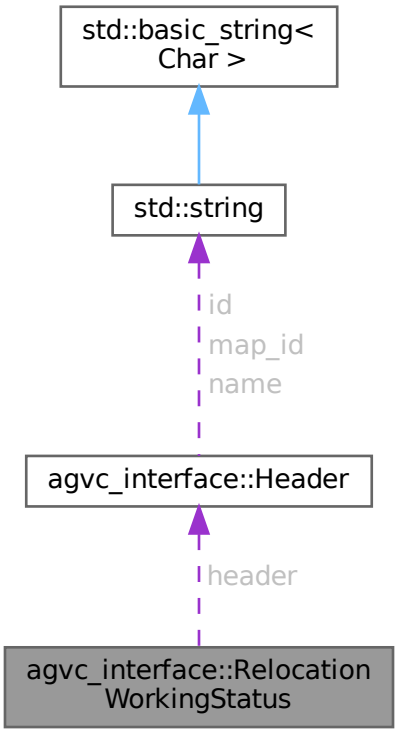
- include/agvc_interface/type.h

12.29 agvc_interface::RelocationWorkingStatus Struct Reference

Relocation

#include <type.h>

Collaboration diagram for agvc_interface::RelocationWorkingStatus:



Public Attributes

- Header header
  header.id Relocation id
- RelocationStatus status
  Relocation NONE - RUNNING- FINISH-

12.29.1 Detailed Description

Relocation

Definition at line 516 of file type.h.

12.29.2 Member Data Documentation

12.29.2.1 header

Header agvc_interface::RelocationWorkingStatus::header

header.id Relocation id

Definition at line 518 of file type.h.

12.29.2.2 status

RelocationStatus agvc_interface::RelocationWorkingStatus::status

Relocation NONE - RUNNING- FINISH-

Definition at line 519 of file type.h.

The documentation for this struct was generated from the following file:

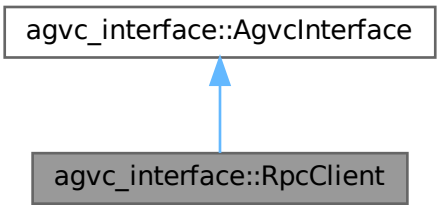
- include/agvc_interface/type.h

12.30 agvc_interface::RpcClient Class Reference

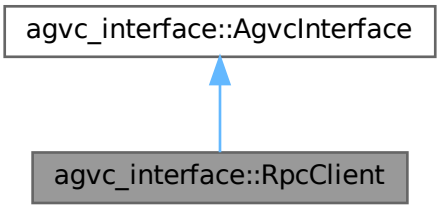
RPC

#include <rpc.h>

Inheritance diagram for agvc_interface::RpcClient:



Collaboration diagram for agvc_interface::RpcClient:



Public Types

- enum Event { Connected , Disconnected }

Public Member Functions

- RpcClient (int mode=0)
  RpcClient.
- ~RpcClient ()
- void setLogHandler (std::function< void(int, const char *, int, const std::string &)> handler)
- int connect (const std::string &ip="", int port=0)
  RPC
- int disconnect ()

RPC

- bool hasConnected () const

  RPC
- int login (const std::string &usrname, const std::string &passwd)
- int logout ()
- bool hasLogined ()
- int setRequestTimeout (int timeout=10)

  RPC
- int setEventHandler (std::function< void(int)> cb)
- int setExceptionFree (bool enable)
- int errorCode () const

Public Member Functions inherited from agvc_interface::AgvcInterface

- AgvcInterface ()
- virtual ~AgvcInterface ()
- int setSystemClock (const std::string &stamp)

  Sets the AGV system time.
- AgvDetails getAgvDetails ()

  Queries current AGV information.
- RunningInfo getRunningInfo ()

  Queries current AGV running information.
- Pose2d getAgvCurrentPose ()

  Queries the current AGV pose.
- std::vector< Point2d > getLaserPointCloud ()

  Queries the current laser point cloud data.
- AsyncInterfaceResultStatus getAsyncInterfaceResultStatus ()

  Queries the running status of asynchronous APIs; this does not represent each API's own execution result.
- std::vector< StationMark > getAllStations ()

  Queries all station information on the current map.
- std::vector< StationMark > getAllStationsOfTargetMap (const Header &map_header)

  Queries all station information on the specified map.
- int addStation (const StationMark &station)

  Adds or modifies one station.
- int addStations (const std::vector< StationMark > &stations)

  Adds or modifies multiple stations.
- int addCPStationUseChargingBoard (const Header &station_header, const double &dis_station↩ _board=1.5)

  Adds a charging station by automatically recognizing the charging board pose.
- int deleteStation (const Header &station_header)

  Deletes the specified station and its related paths.
- int deleteStations (const std::vector< Header > &stations_header)

  Deletes multiple stations and their related paths.
- std::vector< PathStation > getAllPaths ()

Queries all path information on the current map.

- std::vector< PathStation > getAllPathsOfTargetMap (const Header &map_header)

  Queries all path information on the specified map.
- PathStation getCurrentPath ()

  Queries the path currently being tracked by the AGV.
- int generatePath (const PathStation &path_station)

  Generates or modifies one path.
- int generatePaths (const std::vector< PathStation > &paths_station)

  Generates or modifies multiple paths.
- int deletePath (const Header &path_header)

  Deletes one path.
- int deletePaths (const std::vector< Header > &paths_header)

  Deletes multiple paths.
- std::vector< Header > getMapList ()

  Gets the headers of all maps on the AGV.
- Header getCurrentMapHeader ()

  Queries the header of the current AGV map.
- OccupancyGridMap getGridMapFromAgv (const Header &map_header)

  Gets the specified occupancy grid map information.
- Base64PngMap getBase64PngMapFromAgv (const Header &map_header)

  Gets Base64-encoded PNG map information.
- Base64PngMap previewPngMapFromAgv (const Header &map_header, const int &image_width_px=0, const int &image_height_px=0)

  Gets a preview image with optional width and height.
- int sendGridMapToAgv (const OccupancyGridMap &map)

  Sends occupancy grid map information to the AGV.
- int sendBase64PngMapToAgv (const Base64PngMap &map)

  Sends a Base64-encoded PNG map to the AGV.
- int saveMap (const Header &map_header)

  Saves the map after mapping is complete.
- int switchMap (const Header &map_header)

  Switches to the specified map.
- int deleteMap (const Header &map_header)

  Deletes one specified map.
- int deleteMaps (const std::vector< Header > &map_headers)

  Deletes multiple specified maps.
- std::vector< MapVirtualArea > getAllMapVirtualArea ()

  Queries virtual areas on the current map.
- std::vector< MapVirtualArea > getAllMapVirtualAreaOfTargetMap (const Header &map_header)

  Queries virtual areas on the target map.
- int addMapVirtualArea (const MapVirtualArea &map_virtual_area)

  Adds or modifies one virtual area.
- int addMapVirtualAreas (const std::vector< MapVirtualArea > &map_virtual_areas)

  Adds or modifies multiple virtual areas.
- int deleteMapVirtualArea (const Header &virtual_area_header)

  Deletes the specified virtual area.
- int deleteMapVirtualAreas (const std::vector< Header > &virtual_areas_header)

  Deletes multiple specified virtual areas.
- MapAllInfo getGridMapAllInfo (const Header &map_header)

  Gets occupancy grid map data, paths, stations, and virtual areas for the specified map.
- MapAllInfo getPngMapAllInfo (const Header &map_header)

Gets Base64 map data, paths, stations, and virtual areas for the specified map.

- int setGridMapAllInfo (const MapAllInfo &map_all_info, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Sets occupancy grid map, path, station, and virtual area information.
- int setPngMapAllInfo (const MapAllInfo &map_all_info, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Sets Base64 map, path, station, and virtual area information.
- int deleteMapAllInfo (const Header &map_header)

  Deletes all information of the specified map, including map data, stations, paths, and virtual areas.
- std::vector< std::string > getWifiList ()

  Gets the currently scanned WiFi list.
- int connectWifi (const std::string &ssid, const std::string &password)

  Connects to the specified WiFi and automatically switches to WiFi mode.
- int enableHotspot (const std::string &password)

  Enables hotspot mode and automatically switches to hotspot mode.
- std::vector< IpAddressInfo > getIpAddressList ()

  Gets all local IP addresses.
- int changeRunningMode (const RunningMode &running_mode, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Switches the AGV running mode.
- int setControlSpeed (const Speed &speed)

  Controls AGV motion.
- int setNavGoal (const NavGoalType &target)

  Sends a navigation goal to the AGV.
- NavInfo getNavInfo ()

  Gets current navigation information.
- int pauseAgvSpeed ()

  Pauses AGV speed.
- int resumeAgvSpeed ()

  Resumes AGV speed after a pause.
- int cancelNavigation ()

  Cancels the navigation task.
- int relocation (const Pose2d &init_pose, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Relocates the AGV.
- std::string getAgvControllerParametersFile ()

  Gets the AGV controller parameter file.
- int setAgvControllerParametersFile (const std::string &agv_parameters)

  Sets the AGV controller parameter file.
- int refreshAgvControllerParametersFile ()

  Refreshes parameters on AGV nodes after the AGV controller parameter file is modified.
- int resetAgvControllerParametersFile ()

  Restores the AGV controller parameter file to factory defaults.
- int checkPowerAndAutoCharge (const Header &check_power_header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &nav_board_charge_station_id=”99999”)

  Checks the current battery level and automatically docks for charging when the battery is low.
- int setLeaveBoardTargetStation (const Header &leave_board_target_station_header)

  Sets the target station for automatic undocking.
- Header getLeaveBoardTargetStation ()

  Gets the target station for automatic undocking.

- int forcedAgvToChargingBoard (const Header &forced_charge_header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &nav_board_charge_station_id=”99999”)

  Forces the AGV to dock for charging.
- int forcedAgvLeaveChargingBoard (const Header &forced_leave_header={ ”99999”, ”99999”, 99999, ”99999” }, const std::string &leave_board_target_station_id=”99999”)

  Forces the AGV to leave the charging board.
- int sendAutoChargingCommand (const AutoChargingCommand &auto_charging_command)

  Sends an automatic charging command.
- std::string getAgvLogMessage ()

  Gets AGV runtime log messages.
- int updateSoftware (const std::string &upgrade_pack_path)

  Upgrades AGV software.
- std::vector< std::string > getSoftwareVersionList ()

  Gets the AGV software version list.
- int switchSoftwareVersion (const std::string &software_version)

  Switches the AGV software version.
- int uninstallSoftware (const std::string &software_version)

  Uninstalls an AGV software version.
- int updateFirmware (const FirmwareUpdateParam &update_firmware)

  Updates AGV firmware.
- FirmwareUpdateProcessInfo getUpdateFirmwareProcess ()

  Gets firmware update process information.
- int startCollectCalibrationData (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Starts collecting calibration data.
- int cancelCollectCalibrationData (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Cancels calibration data collection.
- int startCalibration (const CalibrationType &type, const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Starts calibration.
- CalibrationProcessInfo getCalibrationProcessInfo (const CalibrationType &type)

  Gets calibration information.
- int restartAgv ()

  Restarts the AGV.
- int setAgvName (const std::string &agv_name)

  Sets the AGV name.
- int setPriority (const std::string &name, const std::string &ip=””)

  Sets control priority.
- int releasePriority ()

  Releases control priority.
- int scriptPaused ()

  Pauses the script.
- int scriptResume ()

  Resumes the script.
- int scriptStop ()

  Stops the script.
- ScriptRuntimeState getScriptStatus ()

  Gets the script runtime state.
- int setScriptStatus (ScriptRuntimeState status)

  Sets the script runtime state.

- std::string errorCodeDecoder (const int &error_code)

  Queries an error code.
- std::vector< Header > getCurrentErrorCodes ()

  Gets current error codes.
- int soundLightPrompt ()

  Locates the AGV by automatically playing audio and flashing special lights.
- bool isObstacleAhead (const double &detect_distance=1.0)

  Checks whether there is an obstacle in the AGV travel direction.
- StationMark getNearestStation (const double &detect_distance=1.0)

  Gets the nearest station within the specified detection distance.
- int autoAlignRailway (const Header &command_header={ ”99999”, ”99999”, 1, ”99999” })

  Automatically aligns with the rail and boards the rail.

12.30.1 Detailed Description

RPC

Definition at line 17 of file rpc.h.

12.30.2 Member Enumeration Documentation

12.30.2.1 Event

enum agvc_interface::RpcClient::Event

Enumerator

Connected	
Disconnected	

Definition at line 20 of file rpc.h.

12.30.3 Constructor & Destructor Documentation

12.30.3.1 RpcClient()

agvc_interface::RpcClient::RpcClient (
int mode = 0)

RpcClient.

Parameters

mode	0-TCP 1-UDS

12.30.3.2 ~RpcClient()

agvc_interface::RpcClient::~RpcClient ()

12.30.4 Member Function Documentation

12.30.4.1 connect()

int agvc_interface::RpcClient::connect (
const std::string & ip = "",
int port = 0)

RPC

Parameters

ip	IP
port	RPC 30104
ip port unix	domain sockets

Return values

0	RPC
-8	RPC RPC
-15	RPC SDK Server

12.30.4.2 disconnect()

int agvc_interface::RpcClient::disconnect ()

RPC

Return values

0	
-1	

12.30.4.3 errorCode()

int agvc_interface::RpcClient::errorCode () const

Returns

12.30.4.4 hasConnected()

bool agvc_interface::RpcClient::hasConnected () const

RPC

Return values

true	RPC
false	RPC

12.30.4.5 hasLogined()

bool agvc_interface::RpcClient::hasLogined ()

Return values

true	
false	

12.30.4.6 login()

int agvc_interface::RpcClient::login (
const std::string & usrname,
const std::string & passwd)

Parameters

usrname	
passwd	

Returns

0

12.30.4.7 logout()

int agvc_interface::RpcClient::logout ()

Returns

0

12.30.4.8 setEventHandler()

int agvc_interface::RpcClient::setEventHandler (
std::function< void(int)> cb)

Parameters

cb	

Returns

12.30.4.9 setExceptionFree()

int agvc_interface::RpcClient::setExceptionFree (
bool enable)

Parameters

enable	

Returns

12.30.4.10 setLogHandler()

void agvc_interface::RpcClient::setLogHandler (
std::function< void(int, const char *, int, const std::string &)> handler)

Agvc SDK Agvc SDK

Note

setLogHandler

Parameters

handler	: void handler(int level, const char* filename, int line, const std::string& message) level 0: LOGLEVEL_FATAL 1: LOGLEVEL_ERROR 2: LOGLEVEL_WARNING 3: LOGLEVEL_INFO 4: LOGLEVEL_DEBUG 5: LOGLEVEL_BACKTRACE filename line message

Returns

12.30.4.11 setRequestTimeout()

int agvc_interface::RpcClient::setRequestTimeout (
int timeout = 10)

RPC

Parameters

timeout	ms

Returns

0

The documentation for this class was generated from the following file:

- include/agvc_interface/rpc.h

12.31 agvc_interface::RtdeClient Class Reference

RTDE

#include <rtde.h>

Public Types

- enum Event { Connected , Disconnected }

Public Member Functions

- RtdeClient (int mode=0)
  RTDE
- ~RtdeClient ()
- void setLogHandler (std::function< void(int, const char *, int, const std::string &)> handler)
- int connect (const std::string &ip="", int port=0)
- bool hasConnected () const
  socket
- bool hasConnected1 (std::function< void(bool)> callback)
  socket
- int login (const std::string &usrname, const std::string &passwd)
- bool hasLogined ()
- int logout ()
- int disconnect ()
- int getProtocolVersion ()
- std::map< std::string, int > getInputMaps ()

- std::map< std::string, int > getOutputMaps ()
- int setTopic (bool to_server, const std::vector< std::string > &names, double freq, int expected_chanel)
- int removeTopic (bool to_server, int chanel)
- std::unordered_map< int, agvc_interface::RtdeRecipe > getRegisteredInputRecipe ()
- std::unordered_map< int, agvc_interface::RtdeRecipe > getRegisteredOutputRecipe ()
- int subscribe (int chanel, std::function< void(InputParser &)> callback)
  subscribe from output
- int publish (int chanel, std::function< void(OutputBuilder &)> callback)
  publish to input
- int setEventHandler (std::function< void(int)> cb)

Private Attributes

- Impl * impl

12.31.1 Detailed Description

RTDE

Definition at line 79 of file rtde.h.

12.31.2 Member Enumeration Documentation

12.31.2.1 Event

enum agvc_interface::RtdeClient::Event

Enumerator

Connected	
Disconnected	

Definition at line 82 of file rtde.h.

12.31.3 Constructor & Destructor Documentation

12.31.3.1 RtdeClient()

agvc_interface::RtdeClient::RtdeClient (
int mode = 0)

Parameters

mode	0 tcp 1 uds

12.31.3.2 ~RtdeClient()

agvc_interface::RtdeClient::~RtdeClient ()

12.31.4 Member Function Documentation

12.31.4.1 connect()

int agvc_interface::RtdeClient::connect (
const std::string & ip = "",
int port = 0)

Parameters

ip	IP
port	RTDE 30110

Return values

0	
1	
-1	

12.31.4.2 disconnect()

int agvc_interface::RtdeClient::disconnect ()

Return values

0	
-1	

12.31.4.3 getInputMaps()

std::map< std::string, int > agvc_interface::RtdeClient::getInputMaps ()

Returns

12.31.4.4 getOutputMaps()

std::map< std::string, int > agvc_interface::RtdeClient::getOutputMaps ()

Returns

12.31.4.5 getProtocolVersion()

int agvc_interface::RtdeClient::getProtocolVersion ()

Returns

12.31.4.6 getRegisteredInputRecipe()

std::unordered_map< int, agvc_interface::RtdeRecipe > agvc_interface::RtdeClient::getRegisteredInputRecipe ()

Returns

12.31.4.7 getRegisteredOutputRecipe()

std::unordered_map< int, agvc_interface::RtdeRecipe > agvc_interface::RtdeClient::getRegisteredOutputRecipe ()

Return values

true	socket
false	socket

12.31.4.9 hasConnected1()

bool agvc_interface::RtdeClient::hasConnected1 (
std::function< void(bool)> callback)

socket

Parameters

callback	

Return values

true	socket
false	socket

12.31.4.10 hasLogined()

bool agvc_interface::RtdeClient::hasLogined ()

Return values

true	
false	

12.31.4.11 login()

int agvc_interface::RtdeClient::login (
const std::string & usrname,
const std::string & passwd)

Parameters

usrname	
passwd	

Return values

0	
-1	

12.31.4.12 logout()

int agvc_interface::RtdeClient::logout ()

Returns

0

12.31.4.13 publish()

int agvc_interface::RtdeClient::publish (
int chanel,
std::function< void(OutputBuilder &)> callback)

publish to input

Parameters

chanel	
callback	void callback(OutputBuilder &builder)

Return values

0	

-1	

12.31.4.14 removeTopic()

int agvc_interface::RtdeClient::removeTopic (
bool to_server,
int chanel)

Parameters

to_server	true false
chanel	

Return values

0	
1	

12.31.4.15 setEventHandler()

int agvc_interface::RtdeClient::setEventHandler (
std::function< void(int)> cb)

Parameters

cb	

Returns

12.31.4.16 setLogHandler()

```
void agvc_interface::RtdeClient::setLogHandler (
        std::function< void(int, const char *, int, const std::string &)> handler)
```

Aubo SDK Aubo SDK

Note

setLogHandler

Parameters

handler	: void handler(int level, const char* filename, int line, const std::string& message) level 0: LOGLEVEL_FATAL 1: LOGLEVEL_ERROR 2: LOGLEVEL_WARNING 3: LOGLEVEL_INFO 4: LOGLEVEL_DEBUG 5: LOGLEVEL_BACKTRACE filename line message

Returns

12.31.4.17 setTopic()

```
int agvc_interface::RtdeClient::setTopic (
        bool to_server,
        const std::vector< std::string > & names,
        double freq,
        int expected_chanel)
```

Parameters

to_server	true	false
names		
freq		
expected_chanel	0~99	

Return values

expected_chanel	
-1	

12.31.4.18 subscribe()

```
int agvc_interface::RtdeClient::subscribe (
        int chanel,
        std::function< void(InputParser &)> callback)
```

subscribe from output

Parameters

chanel	
callback	void callback(InputParser &parser)

Returns

0

12.31.5 Member Data Documentation

12.31.5.1 impl

```
Impl* agvc_interface::RtdeClient::impl  [private]
```

Definition at line 282 of file rtde.h.

The documentation for this class was generated from the following file:

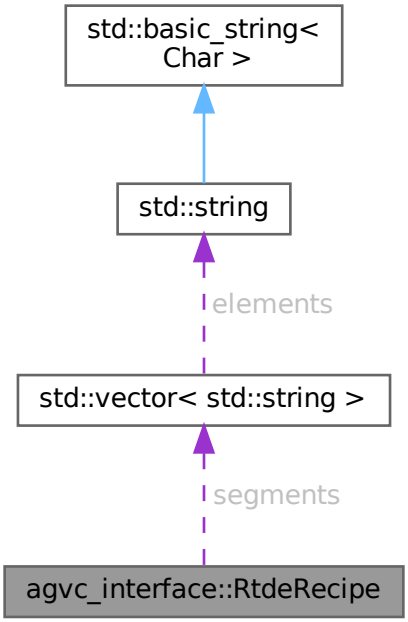
- include/agvc_interface/rtde.h

12.32 agvc_interface::RtdeRecipe Struct Reference

RTDE

#include <type.h>

Collaboration diagram for agvc_interface::RtdeRecipe:



Public Attributes

- bool to_server
  /
- int chanel
- double frequency
- int trigger
  ( ): 0 - ; 1 -
- std::vector< std::string > segments

12.32.1 Detailed Description

RTDE

Definition at line 841 of file type.h.

12.32.2 Member Data Documentation

12.32.2.1 chanel

int agvc_interface::RtdeRecipe::chanel

Definition at line 844 of file type.h.

12.32.2.2 frequency

double agvc_interface::RtdeRecipe::frequency

Definition at line 845 of file type.h.

12.32.2.3 segments

std::vector<std::string> agvc_interface::RtdeRecipe::segments

Definition at line 847 of file type.h.

12.32.2.4 to_server

bool agvc_interface::RtdeRecipe::to_server

/

Definition at line 843 of file type.h.

12.32.2.5 trigger

int agvc_interface::RtdeRecipe::trigger

( ): 0 - ; 1 -

Definition at line 846 of file type.h.

The documentation for this struct was generated from the following file:

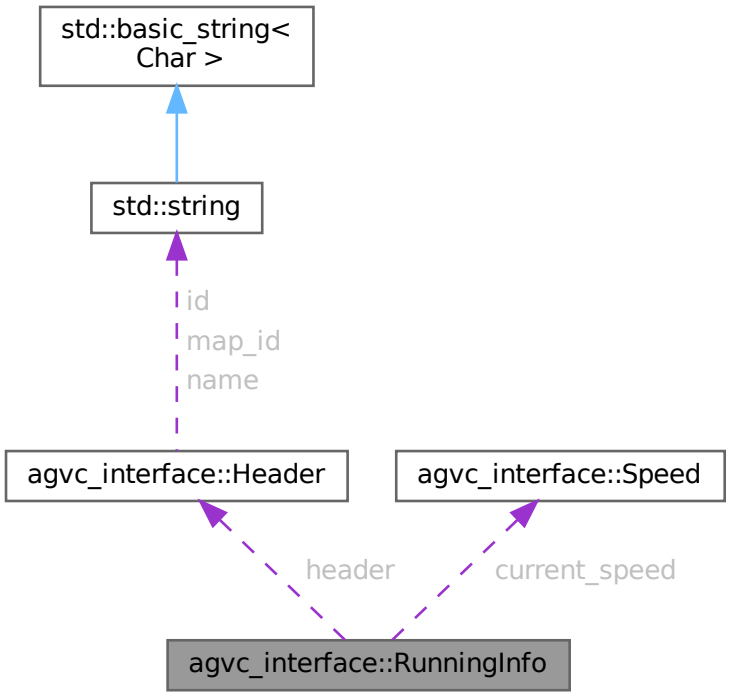
- include/agvc_interface/type.h

12.33 agvc_interface::RunningInfo Struct Reference

agv

#include <type.h>

Collaboration diagram for agvc_interface::RunningInfo:



Public Attributes

- Header header
  agv SN agv id
- RunningMode running_mode
  agv
- Speed current_speed
  {v w} {m/s rad/s}
- int8_t location_score
  [0-100]
- double total_dist
  m
- double current_dist
  m
- double total_running_time
  s

- double current_running_time
  s
- double battery_voltage
  V
- double battery_current
  A
- int8_t remaining_voltage
  % [0-100]
- int8_t battery_status
  0- ,1- 2- 3- 4-
- int8_t safety_edge_status
  0- ,1- 2- 3- 4-
- bool low_battery
- bool pause
- bool emergency_stop
- bool blocked
- int8_t obstacle_level
  0- 1- 2- 3-
- bool localization_loss
- bool fault_alarm
  ,
- bool break_release_flag
  true false

12.33.1 Detailed Description

agv

Definition at line 460 of file type.h.

12.33.2 Member Data Documentation

12.33.2.1 battery_current

double agvc_interface::RunningInfo::battery_current

A

Definition at line 471 of file type.h.

12.33.2.2 battery_status

int8_t agvc_interface::RunningInfo::battery_status

0- ,1- 2- 3- 4-

Definition at line 473 of file type.h.

12.33.2.3 battery_voltage

double agvc_interface::RunningInfo::battery_voltage

V

Definition at line 470 of file type.h.

12.33.2.4 blocked

bool agvc_interface::RunningInfo::blocked

Definition at line 478 of file type.h.

12.33.2.5 break_release_flag

bool agvc_interface::RunningInfo::break_release_flag

true false

Definition at line 482 of file type.h.

12.33.2.6 current_dist

double agvc_interface::RunningInfo::current_dist

m

Definition at line 467 of file type.h.

12.33.2.7 current_running_time

double agvc_interface::RunningInfo::current_running_time

s

Definition at line 469 of file type.h.

12.33.2.8 current_speed

Speed agvc_interface::RunningInfo::current_speed

{v w} {m/s rad/s}

Definition at line 464 of file type.h.

12.33.2.9 emergency_stop

bool agvc_interface::RunningInfo::emergency_stop

Definition at line 477 of file type.h.

12.33.2.10 fault_alarm

bool agvc_interface::RunningInfo::fault_alarm

,

Definition at line 481 of file type.h.

12.33.2.11 header

Header agvc_interface::RunningInfo::header

agv SN agv id

Definition at line 462 of file type.h.

12.33.2.12 localization_loss

bool agvc_interface::RunningInfo::localization_loss

Definition at line 480 of file type.h.

12.33.2.13 location_score

int8_t agvc_interface::RunningInfo::location_score

[0-100]

Definition at line 465 of file type.h.

12.33.2.14 low_battery

bool agvc_interface::RunningInfo::low_battery

Definition at line 475 of file type.h.

12.33.2.15 obstacle_level

int8_t agvc_interface::RunningInfo::obstacle_level

0- 1- 2- 3-

Definition at line 479 of file type.h.

12.33.2.16 pause

bool agvc_interface::RunningInfo::pause

Definition at line 476 of file type.h.

12.33.2.17 remaining_voltage

int8_t agvc_interface::RunningInfo::remaining_voltage

% [0-100]

Definition at line 472 of file type.h.

12.33.2.18 running_mode

RunningMode agvc_interface::RunningInfo::running_mode

agv

Definition at line 463 of file type.h.

12.33.2.19 safety_edge_status

int8_t agvc_interface::RunningInfo::safety_edge_status

0- ,1- 2- 3- 4-

Definition at line 474 of file type.h.

12.33.2.20 total_dist

double agvc_interface::RunningInfo::total_dist

m

Definition at line 466 of file type.h.

12.33.2.21 total_running_time

double agvc_interface::RunningInfo::total_running_time

s

Definition at line 468 of file type.h.

The documentation for this struct was generated from the following file:

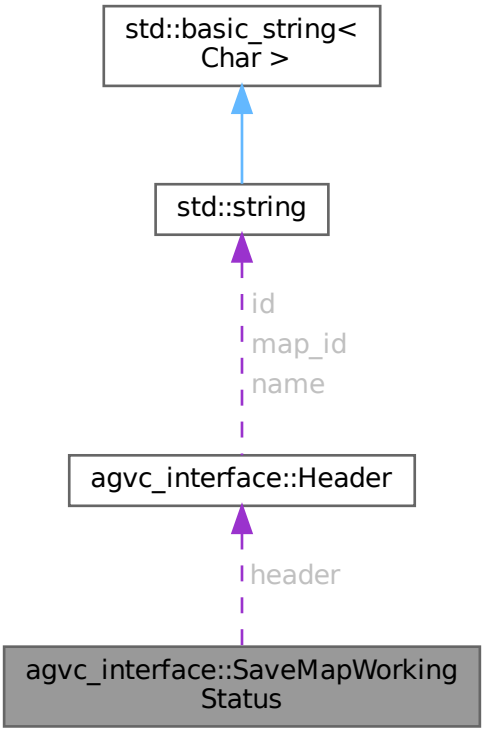
- include/agvc_interface/type.h

12.34 agvc_interface::SaveMapWorkingStatus Struct Reference

saveMap

#include <type.h>

Collaboration diagram for agvc_interface::SaveMapWorkingStatus:



Public Attributes

- Header header
  header.id saveMap id
- SaveMapStatus status
  SaveMap NONE - RUNNING- FINISH-

12.34.1 Detailed Description

saveMap

Definition at line 488 of file type.h.

12.34.2 Member Data Documentation

12.34.2.1 header

Header agvc_interface::SaveMapWorkingStatus::header

header.id saveMap id

Definition at line 490 of file type.h.

12.34.2.2 status

SaveMapStatus agvc_interface::SaveMapWorkingStatus::status

SaveMap NONE - RUNNING- FINISH-

Definition at line 491 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.35 agvc_interface::ScriptClient Class Reference

SCRIPT

#include <script.h>

Public Types

- enum Event { Connected , Disconnected }

Public Member Functions

- ScriptClient (int mode=0)
- ~ScriptClient ()
- void setLogHandler (std::function< void(int, const char *, int, const std::string &)> handler)
- int connect (const std::string &ip="", int port=0)
- bool hasConnected () const
- int login (const std::string &usrname, const std::string &passwd)
- bool hasLogined ()
- int logout ()
- int disconnect ()
- int sendFile (const std::string &path)
- int sendString (const std::string &script)
- int send (const std::string &chunck_name, std::function< int(ScriptWriter &)> cb)
  ScriptWriter
- int subscribeVariableUpdate (std::function< void(const std::string &, const std::string &)> cb)
- int subscribeScriptError (std::function< void(const std::string &)> cb)
- int subscribeScriptError2 (std::function< void(const std::string &, const std::string &)> cb)
- int setEventHandler (std::function< void(int)> cb)

Private Attributes

- Impl * impl

12.35.1 Detailed Description

SCRIPT

Definition at line 31 of file script.h.

12.35.2 Member Enumeration Documentation

12.35.2.1 Event

enum agvc_interface::ScriptClient::Event

Enumerator

Connected	
Disconnected	

Definition at line 34 of file script.h.

12.35.3 Constructor & Destructor Documentation

12.35.3.1 ScriptClient()

```
agvc_interface::ScriptClient::ScriptClient (
        int mode = 0)
```

12.35.3.2 ~ScriptClient()

```
agvc_interface::ScriptClient::~ScriptClient ()
```

12.35.4 Member Function Documentation

12.35.4.1 connect()

```
int agvc_interface::ScriptClient::connect (
        const std::string & ip = "",
        int port = 0)
```

Parameters

ip	IP
port	SCRIPT 30004

Return values

0	
1	
-1	

12.35.4.2 disconnect()

Return values

0	
-1	

12.35.4.3 hasConnected()

bool agvc_interface::ScriptClient::hasConnected () const

Return values

true	
false	

12.35.4.4 hasLogined()

bool agvc_interface::ScriptClient::hasLogined ()

Return values

true	
false	

12.35.4.5 login()

int agvc_interface::ScriptClient::login (
const std::string & usrname,
const std::string & passwd)

Parameters

usrname	
passwd	

Return values

0	
-1	

12.35.4.6 logout()

int agvc_interface::ScriptClient::logout ()

Returns

0

12.35.4.7 send()

int agvc_interface::ScriptClient::send (
const std::string & chunck_name,
std::function< int(ScriptWriter &)> cb)

ScriptWriter

Parameters

chunck_name	
cb	

Return values

0	
-1	

12.35.4.8 sendFile()

Parameters

path	

Return values

0	
-1	

12.35.4.9 sendString()

int agvc_interface::ScriptClient::sendString (
const std::string & script)

Parameters

script	

Return values

0	
-1	

12.35.4.10 setEventHandler()

int agvc_interface::ScriptClient::setEventHandler (
std::function< void(int)> cb)

Parameters

cb	

Returns

12.35.4.11 setLogHandler()

void agvc_interface::ScriptClient::setLogHandler (
std::function< void(int, const char *, int, const std::string &)> handler)

Aubo SDK Aubo SDK

Note

setLogHandler

Parameters

handler	: void handler(int level, const char* filename, int line, const std::string& message) level 0: LOGLEVEL_FATAL 1: LOGLEVEL_ERROR 2: LOGLEVEL_WARNING 3: LOGLEVEL_INFO 4: LOGLEVEL_DEBUG 5: LOGLEVEL_BACKTRACE filename line message

Returns

12.35.4.12 subscribeScriptError()

int agvc_interface::ScriptClient::subscribeScriptError (
std::function< void(const std::string &)> cb)

Parameters

cb	

Returns

0

12.35.4.13 subscribeScriptError2()

int agvc_interface::ScriptClient::subscribeScriptError2 (
std::function< void(const std::string &, const std::string &)> cb)

12.35.4.14 subscribeVariableUpdate()

int agvc_interface::ScriptClient::subscribeVariableUpdate (
std::function< void(const std::string &, const std::string &)> cb)

Parameters

cb	

Returns

0

12.35.5 Member Data Documentation

12.35.5.1 impl

Impl* agvc_interface::ScriptClient::impl [private]

Definition at line 186 of file script.h.

The documentation for this class was generated from the following file:

- include/agvc_interface/script.h

12.36 agvc_interface::ScriptWriter Class Reference

#include <script.h>

Public Member Functions

- virtual ~ScriptWriter ()=default
- virtual ScriptWriter & append (const std::string &line)=0
- virtual ScriptWriter & append (const char *buf, size_t len)=0
- virtual ScriptWriter & moveJoint ()=0
- virtual ScriptWriter & moveLine ()=0
- virtual ScriptWriter & ifCondition ()=0
- virtual ScriptWriter & elseCondition ()=0
- virtual ScriptWriter & elseIfCondition ()=0
- virtual ScriptWriter & whileCondition ()=0
- virtual ScriptWriter & end ()=0

12.36.1 Detailed Description

Definition at line 14 of file script.h.

12.36.2 Constructor & Destructor Documentation

12.36.2.1 ~ScriptWriter()

virtual agvc_interface::ScriptWriter::~ScriptWriter () [virtual], [default]

12.36.3 Member Function Documentation

12.36.3.1 append() [1/2]

virtual ScriptWriter & agvc_interface::ScriptWriter::append (
const char * buf,
size_t len) [pure virtual]

12.36.3.2 append() [2/2]

virtual ScriptWriter & agvc_interface::ScriptWriter::append (
const std::string & line) [pure virtual]

12.36.3.3 elseCondition()

virtual ScriptWriter & agvc_interface::ScriptWriter::elseCondition () [pure virtual]

12.36.3.4 elseIfCondition()

virtual ScriptWriter & agvc_interface::ScriptWriter::elseIfCondition () [pure virtual]

12.36.3.5 end()

virtual ScriptWriter & agvc_interface::ScriptWriter::end () [pure virtual]

12.36.3.6 ifCondition()

virtual ScriptWriter & agvc_interface::ScriptWriter::ifCondition () [pure virtual]

12.36.3.7 moveJoint()

virtual ScriptWriter & agvc_interface::ScriptWriter::moveJoint () [pure virtual]

12.36.3.8 moveLine()

virtual ScriptWriter & agvc_interface::ScriptWriter::moveLine () [pure virtual]

12.36.3.9 whileCondition()

virtual ScriptWriter & agvc_interface::ScriptWriter::whileCondition () [pure virtual]

The documentation for this class was generated from the following file:

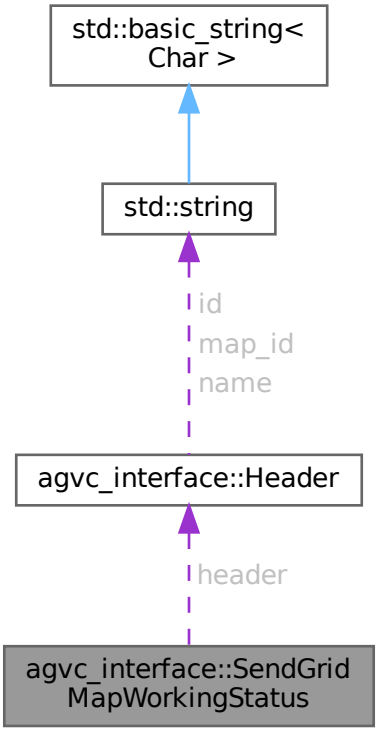
- include/agvc_interface/script.h

12.37 agvc_interface::SendGridMapWorkingStatus Struct Reference

sendGridMapToAgv

#include <type.h>

Collaboration diagram for agvc_interface::SendGridMapWorkingStatus:



Public Attributes

- Header header
  header.id sendGridMapToAgv id
- SendGridMapStatus status
  sendGridMapToAgv NONE - RUNNING- FINISH-

12.37.1 Detailed Description

sendGridMapToAgv

Definition at line 552 of file type.h.

12.37.2 Member Data Documentation

12.37.2.1 header

Header agvc_interface::SendGridMapWorkingStatus::header

header.id sendGridMapToAgv id

Definition at line 554 of file type.h.

12.37.2.2 status

SendGridMapStatus agvc_interface::SendGridMapWorkingStatus::status

sendGridMapToAgv NONE - RUNNING- FINISH-

Definition at line 555 of file type.h.

The documentation for this struct was generated from the following file:

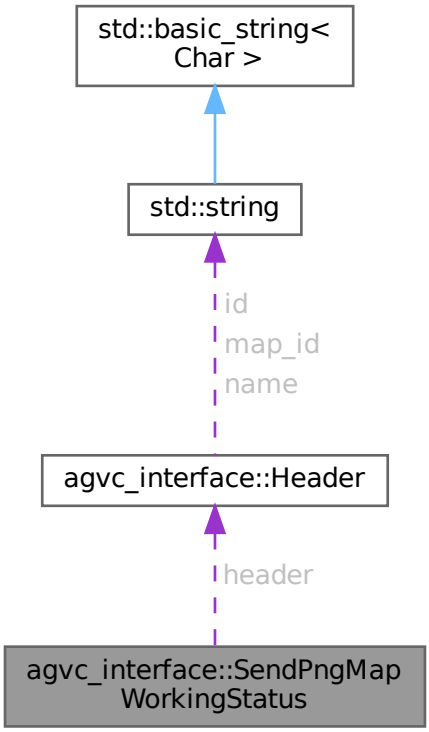
- include/agvc_interface/type.h

12.38 agvc_interface::SendPngMapWorkingStatus Struct Reference

sendBase64PngMapToAgv

#include <type.h>

Collaboration diagram for agvc_interface::SendPngMapWorkingStatus:



Public Attributes

- Header header
  header.id sendBase64PngMapToAgv id
- SendPngMapStatus status
  sendBase64PngMapToAgv NONE - RUNNING- FINISH-

12.38.1 Detailed Description

sendBase64PngMapToAgv

Definition at line 567 of file type.h.

12.38.2 Member Data Documentation

12.38.2.1 header

Header agvc_interface::SendPngMapWorkingStatus::header

header.id sendBase64PngMapToAgv id

Definition at line 569 of file type.h.

12.38.2.2 status

SendPngMapStatus agvc_interface::SendPngMapWorkingStatus::status

sendBase64PngMapToAgv NONE - RUNNING- FINISH-

Definition at line 570 of file type.h.

The documentation for this struct was generated from the following file:

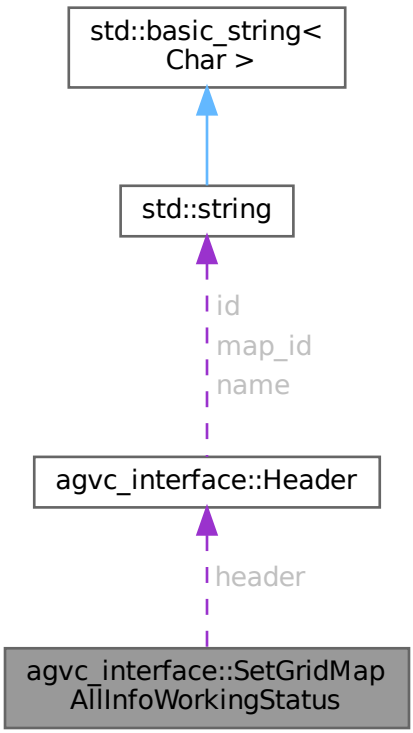
- include/agvc_interface/type.h

12.39 agvc_interface::SetGridMapAllInfoWorkingStatus Struct Reference

setGridMapAllInfo

#include <type.h>

Collaboration diagram for agvc_interface::SetGridMapAllInfoWorkingStatus:



Public Attributes

- Header header
  header.id setGridMapAllInfo id
- SetGridMapAllInfoStatus status
  setGridMapAllInfo NONE - RUNNING- FINISH-

12.39.1 Detailed Description

setGridMapAllInfo

Definition at line 576 of file type.h.

12.39.2 Member Data Documentation

12.39.2.1 header

Header agvc_interface::SetGridMapAllInfoWorkingStatus::header

header.id setGridMapAllInfo id

Definition at line 578 of file type.h.

12.39.2.2 status

SetGridMapAllInfoStatus agvc_interface::SetGridMapAllInfoWorkingStatus::status

setGridMapAllInfo NONE - RUNNING- FINISH-

Definition at line 579 of file type.h.

The documentation for this struct was generated from the following file:

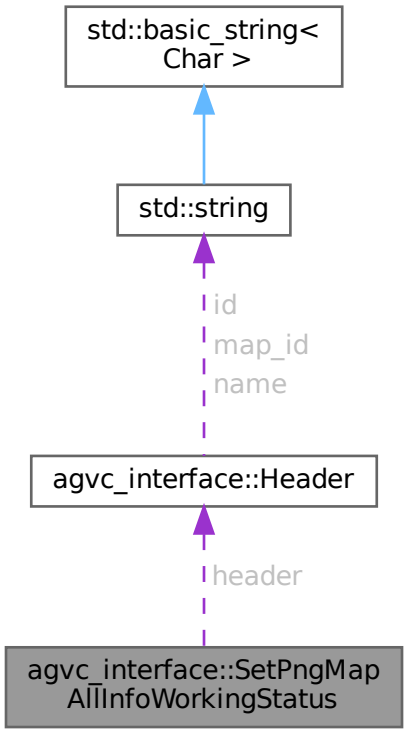
- include/agvc_interface/type.h

12.40 agvc_interface::SetPngMapAllInfoWorkingStatus Struct Reference

setPngMapAllInfo

#include <type.h>

Collaboration diagram for agvc_interface::SetPngMapAllInfoWorkingStatus:



Public Attributes

- Header header
  header.id setPngMapAllInfo id
- SetPngMapAllInfoStatus status
  setPngMapAllInfo NONE - RUNNING- FINISH-

12.40.1 Detailed Description

setPngMapAllInfo

Definition at line 525 of file type.h.

12.40.2 Member Data Documentation

12.40.2.1 header

Header agvc_interface::SetPngMapAllInfoWorkingStatus::header

header.id setPngMapAllInfo id

Definition at line 527 of file type.h.

12.40.2.2 status

SetPngMapAllInfoStatus agvc_interface::SetPngMapAllInfoWorkingStatus::status

setPngMapAllInfo NONE - RUNNING- FINISH-

Definition at line 528 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.41 agvc_interface::Speed Struct Reference

#include <type.h>

Public Attributes

- double v
  m/s
- double w
  rad/s

12.41.1 Detailed Description

Definition at line 416 of file type.h.

12.41.2 Member Data Documentation

12.41.2.1 v

double agvc_interface::Speed::v

m/s

Definition at line 418 of file type.h.

12.41.2.2 w

double agvc_interface::Speed::w

rad/s

Definition at line 419 of file type.h.

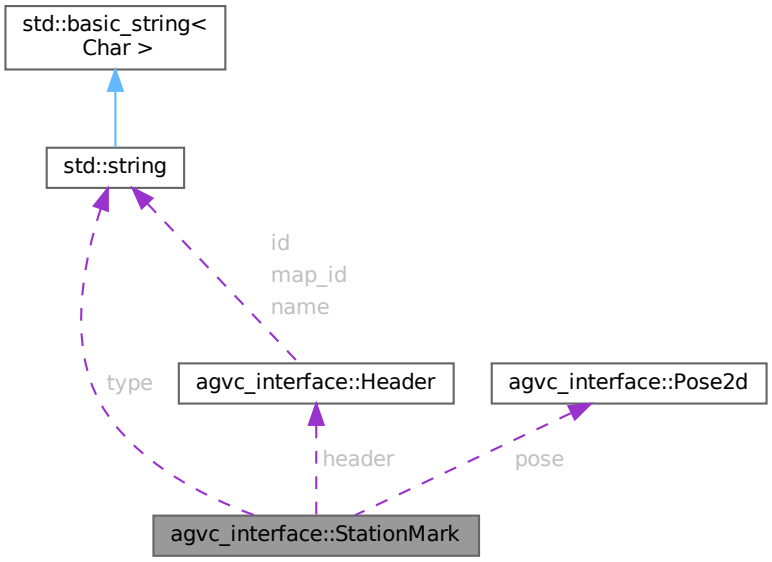
The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

12.42 agvc_interface::StationMark Struct Reference

#include <type.h>

Collaboration diagram for agvc_interface::StationMark:



Public Attributes

- Header header
  id id
- std::string type
  "ST", "CP"
- Pose2d pose
  {x(m) y(m) theta(rad)}

12.42.1 Detailed Description

Definition at line 641 of file type.h.

12.42.2 Member Data Documentation

12.42.2.1 header

Header agvc_interface::StationMark::header

id id

Definition at line 643 of file type.h.

12.42.2.2 pose

Pose2d agvc_interface::StationMark::pose

{x(m) y(m) theta(rad)}

Definition at line 645 of file type.h.

12.42.2.3 type

std::string agvc_interface::StationMark::type

"ST", "CP"

Definition at line 644 of file type.h.

The documentation for this struct was generated from the following file:

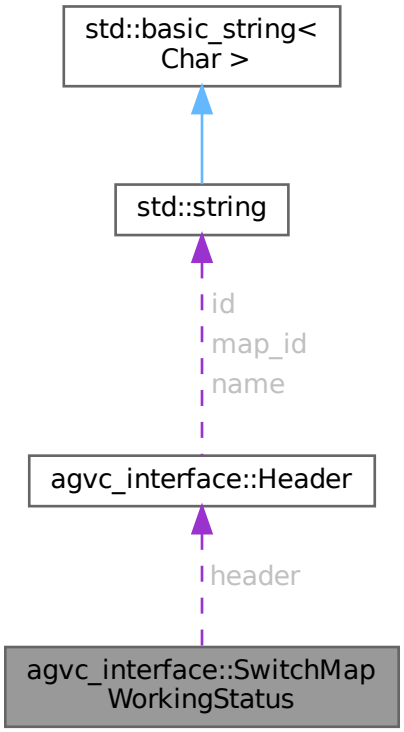
- include/agvc_interface/type.h

12.43 agvc_interface::SwitchMapWorkingStatus Struct Reference

switchMap

#include <type.h>

Collaboration diagram for agvc_interface::SwitchMapWorkingStatus:



Public Attributes

- Header header
  header.id switchMap id
- SwitchMapStatus status
  switchMap NONE - RUNNING- FINISH-

12.43.1 Detailed Description

switchMap

Definition at line 497 of file type.h.

12.43.2 Member Data Documentation

12.43.2.1 header

Header agvc_interface::SwitchMapWorkingStatus::header

header.id switchMap id

Definition at line 499 of file type.h.

12.43.2.2 status

SwitchMapStatus agvc_interface::SwitchMapWorkingStatus::status

switchMap NONE - RUNNING- FINISH-

Definition at line 500 of file type.h.

The documentation for this struct was generated from the following file:

- include/agvc_interface/type.h

Chapter 13

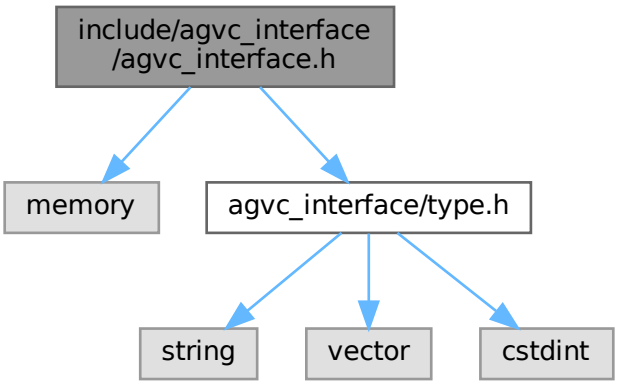
File Documentation

13.1 doc/deprecated_en.md File Reference

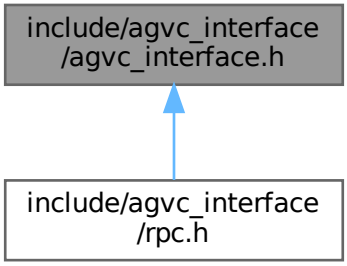
13.2 doc/SDK_overview_en.md File Reference

13.3 include/agvc_interface/agvc_interface.h File Reference

#include <memory>
#include "agvc_interface/type.h"
Include dependency graph for agvc_interface.h:



This graph shows which files directly or indirectly include this file:



Classes

- class agvc_interface::AgvcInterface

Namespaces

- namespace agvc_interface

Macros

- #define INTERFACE_VERSION_MAJOR 0
- #define INTERFACE_VERSION_MINOR 4
- #define INTERFACE_VERSION_PATCH 0
- #define INTERFACE_VERSION (INTERFACE_VERSION_MAJOR * 1000000 + INTERFACE_VERSION_MI
  * 1000 + INTERFACE_VERSION_PATCH)
- #define AgvcInterface_DECLARES

Typedefs

- using agvc_interface::AgvcInterfacePtr = std::shared_ptr<AgvcInterface>

13.3.1 Macro Definition Documentation

13.3.1.1 AgvcInterface_DECLARES

#define AgvcInterface_DECLARES

Definition at line 1496 of file agvc_interface.h.

13.3.1.2 INTERFACE_VERSION

#define INTERFACE_VERSION (INTERFACE_VERSION_MAJOR * 1000000 + INTERFACE_VERSION_MINOR * 1000 + INTERFACE_VERSION_PATCH)

Definition at line 10 of file agvc_interface.h.

13.3.1.3 INTERFACE_VERSION_MAJOR

#define INTERFACE_VERSION_MAJOR 0

Definition at line 7 of file agvc_interface.h.

13.3.1.4 INTERFACE_VERSION_MINOR

#define INTERFACE_VERSION_MINOR 4

Definition at line 8 of file agvc_interface.h.

13.3.1.5 INTERFACE_VERSION_PATCH

#define INTERFACE_VERSION_PATCH 0

Definition at line 9 of file agvc_interface.h.

13.4 agvc_interface.h

Go to the documentation of this file.

```
00001 #ifndef AGVC_INTERFACE_H
00002 #define AGVC_INTERFACE_H
00003
00004 #include <memory>
00005 #include "agvc_interface/type.h"
00006
00007 #define INTERFACE_VERSION_MAJOR 0
00008 #define INTERFACE_VERSION_MINOR 4
00009 #define INTERFACE_VERSION_PATCH 0
00010 #define INTERFACE_VERSION       (INTERFACE_VERSION_MAJOR * 1000000 +
      INTERFACE_VERSION_MINOR * 1000 + INTERFACE_VERSION_PATCH)
00011
00012 /**
00013  * \chinese
00014  * @defgroup Map Map (   )
00015  * \endchinese
00016  * \english
00017  * @defgroup Map Map
00018  * \endenglish
00019  * @brief
00020  * @zh AGVC
00021  * @en Core functions for creating, switching, deleting, reading, and writing AGVC custom maps.
00022  */
00023
00024 /**
00025  * \chinese
00026  * @defgroup Station Station (   )
00027  * \endchinese
00028  * \english
00029  * @defgroup Station Station
00030  * \endenglish
00031  * @brief
00032  * @zh AGVC
00033  * @en Functions for adding, deleting, modifying, querying AGVC map stations, and auto-detecting charging stations.
```

```
00034  */
00035
00036 /**
00037  * \chinese
00038  * @defgroup Path Path (   )
00039  * \endchinese
00040  * \english
00041  * @defgroup Path Path
00042  * \endenglish
00043  * @brief
00044  * @zh AGVC
00045  * @en Functions for generating, modifying, deleting, and querying AGVC map paths.
00046  */
00047
00048 /**
00049  * \chinese
00050  * @defgroup Navigation Navigation (   )
00051  * \endchinese
00052  * \english
00053  * @defgroup Navigation Navigation
00054  * \endenglish
00055  * @brief
00056  * @zh AGVC
00057  * @en Functions for AGVC motion control, navigation goals, relocation, and mode switching.
00058  */
00059
00060 /**
00061  * \chinese
00062  * @defgroup Charging Charging (   )
00063  * \endchinese
00064  * \english
00065  * @defgroup Charging Charging
00066  * \endenglish
00067  * @brief
00068  * @zh AGVC
00069  * @en AGVC charging functions, including auto charging, forced docking or undocking, and battery checks.
00070  */
00071
00072 /**
00073  * \chinese
00074  * @defgroup AgvInfo AgvInfo (AGV   )
00075  * \endchinese
00076  * \english
00077  * @defgroup AgvInfo AgvInfo
00078  * \endenglish
00079  * @brief
00080  * @zh AGVC
00081  * @en Query functions for AGVC status, pose, logs, error codes, and related information.
00082  */
00083
00084 /**
00085  * \chinese
00086  * @defgroup System System (   )
00087  * \endchinese
00088  * \english
00089  * @defgroup System System
00090  * \endenglish
00091  * @brief
00092  * @zh AGVC
00093  * @en System-level functions for AGVC configuration, control priority, software upgrades, and firmware updates.
00094  */
00095
00096 namespace agvc_interface {
00097
00098 class AgvcInterface
00099 {
00100 public:
00101     AgvcInterface();
00102     virtual ~AgvcInterface();
00103
00104     /**
00105      * @ingroup System
00106      * @brief
00107      * @zh   AGV
00108      * @en Sets the AGV system time.
00109      * @param[in] stamp
00110      * @zh       YYYY-MM-DDTHH:mm:ss.ssZ   “2017-04-15T11:40:03.12Z”
00111      * @en Timestamp in YYYY-MM-DDTHH:mm:ss.ssZ format, for example “2017-04-15T11:40:03.12Z”.
00112      */
00113     int setSystemClock(const std::string &stamp);
00114
00115     /**
00116      * @ingroup AgvInfo
00117      * @attention
00118      * @zh RTDE
00119      * @en RTDE pushed data.
00120      * @brief
```

```
00121     * @zh    AGV
00122     * @en Queries current AGV information.
00123     * @return
00124     * @zh AGV
00125     * @en Detailed AGV information.
00126     */
00127    AgvDetails getAgvDetails();
00128
00129    /**
00130     * @ingroup AgvInfo
00131     * @attention
00132     * @zh RTDE
00133     * @en RTDE pushed data.
00134     * @brief
00135     * @zh    AGV
00136     * @en Queries current AGV running information.
00137     * @return
00138     * @zh AGV
00139     * @en AGV running information.
00140     */
00141    RunningInfo getRunningInfo();
00142
00143    /**
00144     * @ingroup AgvInfo
00145     * @attention
00146     * @zh RTDE
00147     * @en RTDE pushed data.
00148     * @brief
00149     * @zh   AGV
00150     * @en Queries the current AGV pose.
00151     * @return
00152     * @zh
00153     * @en Current pose.
00154     */
00155    Pose2d getAgvCurrentPose();
00156
00157    /**
00158     * @ingroup AgvInfo
00159     * @attention
00160     * @zh RTDE
00161     * @en RTDE pushed data.
00162     * @brief
00163     * @zh
00164     * @en Queries the current laser point cloud data.
00165     * @return
00166     * @zh
00167     * @en 2D laser point coordinates.
00168     */
00169    std::vector<Point2d> getLaserPointCloud();
00170
00171    /**
00172     * @ingroup System
00173     * @brief
00174     * @zh
00175     * @en Queries the running status of asynchronous APIs; this does not represent each API's own execution result.
00176     * @note
00177     * @zh     saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv
00178     * @en Asynchronous APIs: saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv.
00179     * @note
00180     * @zh     getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv
00181     * @en Asynchronous APIs: getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv.
00182     * @note
00183     * @zh     getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, previewPngMapFromAgv
00184     * @en Asynchronous APIs: getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv,
        previewPngMapFromAgv.
00185     * @note
00186     * @zh     autoAlignRailway
00187     * @en Asynchronous API: autoAlignRailway.
00188     * @note
00189     * @zh    AGV
00190     * @en After the client disconnects from the AGV, call this API to obtain asynchronous API status.
00191     * @note
00192     * @zh        header  header.id      id
00193     * @en Each field in the return value contains a header, where header.id is the ID used when issuing the asynchronous
        task.
00194     * @return
00195     * @zh NONE      RUNNING      FINISH
00196     * @en NONE: asynchronous API is not running; RUNNING: asynchronous API is running; FINISH: asynchronous API
        has finished.
00197     */
00198    AsyncInterfaceResultStatus getAsyncInterfaceResultStatus();
00199
00200    /**
00201     * @ingroup Station
00202     * @brief
00203     * @zh
00204     * @en Queries all station information on the current map.
```

```
00205     * @return
00206     * @zh
00207     * @en All station information on the current map.
00208     */
00209    std::vector<StationMark> getAllStations();
00210
00211    /**
00212     * @ingroup Station
00213     * @brief
00214     * @zh
00215     * @en Queries all station information on the specified map.
00216     * @param[in] map_header
00217     * @zh    header( id      id)
00218     * @en Header of the specified map (command ID, name, time, map ID).
00219     * @return
00220     * @zh
00221     * @en All station information on the specified map.
00222     */
00223    std::vector<StationMark> getAllStationsOfTargetMap(const Header &map_header);
00224
00225    /**
00226     * @ingroup Station
00227     * @brief
00228     * @zh
00229     * @en Adds or modifies one station.
00230     * @param[in] station
00231     * @zh
00232     * @en Station information to add or modify.
00233     * @return
00234     * @zh 10100000       else
00235     * @en 10100000: station added or modified successfully; else: failed to add or modify the station.
00236     */
00237    int addStation(const StationMark &station);
00238
00239    /**
00240     * @ingroup Station
00241     * @brief
00242     * @zh
00243     * @en Adds or modifies multiple stations.
00244     * @param[in] stations
00245     * @zh
00246     * @en Station information to add or modify.
00247     * @return
00248     * @zh 10100000       else
00249     * @en 10100000: stations added or modified successfully; else: failed to add or modify stations.
00250     */
00251    int addStations(const std::vector<StationMark> &stations);
00252
00253    /**
00254     * @ingroup Station
00255     * @brief
00256     * @zh
00257     * @en Adds a charging station by automatically recognizing the charging board pose.
00258     * @since 0.7.0
00259     * @param[in] station_header
00260     * @zh          ( id      id)
00261     * @en Charging station information to add or modify (station ID, name, time, map ID).
00262     * @param[in] dis_station_board
00263     * @zh          (    AGV  )   m   [1.0~2.0]
00264     * @en Distance between the charging station and charging board. The charging station is at the AGV center. Unit: m;
        range: [1.0, 2.0].
00265     * @return
00266     * @zh 10100000       else
00267     * @en 10100000: charging station added successfully; else: failed to add the charging station.
00268     */
00269    int addCPStationUseChargingBoard(const Header &station_header, const double &dis_station_board = 1.5);
00270
00271    /**
00272     * @ingroup Station
00273     * @brief
00274     * @zh
00275     * @en Deletes the specified station and its related paths.
00276     * @param[in] station_header
00277     * @zh      ( id      id)
00278     * @en Station information to delete (station ID, name, time, map ID).
00279     * @return
00280     * @zh 10100000      else
00281     * @en 10100000: station deleted successfully; else: failed to delete the station.
00282     */
00283    int deleteStation(const Header &station_header);
00284
00285    /**
00286     * @ingroup Station
00287     * @brief
00288     * @zh
00289     * @en Deletes multiple stations and their related paths.
00290     * @param[in] stations_header
```

```
00291     * @zh      ( id       id)
00292     * @en Station information to delete (station ID, name, time, map ID).
00293     * @return
00294     * @zh 10100000      else
00295     * @en 10100000: stations deleted successfully; else: failed to delete stations.
00296     */
00297    int deleteStations(const std::vector<Header> &stations_header);
00298
00299    /**
00300     * @ingroup Path
00301     * @brief
00302     * @zh
00303     * @en Queries all path information on the current map.
00304     * @return
00305     * @zh
00306     * @en All path information on the current map.
00307     */
00308    std::vector<PathStation> getAllPaths();
00309
00310    /**
00311     * @ingroup Path
00312     * @brief
00313     * @zh
00314     * @en Queries all path information on the specified map.
00315     * @param[in] map_header
00316     * @zh    header( id       id)
00317     * @en Header of the specified map (command ID, name, time, map ID).
00318     * @return
00319     * @zh
00320     * @en All path information on the specified map.
00321     */
00322    std::vector<PathStation> getAllPathsOfTargetMap(const Header &map_header);
00323
00324    /**
00325     * @ingroup Path
00326     * @brief
00327     * @zh   AGV
00328     * @en Queries the path currently being tracked by the AGV.
00329     * @return
00330     * @zh
00331     * @en Information about the path currently being tracked.
00332     */
00333    PathStation getCurrentPath();
00334
00335    /**
00336     * @ingroup Path
00337     * @brief
00338     * @zh
00339     * @en Generates or modifies one path.
00340     * @param[in] path_station
00341     * @zh
00342     * @en Path information to generate or modify.
00343     * @return
00344     * @zh 10100000        else
00345     * @en 10100000: path modified or generated successfully; else: failed to modify or generate the path.
00346     */
00347    int generatePath(const PathStation &path_station);
00348
00349    /**
00350     * @ingroup Path
00351     * @brief
00352     * @zh
00353     * @en Generates or modifies multiple paths.
00354     * @param[in] paths_station
00355     * @zh
00356     * @en Path information to generate or modify.
00357     * @return
00358     * @zh 10100000        else
00359     * @en 10100000: paths modified or generated successfully; else: failed to modify or generate paths.
00360     */
00361    int generatePaths(const std::vector<PathStation> &paths_station);
00362
00363    /**
00364     * @ingroup Path
00365     * @brief
00366     * @zh
00367     * @en Deletes one path.
00368     * @param[in] path_header
00369     * @zh      ( id       id)
00370     * @en Path information to delete (path ID, name, time, map ID).
00371     * @return
00372     * @zh 10100000      else
00373     * @en 10100000: path deleted successfully; else: failed to delete the path.
00374     */
00375    int deletePath(const Header &path_header);
00376
00377    /**
```

```
00378     * @ingroup Path
00379     * @brief
00380     * @zh
00381     * @en Deletes multiple paths.
00382     * @param[in] paths_header
00383     * @zh      ( id       id)
00384     * @en Path information to delete (path ID, name, time, map ID).
00385     * @return
00386     * @zh 10100000      else
00387     * @en 10100000: paths deleted successfully; else: failed to delete paths.
00388     */
00389    int deletePaths(const std::vector<Header> &paths_header);
00390
00391    /**
00392     * @ingroup Map
00393     * @brief
00394     * @zh   AGV
00395     * @en Gets the headers of all maps on the AGV.
00396     * @return
00397     * @zh       ( id       id)
00398     * @en Headers of all maps (command ID, name, time, map ID).
00399     */
00400    std::vector<Header> getMapList();
00401
00402    /**
00403     * @ingroup Map
00404     * @brief
00405     * @zh    AGV
00406     * @en Queries the header of the current AGV map.
00407     * @return
00408     * @zh       ( id       id)
00409     * @en Header of the current map (command ID, name, time, map ID).
00410     */
00411    Header getCurrentMapHeader();
00412
00413    /**
00414     * @ingroup Map
00415     * @brief
00416     * @zh
00417     * @en Gets the specified occupancy grid map information.
00418     * @param[in] map_header
00419     * @zh       (     id       id)
00420     * @en Header of the specified map (this asynchronous command ID, name, time, map ID).
00421     * @note
00422     * @zh map_id "current_map"
00423     * @en When map_id is "current_map", it refers to the current map.
00424     * @return
00425     * @zh
00426     * @en Specified occupancy grid map information.
00427     */
00428    OccupancyGridMap getGridMapFromAgv(const Header &map_header);
00429
00430    /**
00431     * @ingroup Map
00432     * @attention
00433     * @zh 1.
00434     * @en 1. Asynchronous API.
00435     * @brief
00436     * @zh   Base64   PNG
00437     * @en Gets Base64-encoded PNG map information.
00438     * @param[in] map_header
00439     * @zh       (     id       id)
00440     * @en Header of the specified map (this asynchronous command ID, name, time, map ID).
00441     * @note
00442     * @zh map_id "current_map"
00443     * @en When map_id is "current_map", it refers to the current map.
00444     * @return
00445     * @zh Base64   PNG
00446     * @en Base64-encoded PNG map information.
00447     */
00448    Base64PngMap getBase64PngMapFromAgv(const Header &map_header);
00449
00450    /**
00451     * @ingroup Map
00452     * @attention
00453     * @zh 1.
00454     * @en 1. Asynchronous API.
00455     * @brief
00456     * @zh
00457     * @en Gets a preview image with optional width and height.
00458     * @param[in] map_header
00459     * @zh       (     id       id)
00460     * @en Header of the specified map (this asynchronous command ID, name, time, map ID).
00461     * @param[in] image_width_px
00462     * @zh    ( )    0
00463     * @en Map width in pixels. If the default value 0 is used, returned thumbnail data is empty.
00464     * @param[in] image_height_px
```

```
00465     * @zh  ( )  0
00466     * @en Map height in pixels. If the default value 0 is used, returned thumbnail data is empty.
00467     * @note
00468     * @zh map_id "current_map"
00469     * @en When map_id is "current_map", it refers to the current map.
00470     * @return
00471     * @zh Base64 PNG
00472     * @en Base64-encoded PNG thumbnail.
00473     */
00474    Base64PngMap previewPngMapFromAgv(const Header &map_header, const int &image_width_px = 0, const int
      &image_height_px = 0);
00475
00476    /**
00477     * @ingroup Map
00478     * @attention
00479     * @zh 1.
00480     * @en 1. Asynchronous API.
00481     * @brief
00482     * @zh  AGV
00483     * @en Sends occupancy grid map information to the AGV.
00484     * @param[in] map
00485     * @zh
00486     * @en Occupancy grid map information.
00487     * @return
00488     * @zh 10100000  10100201  else
00489     * @en 10100000: map sent successfully; 10100201: map is being sent; else: failed to send the map.
00490     */
00491    int sendGridMapToAgv(const OccupancyGridMap &map);
00492
00493    /**
00494     * @ingroup Map
00495     * @attention
00496     * @zh 1.
00497     * @en 1. Asynchronous API.
00498     * @brief
00499     * @zh Base64 PNG  AGV
00500     * @en Sends a Base64-encoded PNG map to the AGV.
00501     * @param[in] map
00502     * @zh Base64 PNG
00503     * @en Base64-encoded PNG map.
00504     * @return
00505     * @zh 10100000  10100201  else
00506     * @en 10100000: map sent successfully; 10100201: map is being sent; else: failed to send the map.
00507     */
00508    int sendBase64PngMapToAgv(const Base64PngMap &map);
00509
00510    /**
00511     * @ingroup Map
00512     * @attention
00513     * @zh 1.  2.
00514     * @en 1. Asynchronous API; 2. Requires control priority.
00515     * @brief
00516     * @zh
00517     * @en Saves the map after mapping is complete.
00518     * @param[in] map_header
00519     * @zh  ( id  id)
00520     * @en Map header (this asynchronous command ID, name, time, map ID).
00521     * @return
00522     * @zh 10100000  10100201  10120202  else
00523     * @en 10100000: map saved successfully; 10100201: asynchronous API is running; 10120202: no control priority; else:
      failed to save the map.
00524     */
00525    int saveMap(const Header &map_header);
00526
00527    /**
00528     * @ingroup Map
00529     * @attention
00530     * @zh 1.  2.
00531     * @en 1. Asynchronous API; 2. Requires control priority.
00532     * @brief
00533     * @zh
00534     * @en Switches to the specified map.
00535     * @param[in] map_header
00536     * @zh  ( id  id)
00537     * @en Header of the specified map (this asynchronous command ID, name, time, map ID).
00538     * @return
00539     * @zh 10100000  10100201  10120202  else
00540     * @en 10100000: map switched successfully; 10100201: asynchronous API is running; 10120202: no control priority; else:
      failed to switch the map.
00541     */
00542    int switchMap(const Header &map_header);
00543
00544    /**
00545     * @ingroup Map
00546     * @brief
00547     * @zh
00548     * @en Deletes one specified map.
```

```
00549     * @param[in] map_header
00550     * @zh       ( id       id)
00551     * @en Header of the specified map (command ID, name, time, map ID).
00552     * @return
00553     * @zh 10100000       else
00554     * @en 10100000: map deleted successfully; else: failed to delete the map.
00555     */
00556    int deleteMap(const Header &map_header);
00557
00558    /**
00559     * @ingroup Map
00560     * @brief
00561     * @zh
00562     * @en Deletes multiple specified maps.
00563     * @param[in] map_headers
00564     * @zh         ( id       id)
00565     * @en Headers of the specified maps (command ID, name, time, map ID).
00566     * @return
00567     * @zh 10100000       else
00568     * @en 10100000: maps deleted successfully; else: failed to delete maps.
00569     */
00570    int deleteMaps(const std::vector<Header> &map_headers);
00571
00572    /**
00573     * @ingroup Map
00574     * @brief
00575     * @zh
00576     * @en Queries virtual areas on the current map.
00577     * @return
00578     * @zh
00579     * @en Virtual area information on the current map.
00580     */
00581    std::vector<MapVirtualArea> getAllMapVirtualArea();
00582
00583    /**
00584     * @ingroup Map
00585     * @brief
00586     * @zh
00587     * @en Queries virtual areas on the target map.
00588     * @param[in] map_header
00589     * @zh    header( id       id)
00590     * @en Header of the specified map (command ID, name, time, map ID).
00591     * @return
00592     * @zh
00593     * @en All virtual areas on the specified map.
00594     */
00595    std::vector<MapVirtualArea> getAllMapVirtualAreaOfTargetMap(const Header &map_header);
00596
00597    /**
00598     * @ingroup Map
00599     * @brief
00600     * @zh
00601     * @en Adds or modifies one virtual area.
00602     * @param[in] map_virtual_area
00603     * @zh
00604     * @en Virtual area information to add or modify.
00605     * @return
00606     * @zh 10100000          else
00607     * @en 10100000: virtual area added or modified successfully; else: failed to add or modify the virtual area.
00608     */
00609    int addMapVirtualArea(const MapVirtualArea &map_virtual_area);
00610
00611    /**
00612     * @ingroup Map
00613     * @brief
00614     * @zh
00615     * @en Adds or modifies multiple virtual areas.
00616     * @param[in] map_virtual_areas
00617     * @zh
00618     * @en Virtual area information to add or modify.
00619     * @return
00620     * @zh 10100000          else
00621     * @en 10100000: virtual areas added or modified successfully; else: failed to add or modify virtual areas.
00622     */
00623    int addMapVirtualAreas(const std::vector<MapVirtualArea> &map_virtual_areas);
00624
00625    /**
00626     * @ingroup Map
00627     * @brief
00628     * @zh
00629     * @en Deletes the specified virtual area.
00630     * @param[in] virtual_area_header
00631     * @zh       ( id           id)
00632     * @en Header of the specified virtual area (command ID, virtual area name, time, map ID).
00633     * @return
00634     * @zh 10100000       else
00635     * @en 10100000: virtual area deleted successfully; else: failed to delete the virtual area.
```

```
00636     */
00637     int deleteMapVirtualArea(const Header &virtual_area_header);
00638
00639     /**
00640      * @ingroup Map
00641      * @brief
00642      * @zh
00643      * @en Deletes multiple specified virtual areas.
00644      * @param[in] virtual_areas_header
00645      * @zh        ( id           id)
00646      * @en Headers of the specified virtual areas (command ID, virtual area name, time, map ID).
00647      * @return
00648      * @zh 10100000        else
00649      * @en 10100000: virtual areas deleted successfully; else: failed to delete virtual areas.
00650      */
00651     int deleteMapVirtualAreas(const std::vector<Header> &virtual_areas_header);
00652
00653     /**
00654      * @ingroup Map
00655      * @attention
00656      * @zh 1.
00657      * @en 1. Asynchronous API.
00658      * @brief
00659      * @zh
00660      * @en Gets occupancy grid map data, paths, stations, and virtual areas for the specified map.
00661      * @param[in] map_header
00662      * @zh       (     id         id)
00663      * @en Header of the specified map (this asynchronous command ID, map name, time, map ID).
00664      * @note
00665      * @zh map_id "current_map"
00666      * @en When map_id is "current_map", it refers to the current map.
00667      * @return
00668      * @zh
00669      * @en All information on the occupancy grid map.
00670      */
00671     MapAllInfo getGridMapAllInfo(const Header &map_header);
00672
00673     /**
00674      * @ingroup Map
00675      * @attention
00676      * @zh 1.
00677      * @en 1. Asynchronous API.
00678      * @brief
00679      * @zh       Base64
00680      * @en Gets Base64 map data, paths, stations, and virtual areas for the specified map.
00681      * @param[in] map_header
00682      * @zh       (     id         id)
00683      * @en Header of the specified map (this asynchronous command ID, map name, time, map ID).
00684      * @note
00685      * @zh map_id "current_map"
00686      * @en When map_id is "current_map", it refers to the current map.
00687      * @return
00688      * @zh Base64
00689      * @en All information on the Base64 map.
00690      */
00691     MapAllInfo getPngMapAllInfo(const Header &map_header);
00692
00693     /**
00694      * @ingroup Map
00695      * @attention
00696      * @zh 1.    2.
00697      * @en 1. Asynchronous API; 2. Requires control priority.
00698      * @brief
00699      * @zh
00700      * @en Sets occupancy grid map, path, station, and virtual area information.
00701      * @param[in] map_all_info
00702      * @zh
00703      * @en All information on the occupancy grid map.
00704      * @param[in] command_header
00705      * @zh id         id
00706      * @en The id field is the ID of this command; other fields have no special meaning.
00707      * @note
00708      * @zh          AGV
00709      * @en Ensure that the data can be applied; the AGV does not validate the data.
00710      * @return
00711      * @zh 10100000         10100201       10120202     else
00712      * @en 10100000: all map information uploaded successfully; 10100201: asynchronous API is running; 10120202: no
    control priority; else: failed to upload all map information.
00713      */
00714     int setGridMapAllInfo(const MapAllInfo &map_all_info,
00715                           const Header &command_header = { "99999", "99999", 1, "99999" });
00716
00717     /**
00718      * @ingroup Map
00719      * @attention
00720      * @zh 1.    2.
00721      * @en 1. Asynchronous API; 2. Requires control priority.
```

```
00722     * @brief
00723     * @zh  Base64
00724     * @en Sets Base64 map, path, station, and virtual area information.
00725     * @param[in] map_all_info
00726     * @zh Base64
00727     * @en All information on the Base64 map.
00728     * @param[in] command_header
00729     * @zh id       id
00730     * @en The id field is the ID of this command; other fields have no special meaning.
00731     * @note
00732     * @zh         AGV
00733     * @en Ensure that the data can be applied; the AGV does not validate the data.
00734     * @return
00735     * @zh 10100000       10100201      10120202    else
00736     * @en 10100000: all map information uploaded successfully; 10100201: asynchronous API is running; 10120202: no
    control priority; else: failed to upload all map information.
00737     */
00738    int setPngMapAllInfo(const MapAllInfo &map_all_info,
00739                         const Header &command_header = { "99999", "99999", 1, "99999" });
00740
00741    /**
00742     * @ingroup Map
00743     * @brief
00744     * @zh
00745     * @en Deletes all information of the specified map, including map data, stations, paths, and virtual areas.
00746     * @param[in] map_header
00747     * @zh      ( id       id)
00748     * @en Header of the specified map (command ID, map name, time, map ID).
00749     * @return
00750     * @zh 10100000    else
00751     * @en 10100000: deleted successfully; else: deletion failed.
00752     */
00753    int deleteMapAllInfo(const Header &map_header);
00754
00755    /**
00756     * @ingroup System
00757     * @attention
00758     * @zh
00759     * @en Asynchronous API.
00760     * @brief
00761     * @zh      WiFi
00762     * @en Gets the currently scanned WiFi list.
00763     * @return
00764     * @zh {RUNNING}      {}      WIFI {SSID}      WIFI
00765     * @en {RUNNING}: asynchronous API is running; {}: empty, no WiFi found; {SSID}: non-empty list of scanned WiFi
    SSIDs.
00766     */
00767    std::vector<std::string> getWifiList();
00768
00769    /**
00770     * @ingroup System
00771     * @brief
00772     * @zh    WiFi    WiFi
00773     * @en Connects to the specified WiFi and automatically switches to WiFi mode.
00774     * @note
00775     * @zh
00776     * @en Must be called over a wired connection and may take a long time.
00777     * @param[in] ssid
00778     * @zh WiFi
00779     * @en WiFi SSID.
00780     * @param[in] password
00781     * @zh WiFi
00782     * @en WiFi password.
00783     * @return
00784     * @zh 10100000  WiFi  else  WiFi
00785     * @en 10100000: connected to WiFi successfully; else: failed to connect to WiFi.
00786     */
00787    int connectWifi(const std::string &ssid, const std::string &password);
00788
00789    /**
00790     * @ingroup System
00791     * @brief
00792     * @zh
00793     * @en Enables hotspot mode and automatically switches to hotspot mode.
00794     * @note
00795     * @zh
00796     * @en Must be called over a wired connection and may take a long time.
00797     * @param[in] password
00798     * @zh        "administrator"    SSID  SN
00799     * @en Hotspot password. If empty, "administrator" is used as the default password, and the SSID defaults to the SN.
00800     * @return
00801     * @zh 10100000      else
00802     * @en 10100000: hotspot enabled successfully; else: failed to enable hotspot.
00803     */
00804    int enableHotspot(const std::string &password);
00805
00806    /**
```

```
00807     * @ingroup System
00808     * @brief
00809     * @zh      IP
00810     * @en Gets all local IP addresses.
00811     * @return
00812     * @zh    IP
00813     * @en All local IP addresses.
00814     */
00815    std::vector<IpAddressInfo> getIpAddressList();
00816
00817    /**
00818     * @ingroup Navigation
00819     * @attention
00820     * @zh 1.    2.
00821     * @en 1. Asynchronous API; 2. Requires control priority.
00822     * @brief
00823     * @zh   AGV
00824     * @en Switches the AGV running mode.
00825     * @param[in] running_mode
00826     * @zh
00827     * @en Target mode to switch to, such as navigation mode or mapping mode.
00828     * @param[in] command_header
00829     * @zh id         id
00830     * @en The id field is the ID of this command; other fields have no special meaning.
00831     * @return
00832     * @zh 10100000      10100201      10120202     else
00833     * @en 10100000: mode switched successfully; 10100201: asynchronous API is running; 10120202: no control priority;
    else: failed to switch mode.
00834     */
00835    int changeRunningMode(const RunningMode &running_mode,
00836                          const Header &command_header = { "99999", "99999", 1, "99999" });
00837
00838    /**
00839     * @ingroup Navigation
00840     * @attention
00841     * @zh
00842     * @en Requires control priority.
00843     * @brief
00844     * @zh   AGV
00845     * @en Controls AGV motion.
00846     * @param[in] speed
00847     * @zh
00848     * @en Speed information.
00849     * @return
00850     * @zh 10100000  AGV     10120202     else  AGV
00851     * @en 10100000: speed command sent to the AGV successfully; 10120202: no control priority; else: failed to send the
    speed command.
00852     */
00853    int setControlSpeed(const Speed &speed);
00854
00855    /**
00856     * @ingroup Navigation
00857     * @attention
00858     * @zh
00859     * @en Requires control priority.
00860     * @brief
00861     * @zh  AGV
00862     * @en Sends a navigation goal to the AGV.
00863     * @param[in] target
00864     * @zh
00865     * @en Target pose information and navigation parameters.
00866     * @return
00867     * @zh 10100000       10120202      else
00868     * @en 10100000: navigation goal set successfully; 10120202: no control priority; else: failed to set the navigation goal.
00869     */
00870    int setNavGoal(const NavGoalType &target);
00871
00872    /**
00873     * @ingroup AgvInfo
00874     * @attention
00875     * @zh RTDE
00876     * @en RTDE pushed data.
00877     * @brief
00878     * @zh
00879     * @en Gets current navigation information.
00880     * @return
00881     * @zh
00882     * @en Current navigation information.
00883     */
00884    NavInfo getNavInfo();
00885
00886    /**
00887     * @ingroup Navigation
00888     * @attention
00889     * @zh
00890     * @en Requires control priority.
00891     * @brief
```

```
00892     * @zh  AGV
00893     * @en Pauses AGV speed.
00894     * @return
00895     * @zh 10100000    10120202     else
00896     * @en 10100000: paused successfully; 10120202: no control priority; else: pause failed.
00897     */
00898    int pauseAgvSpeed();
00899
00900    /**
00901     * @ingroup Navigation
00902     * @attention
00903     * @zh
00904     * @en Requires control priority.
00905     * @brief
00906     * @zh     AGV
00907     * @en Resumes AGV speed after a pause.
00908     * @return
00909     * @zh 10100000    10120202     else
00910     * @en 10100000: resumed successfully; 10120202: no control priority; else: resume failed.
00911     */
00912    int resumeAgvSpeed();
00913
00914    /**
00915     * @ingroup Navigation
00916     * @attention
00917     * @zh
00918     * @en Requires control priority.
00919     * @brief
00920     * @zh
00921     * @en Cancels the navigation task.
00922     * @return
00923     * @zh 10100000       10120202     else
00924     * @en 10100000: navigation task canceled successfully; 10120202: no control priority; else: failed to cancel the
    navigation task.
00925     */
00926    int cancelNavigation();
00927
00928    /**
00929     * @ingroup Navigation
00930     * @attention
00931     * @zh 1.    2.
00932     * @en 1. Asynchronous API; 2. Requires control priority.
00933     * @brief
00934     * @zh AGV
00935     * @en Relocates the AGV.
00936     * @param[in] init_pose
00937     * @zh AGV
00938     * @en Current reference pose of the AGV.
00939     * @param[in] command_header
00940     * @zh id        id
00941     * @en The id field is the ID of this command; other fields have no special meaning.
00942     * @return
00943     * @zh 10100000     10100201      10120202     else
00944     * @en 10100000: relocation succeeded; 10100201: asynchronous API is running; 10120202: no control priority; else:
    relocation failed.
00945     */
00946    int relocation(const Pose2d &init_pose, const Header &command_header = { "99999", "99999", 1, "99999" });
00947
00948    /**
00949     * @ingroup System
00950     * @brief
00951     * @zh  AGV
00952     * @en Gets the AGV controller parameter file.
00953     * @return
00954     * @zh  AGV
00955     * @en Parameter information in the current AGV controller.
00956     */
00957    std::string getAgvControllerParametersFile();
00958
00959    /**
00960     * @ingroup System
00961     * @attention
00962     * @zh
00963     * @en Requires control priority.
00964     * @brief
00965     * @zh  AGV
00966     * @en Sets the AGV controller parameter file.
00967     * @note
00968     * @zh          1Hz
00969     * @en Do not call frequently. Keep the call rate below 1 Hz. This API refreshes parameters by default.
00970     * @param[in] agv_parameters
00971     * @zh AGV
00972     * @en Parameter information for the AGV controller.
00973     * @return
00974     * @zh 10100000      else
00975     * @en 10100000: parameters set successfully; else: failed to set parameters.
00976     */
```

```
00977     int setAgvControllerParametersFile(const std::string &agv_parameters);
00978
00979     /**
00980      * @ingroup System
00981      * @attention
00982      * @zh
00983      * @en Requires control priority.
00984      * @brief
00985      * @zh   AGV        AGV
00986      * @en Refreshes parameters on AGV nodes after the AGV controller parameter file is modified.
00987      * @note
00988      * @zh          1Hz
00989      * @en Do not call frequently. Keep the call rate below 1 Hz.
00990      * @return
00991      * @zh 10100000      else
00992      * @en 10100000: API executed successfully; else: API execution failed.
00993      */
00994     int refreshAgvControllerParametersFile();
00995
00996     /**
00997      * @ingroup System
00998      * @attention
00999      * @zh
01000      * @en Requires control priority.
01001      * @brief
01002      * @zh  AGV
01003      * @en Restores the AGV controller parameter file to factory defaults.
01004      * @note
01005      * @zh          1Hz
01006      * @en Do not call frequently. Keep the call rate below 1 Hz.
01007      * @return
01008      * @zh 10100000      else
01009      * @en 10100000: API executed successfully; else: API execution failed.
01010      */
01011     int resetAgvControllerParametersFile();
01012
01013     /**
01014      * @ingroup Charging
01015      * @attention
01016      * @zh
01017      * @en Requires control priority.
01018      * @brief
01019      * @zh
01020      * @en Checks the current battery level and automatically docks for charging when the battery is low.
01021      * @deprecated @if Chinese  0.7.0      sendAutoChargingCommand(const AutoChargingCommand&)   @endif @if
       English Deprecated since 0.7.0. Use sendAutoChargingCommand(const AutoChargingCommand&) instead. @endif
01022      * @note
01023      * @zh            auto_leave_board
01024      * @en After charging completes, the auto_leave_board parameter decides whether the AGV automatically leaves the
       board and moves to a target station or waiting area.
01025      * @note
01026      * @zh          [low_battery_threshold]     [high_battery_threshold]
01027      * @en Configure the low battery threshold [low_battery_threshold] or high battery threshold [high_battery_threshold]
       in the configuration file.
01028      * @param[in] check_power_header
01029      * @zh     id     name      id
01030      * @en ID of this command, name of this command, current time, and map ID.
01031      * @param[in] nav_board_charge_station_id
01032      * @zh
01033      * @en Drives to the specified charging station for docking and charging. If unset, drives to the nearest charging station.
01034      * @return
01035      * @zh 10100000      AGV      10120202     else
01036      * @en 10100000: API call succeeded and the AGV starts the charging command; 10120202: no control priority; else:
       API call failed.
01037      */
01038     [[deprecated("Since 0.7.0: use sendAutoChargingCommand(const AutoChargingCommand&) instead.")]]
01039     int checkPowerAndAutoCharge(const Header &check_power_header              = { "99999", "99999", 99999, "99999" },
01040                                 const std::string &nav_board_charge_station_id = "99999");
01041
01042     /**
01043      * @ingroup Charging
01044      * @brief
01045      * @zh
01046      * @en Sets the target station for automatic undocking.
01047      * @deprecated @if Chinese  0.7.0      sendAutoChargingCommand(const AutoChargingCommand&)   @endif @if
       English Deprecated since 0.7.0. Use sendAutoChargingCommand(const AutoChargingCommand&) instead. @endif
01048      * @param[in] leave_board_target_station_header
01049      * @zh      header
01050      * @en Header information of the station used for automatic undocking.
01051      * @return
01052      * @zh 10100000        else
01053      * @en 10100000: automatic undocking station set successfully; else: failed to set the station.
01054      */
01055     [[deprecated("Since 0.7.0: use sendAutoChargingCommand(const AutoChargingCommand&) instead.")]]
01056     int setLeaveBoardTargetStation(const Header &leave_board_target_station_header);
01057
01058     /**
```

```
01059     * @ingroup Charging
01060     * @brief
01061     * @zh
01062     * @en Gets the target station for automatic undocking.
01063     * @deprecated @if Chinese  0.7.0       getNavInfo()    @endif @if English Deprecated since 0.7.0. Use getNavInfo()
      instead. @endif
01064     * @return
01065     * @zh        Header
01066     * @en Header information of the target station for automatic undocking.
01067     */
01068    [[deprecated("Since 0.7.0: use getNavInfo() instead.")]]
01069    Header getLeaveBoardTargetStation();
01070
01071    /**
01072     * @ingroup Charging
01073     * @attention
01074     * @zh
01075     * @en Requires control priority.
01076     * @brief
01077     * @zh
01078     * @en Forces the AGV to dock for charging.
01079     * @deprecated @if Chinese  0.7.0       sendAutoChargingCommand(const AutoChargingCommand&)    @endif @if
      English Deprecated since 0.7.0. Use sendAutoChargingCommand(const AutoChargingCommand&) instead. @endif
01080     * @param[in] forced_charge_header
01081     * @zh     id     name      id
01082     * @en ID of this command, name of this command, current time, and map ID.
01083     * @param[in] nav_board_charge_station_id
01084     * @zh
01085     * @en Drives to the specified charging station for docking and charging. If unset, drives to the nearest charging station.
01086     * @return
01087     * @zh 10100000      AGV      10120202      else
01088     * @en 10100000: API call succeeded and the AGV starts docking for charging; 10120202: no control priority; else: API
      call failed.
01089     */
01090    [[deprecated("Since 0.7.0: use sendAutoChargingCommand(const AutoChargingCommand&) instead.")]]
01091    int forcedAgvToChargingBoard(const Header &forced_charge_header             = { "99999", "99999", 99999, "99999" },
01092                                 const std::string &nav_board_charge_station_id = "99999");
01093
01094    /**
01095     * @ingroup Charging
01096     * @attention
01097     * @zh
01098     * @en Requires control priority.
01099     * @brief
01100     * @zh
01101     * @en Forces the AGV to leave the charging board.
01102     * @deprecated @if Chinese  0.7.0       sendAutoChargingCommand(const AutoChargingCommand&)    @endif @if
      English Deprecated since 0.7.0. Use sendAutoChargingCommand(const AutoChargingCommand&) instead. @endif
01103     * @param[in] forced_leave_header
01104     * @zh     id     name      id
01105     * @en ID of this command, name of this command, current time, and map ID.
01106     * @param[in] leave_board_target_station_id
01107     * @zh
01108     * @en Station to move to after leaving the board. If unset, the AGV leaves the board to the charging station.
01109     * @return
01110     * @zh 10100000      AGV     10120202      else
01111     * @en 10100000: API call succeeded and the AGV starts leaving the board; 10120202: no control priority; else: API call
      failed.
01112     */
01113    [[deprecated("Since 0.7.0: use sendAutoChargingCommand(const AutoChargingCommand&) instead.")]]
01114    int forcedAgvLeaveChargingBoard(const Header &forced_leave_header = { "99999", "99999", 99999, "99999" },
01115                                    const std::string &leave_board_target_station_id = "99999");
01116
01117    /**
01118     * @ingroup Charging
01119     * @attention
01120     * @zh
01121     * @en Requires control priority.
01122     * @brief
01123     * @zh
01124     * @en Sends an automatic charging command.
01125     * @since 0.7.0
01126     * @param[in] auto_charging_command
01127     * @zh
01128     * @en Automatic charging command.
01129     * @return
01130     * @zh 10100000      AGV       10120202      else
01131     * @en 10100000: API call succeeded and the AGV starts the charging command; 10120202: no control priority; else:
      API call failed.
01132     */
01133    int sendAutoChargingCommand(const AutoChargingCommand &auto_charging_command);
01134
01135    /**
01136     * @ingroup AgvInfo
01137     * @brief
01138     * @zh   AGV
01139     * @en Gets AGV runtime log messages.
```

```
01140     * @return
01141     * @zh AGV
01142     * @en AGV runtime log messages.
01143     */
01144    std::string getAgvLogMessage();
01145
01146    /**
01147     * @ingroup System
01148     * @brief
01149     * @zh AGV
01150     * @en Upgrades AGV software.
01151     * @param[in] upgrade_pack_path
01152     * @zh       "/root/agvc_release-0.1.1+398d1f7-Linux_x86_64.tar.gz"
01153     * @en Upgrade package path, for example "/root/agvc_release-0.1.1+398d1f7-Linux_x86_64.tar.gz".
01154     * @return
01155     * @zh 10100000       10100201     else
01156     * @en 10100000: target version upgraded successfully; 10100201: upgrade is in progress; else: upgrade failed.
01157     */
01158    int updateSoftware(const std::string &upgrade_pack_path);
01159
01160    /**
01161     * @ingroup System
01162     * @brief
01163     * @zh   AGV
01164     * @en Gets the AGV software version list.
01165     * @return
01166     * @zh       {0.3.2+3b807d1-Linux_x86_64,0.6.3-patch.3+b8c6aa4-Linux_x86_64}
01167     * @en Software version list, for example {0.3.2+3b807d1-Linux_x86_64,0.6.3-patch.3+b8c6aa4-Linux_x86_64}.
01168     */
01169    std::vector<std::string> getSoftwareVersionList();
01170
01171    /**
01172     * @ingroup System
01173     * @brief
01174     * @zh   AGV
01175     * @en Switches the AGV software version.
01176     * @param[in] software_version
01177     * @zh       "0.3.2+3b807d1-Linux_x86_64"
01178     * @en Software version, for example "0.3.2+3b807d1-Linux_x86_64".
01179     * @return
01180     * @zh 10100000       else
01181     * @en 10100000: software version switched successfully; else: failed to switch software version.
01182     */
01183    int switchSoftwareVersion(const std::string &software_version);
01184
01185    /**
01186     * @ingroup System
01187     * @brief
01188     * @zh   AGV
01189     * @en Uninstalls an AGV software version.
01190     * @param[in] software_version
01191     * @zh       "0.3.2+3b807d1-Linux_x86_64"
01192     * @en Software version, for example "0.3.2+3b807d1-Linux_x86_64".
01193     * @return
01194     * @zh 10100000       else
01195     * @en 10100000: software version uninstalled successfully; else: failed to uninstall software version.
01196     */
01197    int uninstallSoftware(const std::string &software_version);
01198
01199    /**
01200     * @ingroup System
01201     * @brief
01202     * @zh AGV
01203     * @en Updates AGV firmware.
01204     * @param[in] update_firmware
01205     * @zh
01206     * @en Configures the firmware modules to update and the firmware storage path.
01207     * @return
01208     * @zh 10100000        else
01209     * @en 10100000: firmware update information sent successfully; else: firmware update is in progress.
01210     */
01211    int updateFirmware(const FirmwareUpdateParam &update_firmware);
01212
01213    /**
01214     * @ingroup System
01215     * @brief
01216     * @zh
01217     * @en Gets firmware update process information.
01218     * @return
01219     * @zh       (    failed    )    (    1.0    )
01220     * @en Firmware update step information (failed means the update failed) and update progress (1.0 means the update
    succeeded).
01221     */
01222    FirmwareUpdateProcessInfo getUpdateFirmwareProcess();
01223
01224    /**
01225     * @ingroup System
```

```
01226     * @brief
01227     * @zh
01228     * @en Starts collecting calibration data.
01229     * @note
01230     * @zh LASER_ODOM        1000
01231     * @en LASER_ODOM calibration requires more than 1000 data samples. The exact amount can be changed in the
      configuration file.
01232     * @param[in] type
01233     * @zh
01234     * @en Calibration type.
01235     * @param[in] command_header
01236     * @zh id        id
01237     * @en The id field is the ID of this command; other fields have no special meaning.
01238     * @return
01239     * @zh 10100000      10120202     else
01240     * @en 10100000: API call succeeded; 10120202: no control priority; else: API call failed.
01241     */
01242    int startCollectCalibrationData(const CalibrationType &type,
01243                                    const Header &command_header = { "99999", "99999", 1, "99999" });
01244
01245    /**
01246     * @ingroup System
01247     * @brief
01248     * @zh
01249     * @en Cancels calibration data collection.
01250     * @param[in] type
01251     * @zh
01252     * @en Calibration type.
01253     * @param[in] command_header
01254     * @zh id        id
01255     * @en The id field is the ID of this command; other fields have no special meaning.
01256     * @return
01257     * @zh 10100000      10120202     else
01258     * @en 10100000: API call succeeded; 10120202: no control priority; else: API call failed.
01259     */
01260    int cancelCollectCalibrationData(const CalibrationType &type,
01261                                     const Header &command_header = { "99999", "99999", 1, "99999" });
01262
01263    /**
01264     * @ingroup System
01265     * @brief
01266     * @zh
01267     * @en Starts calibration.
01268     * @param[in] type
01269     * @zh
01270     * @en Calibration type.
01271     * @param[in] command_header
01272     * @zh id        id
01273     * @en The id field is the ID of this command; other fields have no special meaning.
01274     * @return
01275     * @zh 10100000      10120202     else
01276     * @en 10100000: API call succeeded; 10120202: no control priority; else: API call failed.
01277     */
01278    int startCalibration(const CalibrationType &type, const Header &command_header = { "99999", "99999", 1, "99999" });
01279
01280    /**
01281     * @ingroup System
01282     * @brief
01283     * @zh
01284     * @en Gets calibration information.
01285     * @param[in] type
01286     * @zh
01287     * @en Calibration type.
01288     * @return
01289     * @zh
01290     * @en Calibration information.
01291     */
01292    CalibrationProcessInfo getCalibrationProcessInfo(const CalibrationType &type);
01293
01294    /**
01295     * @ingroup System
01296     * @brief
01297     * @zh  AGV
01298     * @en Restarts the AGV.
01299     * @return
01300     * @zh 10100000       else
01301     * @en 10100000: restart command executed successfully; else: restart command failed.
01302     */
01303    int restartAgv();
01304
01305    /**
01306     * @ingroup System
01307     * @brief
01308     * @zh  AGV
01309     * @en Sets the AGV name.
01310     * @param[in] agv_name
01311     * @zh   AGV
```

```
01312     * @en AGV name to set.
01313     * @return
01314     * @zh 10100000      else
01315     * @en 10100000: name set successfully; else: failed to set the name.
01316     */
01317    int setAgvName(const std::string &agv_name);
01318
01319    /**
01320     * @ingroup System
01321     * @brief
01322     * @zh
01323     * @en Sets control priority.
01324     * @note
01325     * @zh 1.              2.            releasePriority 3.Web     name="web"
01326     * @en 1. Control-priority APIs can be used only after this API succeeds; 2. Release another client's control priority with
        releasePriority before setting priority; 3. To preempt priority from the web side, set name="web".
01327     * @param[in] name
01328     * @zh  RPC
01329     * @en User name registered during RPC login.
01330     * @param[in] ip
01331     * @zh IP
01332     * @en IP address.
01333     * @return
01334     * @zh 10100000      else
01335     * @en 10100000: control priority set successfully; else: failed to set control priority.
01336     */
01337    int setPriority(const std::string &name, const std::string &ip = "");
01338
01339    /**
01340     * @ingroup System
01341     * @brief
01342     * @zh
01343     * @en Releases control priority.
01344     * @return
01345     * @zh 10100000      else
01346     * @en 10100000: control priority released successfully; else: failed to release control priority.
01347     */
01348    int releasePriority();
01349
01350    /**
01351     * @ingroup System
01352     * @brief
01353     * @zh
01354     * @en Pauses the script.
01355     * @return
01356     * @zh 10100000      else
01357     * @en 10100000: script paused successfully; else: failed to pause the script.
01358     */
01359    int scriptPaused();
01360
01361    /**
01362     * @ingroup System
01363     * @brief
01364     * @zh
01365     * @en Resumes the script.
01366     * @return
01367     * @zh 10100000      else
01368     * @en 10100000: script resumed successfully; else: failed to resume the script.
01369     */
01370    int scriptResume();
01371
01372    /**
01373     * @ingroup System
01374     * @brief
01375     * @zh
01376     * @en Stops the script.
01377     * @return
01378     * @zh 10100000      else
01379     * @en 10100000: script stopped successfully; else: failed to stop the script.
01380     */
01381    int scriptStop();
01382
01383    /**
01384     * @ingroup AgvInfo
01385     * @brief
01386     * @zh
01387     * @en Gets the script runtime state.
01388     * @return
01389     * @zh
01390     * @en Script runtime state.
01391     */
01392    ScriptRuntimeState getScriptStatus();
01393
01394    /**
01395     * @ingroup System
01396     * @brief
01397     * @zh
```

```
01398     * @en Sets the script runtime state.
01399     * @return
01400     * @zh 10100000          else
01401     * @en 10100000: script runtime state set successfully; else: failed to set script runtime state.
01402     */
01403    int setScriptStatus(ScriptRuntimeState status);
01404
01405    /**
01406     * @ingroup AgvInfo
01407     * @brief
01408     * @zh
01409     * @en Queries an error code.
01410     * @param[in] error_code
01411     * @zh
01412     * @en Error code.
01413     * @return
01414     * @zh
01415     * @en Meaning of the error code. Empty means the error code is unknown.
01416     */
01417    std::string errorCodeDecoder(const int &error_code);
01418
01419    /**
01420     * @ingroup AgvInfo
01421     * @brief
01422     * @zh
01423     * @en Gets current error codes.
01424     * @return
01425     * @zh
01426     * @en Error code collection.
01427     */
01428    std::vector<Header> getCurrentErrorCodes();
01429
01430    /**
01431     * @ingroup AgvInfo
01432     * @brief
01433     * @zh  AGV
01434     * @en Locates the AGV by automatically playing audio and flashing special lights.
01435     * @return
01436     * @zh 10100000        else
01437     * @en 10100000: locate command sent successfully; else: failed to send the command.
01438     */
01439    int soundLightPrompt();
01440
01441    /**
01442     * @ingroup AgvInfo
01443     * @brief
01444     * @zh AGV
01445     * @en Checks whether there is an obstacle in the AGV travel direction.
01446     * @param[in] detect_distance
01447     * @zh       1.0m
01448     * @en Detection distance. Default: 1.0 m.
01449     * @return
01450     * @zh true         false
01451     * @en true: obstacle exists in the travel direction; false: no obstacle in the travel direction.
01452     */
01453    bool isObstacleAhead(const double &detect_distance = 1.0);
01454
01455    /**
01456     * @ingroup AgvInfo
01457     * @brief
01458     * @zh
01459     * @en Gets the nearest station within the specified detection distance.
01460     * @param[in] detect_distance
01461     * @zh       1.0m
01462     * @en Detection distance. Default: 1.0 m.
01463     * @return
01464     * @zh       id NONE
01465     * @en Nearest station information. id NONE means no station is within the detection distance.
01466     */
01467    StationMark getNearestStation(const double &detect_distance = 1.0);
01468
01469    /**
01470     * @ingroup System
01471     * @attention
01472     * @zh 1.    2.
01473     * @en 1. Asynchronous API; 2. Requires control priority.
01474     * @brief
01475     * @zh
01476     * @en Automatically aligns with the rail and boards the rail.
01477     * @param[in] command_header
01478     * @zh id         id
01479     * @en The id field is the ID of this command; other fields have no special meaning.
01480     * @return
01481     * @zh 10100000      10100201      10120202      else
01482     * @en 10100000: rail boarding succeeded; 10100201: rail boarding is in progress; 10120202: no control priority; else: rail
      boarding failed.
01483     */
```

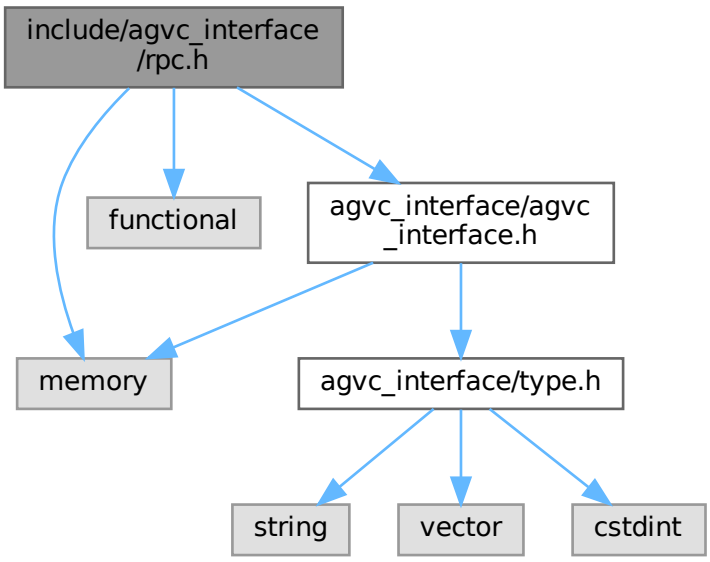
```
01484     int autoAlignRailway(const Header &command_header = { "99999", "99999", 1, "99999" });
01485
01486     /** \cond */
01487 protected:
01488     void *d_;
01489     /** \endcond */
01490 };
01491
01492 using AgvcInterfacePtr = std::shared_ptr<AgvcInterface>;
01493
01494 } // namespace agvc_interface
01495
01496 #define AgvcInterface_DECLARES                                                                                  \
01497     _FUNC(AgvcInterface, 1, setSystemClock, stamp)                                                              \
01498     _FUNC(AgvcInterface, 0, getAgvDetails)                                                                      \
01499     _FUNC(AgvcInterface, 0, getRunningInfo)                                                                     \
01500     _FUNC(AgvcInterface, 0, getAgvCurrentPose)                                                                  \
01501     _FUNC(AgvcInterface, 0, getAsyncInterfaceResultStatus)                                                      \
01502     _FUNC(AgvcInterface, 0, getLaserPointCloud)                                                                 \
01503     _FUNC(AgvcInterface, 0, getAllStations)                                                                     \
01504     _FUNC(AgvcInterface, 1, getAllStationsOfTargetMap, map_header)                                              \
01505     _FUNC(AgvcInterface, 1, addStation, station)                                                                \
01506     _FUNC(AgvcInterface, 1, addStations, stations)                                                              \
01507     _FUNC(AgvcInterface, 2, addCPStationUseChargingBoard, station_header, dis_station_post)                     \
01508     _FUNC(AgvcInterface, 1, deleteStation, station_header)                                                      \
01509     _FUNC(AgvcInterface, 1, deleteStations, stations_header)                                                    \
01510     _FUNC(AgvcInterface, 0, getAllPaths)                                                                        \
01511     _FUNC(AgvcInterface, 1, getAllPathsOfTargetMap, map_header)                                                 \
01512     _FUNC(AgvcInterface, 0, getCurrentPath)                                                                     \
01513     _FUNC(AgvcInterface, 1, generatePath, path_station)                                                         \
01514     _FUNC(AgvcInterface, 1, generatePaths, paths_station)                                                       \
01515     _FUNC(AgvcInterface, 1, deletePath, path_header)                                                            \
01516     _FUNC(AgvcInterface, 1, deletePaths, paths_header)                                                          \
01517     _FUNC(AgvcInterface, 0, getMapList)                                                                         \
01518     _FUNC(AgvcInterface, 0, getCurrentMapHeader)                                                                \
01519     _FUNC(AgvcInterface, 1, getGridMapFromAgv, map_header)                                                      \
01520     _FUNC(AgvcInterface, 1, getBase64PngMapFromAgv, map_header)                                                 \
01521     _FUNC(AgvcInterface, 3, previewPngMapFromAgv, map_header, image_width_px, image_height_px)                  \
01522     _FUNC(AgvcInterface, 1, sendGridMapToAgv, map)                                                              \
01523     _FUNC(AgvcInterface, 1, sendBase64PngMapToAgv, map)                                                         \
01524     _FUNC(AgvcInterface, 1, saveMap, map_header)                                                                \
01525     _FUNC(AgvcInterface, 1, switchMap, map_header)                                                              \
01526     _FUNC(AgvcInterface, 1, deleteMap, map_header)                                                              \
01527     _FUNC(AgvcInterface, 1, deleteMaps, map_headers)                                                            \
01528     _FUNC(AgvcInterface, 0, getAllMapVirtualArea)                                                               \
01529     _FUNC(AgvcInterface, 1, getAllMapVirtualAreaOfTargetMap, map_header)                                        \
01530     _FUNC(AgvcInterface, 1, addMapVirtualArea, map_virtual_area)                                                \
01531     _FUNC(AgvcInterface, 1, addMapVirtualAreas, map_virtual_areas)                                              \
01532     _FUNC(AgvcInterface, 1, deleteMapVirtualArea, virtual_area_header)                                          \
01533     _FUNC(AgvcInterface, 1, deleteMapVirtualAreas, virtual_areas_header)                                        \
01534     _FUNC(AgvcInterface, 1, getGridMapAllInfo, map_header)                                                      \
01535     _FUNC(AgvcInterface, 1, getPngMapAllInfo, map_header)                                                       \
01536     _FUNC(AgvcInterface, 2, setGridMapAllInfo, map_all_info, command_header)                                    \
01537     _FUNC(AgvcInterface, 2, setPngMapAllInfo, map_all_info, command_header)                                     \
01538     _FUNC(AgvcInterface, 1, deleteMapAllInfo, map_header)                                                       \
01539     _FUNC(AgvcInterface, 0, getWifiList)                                                                        \
01540     _FUNC(AgvcInterface, 2, connectWifi, ssid, password)                                                        \
01541     _FUNC(AgvcInterface, 1, enableHotspot, password)                                                            \
01542     _FUNC(AgvcInterface, 0, getIpAddressList)                                                                   \
01543     _FUNC(AgvcInterface, 2, changeRunningMode, running_mode, command_header)                                    \
01544     _FUNC(AgvcInterface, 1, setControlSpeed, speed)                                                             \
01545     _FUNC(AgvcInterface, 1, setNavGoal, target)                                                                 \
01546     _FUNC(AgvcInterface, 0, getNavInfo)                                                                         \
01547     _FUNC(AgvcInterface, 0, pauseAgvSpeed)                                                                      \
01548     _FUNC(AgvcInterface, 0, resumeAgvSpeed)                                                                     \
01549     _FUNC(AgvcInterface, 0, cancelNavigation)                                                                   \
01550     _FUNC(AgvcInterface, 2, relocation, init_pose, command_header)                                              \
01551     _FUNC(AgvcInterface, 0, getAgvControllerParametersFile)                                                     \
01552     _FUNC(AgvcInterface, 1, setAgvControllerParametersFile, agv_parameters)                                     \
01553     _FUNC(AgvcInterface, 0, refreshAgvControllerParametersFile)                                                 \
01554     _FUNC(AgvcInterface, 0, resetAgvControllerParametersFile)                                                   \
01555     _FUNC(AgvcInterface, 2, checkPowerAndAutoCharge, check_power_header, nav_board_charge_station_id)           \
01556     _FUNC(AgvcInterface, 1, setLeaveBoardTargetStation, leave_board_target_station_header)                      \
01557     _FUNC(AgvcInterface, 0, getLeaveBoardTargetStation)                                                         \
01558     _FUNC(AgvcInterface, 2, forcedAgvToChargingBoard, forced_charge_header, nav_board_charge_station_id)        \
01559     _FUNC(AgvcInterface, 2, forcedAgvLeaveChargingBoard, forced_leave_header, leave_board_target_station_id)    \
01560     _FUNC(AgvcInterface, 1, sendAutoChargingCommand, auto_charging_command)                                     \
01561     _FUNC(AgvcInterface, 0, getAgvLogMessage)                                                                   \
01562     _FUNC(AgvcInterface, 1, updateSoftware, upgrade_pack_path)                                                  \
01563     _FUNC(AgvcInterface, 0, getSoftwareVersionList)                                                             \
01564     _FUNC(AgvcInterface, 1, switchSoftwareVersion, software_version)                                            \
01565     _FUNC(AgvcInterface, 1, uninstallSoftware, software_version)                                                \
01566     _FUNC(AgvcInterface, 1, updateFirmware, update_firmware)                                                    \
01567     _FUNC(AgvcInterface, 0, getUpdateFirmwareProcess)                                                           \
01568     _FUNC(AgvcInterface, 2, startCollectCalibrationData, type, command_header)                                  \
01569     _FUNC(AgvcInterface, 2, cancelCollectCalibrationData, type, command_header)                                 \
01570     _FUNC(AgvcInterface, 2, startCalibration, type, command_header)
```

```
01571     _FUNC(AgvcInterface, 1, getCalibrationProcessInfo, type)                                  \
01572     _FUNC(AgvcInterface, 0, restartAgv)                                                   \
01573     _FUNC(AgvcInterface, 1, setAgvName, agv_name)                                          \
01574     _FUNC(AgvcInterface, 2, setPriority, name, ip)                                        \
01575     _FUNC(AgvcInterface, 0, releasePriority)                                              \
01576     _FUNC(AgvcInterface, 0, scriptPaused)                                                 \
01577     _FUNC(AgvcInterface, 0, scriptResume)                                                 \
01578     _FUNC(AgvcInterface, 0, scriptStop)                                                   \
01579     _FUNC(AgvcInterface, 0, getScriptStatus)                                              \
01580     _FUNC(AgvcInterface, 1, setScriptStatus, status)                                      \
01581     _FUNC(AgvcInterface, 1, errorCodeDecoder, error_code)                                  \
01582     _FUNC(AgvcInterface, 0, getCurrentErrorCodes)                                         \
01583     _FUNC(AgvcInterface, 0, soundLightPrompt)                                             \
01584     _FUNC(AgvcInterface, 1, isObstacleAhead, detect_distance)                              \
01585     _FUNC(AgvcInterface, 1, getNearestStation, detect_distance)                            \
01586     _FUNC(AgvcInterface, 1, autoAlignRailway, command_header)
01587 #endif // AGVC_INTERFACE_H
01588
01589
```

13.5 include/agvc_interface/rpc.h File Reference

RPC

#include <memory>
#include <functional>
#include <agvc_interface/agvc_interface.h>
Include dependency graph for rpc.h:



Classes

- class agvc_interface::RpcClient
  RPC

Namespaces

- namespace agvc_interface

Macros

- #define AGVC_ABI
- #define RpcClient_DECLARES

Typedefs

- using agvc_interface::RpcClientPtr = std::shared_ptr<RpcClient>

13.5.1 Detailed Description

RPC

Definition in file rpc.h.

13.5.2 Macro Definition Documentation

13.5.2.1 AGVC_ABI

#define AGVC_ABI

Definition at line 7 of file rpc.h.

13.5.2.2 RpcClient_DECLARES

#define RpcClient_DECLARES

Value:

```
_FUNC(RpcClient, 1, setLogHandler, handler)          \
_FUNC(RpcClient, 2, connect, ip, port)              \
_FUNC(RpcClient, 0, disconnect)                    \
_FUNC(RpcClient, 0, hasConnected)                   \
_FUNC(RpcClient, 2, login, usrname, passwd)         \
_FUNC(RpcClient, 0, logout)                       \
_FUNC(RpcClient, 0, hasLogined)                     \
_FUNC(RpcClient, 1, setRequestTimeout, timeout)     \
_FUNC(RpcClient, 1, setExceptionFree, timenableeout) \
_FUNC(RpcClient, 0, errorCode)
```

Definition at line 151 of file rpc.h.

13.6 rpc.h

Go to the documentation of this file.

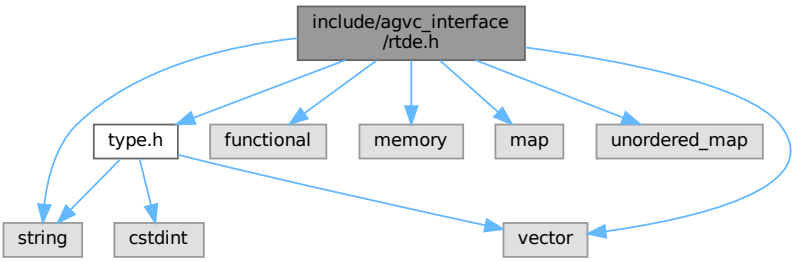
```
00001 /** @file  rpc.h
00002  *  @brief  RPC
00003  */
00004 #ifndef AGVC_INTERFACE_RPC_H
00005 #define AGVC_INTERFACE_RPC_H
00006
00007 #define AGVC_ABI
00008
00009 #include <memory>
00010 #include <functional>
00011
00012 #include <agvc_interface/agvc_interface.h>
00013
00014 namespace agvc_interface {
00015
00016 /// RPC
00017 class AGVC_ABI RpcClient : public AgvcInterface
00018 {
00019 public:
00020     enum Event
00021     {
00022         Connected,
00023         Disconnected,
00024     };
00025
00026     /**
00027      * @brief RpcClient
00028      * @param mode 0-TCP 1-UDS
00029      */
00030     RpcClient(int mode = 0);
00031     ~RpcClient();
00032
00033     /**
00034      *
00035      *
00036      *                  \n
00037      * Agvc SDK
00038      *
00039      *              Agvc SDK
00040      *
00041      * @note setLogHandler
00042      *
00043      *
00044      * @param handler       \n
00045      *          : \n
00046      * void handler(int level, const char* filename, int line, const
00047      * std::string& message) \n
00048      * level       \n
00049      * &amp;nbsp; 0: LOGLEVEL_FATAL      \n
00050      * &amp;nbsp; 1: LOGLEVEL_ERROR        \n
00051      * &amp;nbsp; 2: LOGLEVEL_WARNING       \n
00052      * &amp;nbsp; 3: LOGLEVEL_INFO         \n
00053      * &amp;nbsp; 4: LOGLEVEL_DEBUG         \n
00054      * &amp;nbsp; 5: LOGLEVEL_BACKTRACE     \n
00055      * filename      \n
00056      * line       \n
00057      * message       \n
00058      * @return
00059      */
00060     void setLogHandler(
00061         std::function<void(int /*level*/, const char * /*filename*/, int /*line*/, const std::string & /*message*/)>
00062             handler);
00063
00064     /**
00065      *   RPC
00066      *
00067      * @param ip IP
00068      * @param port    RPC    30104
00069      * @param ip port     unix domain sockets
00070      * @retval 0 RPC
00071      * @retval -8 RPC    RPC
00072      * @retval -15 RPC    SDK   Server
00073      */
00074     int connect(const std::string &ip = "", int port = 0);
00075
00076     /**
00077      *   RPC
00078      *
00079      * @retval 0
00080      * @retval -1
00081      */
00082     int disconnect();
```

```
00083
00084     /**
00085      *      RPC
00086      *
00087      * @retval true    RPC
00088      * @retval false    RPC
00089      */
00090     bool hasConnected() const;
00091
00092     /**
00093      *
00094      *
00095      * @param usrname
00096      * @param passwd
00097      * @return 0
00098      */
00099     int login(const std::string &usrname, const std::string &passwd);
00100
00101     /**
00102      *
00103      *
00104      * @return 0
00105      */
00106     int logout();
00107
00108     /**
00109      *
00110      *
00111      * @retval true
00112      * @retval false
00113      */
00114     bool hasLogined();
00115
00116     /**
00117      *   RPC
00118      *
00119      * @param timeout         ms
00120      * @return 0
00121      */
00122     int setRequestTimeout(int timeout = 10);
00123
00124     /**
00125      *
00126      *
00127      * @param cb
00128      * @return
00129      */
00130     int setEventHandler(std::function<void(int /*event*/)> cb);
00131
00132     /**
00133      *
00134      *
00135      * @param enable
00136      * @return
00137      */
00138     int setExceptionFree(bool enable);
00139
00140     /**
00141      *
00142      *
00143      * @return
00144      */
00145     int errorCode() const;
00146 };
00147 using RpcClientPtr = std::shared_ptr<RpcClient>;
00148
00149 } // namespace agvc_interface
00150
00151 #define RpcClient_DECLARES                              \
00152     _FUNC(RpcClient, 1, setLogHandler, handler)          \
00153     _FUNC(RpcClient, 2, connect, ip, port)              \
00154     _FUNC(RpcClient, 0, disconnect)                     \
00155     _FUNC(RpcClient, 0, hasConnected)                    \
00156     _FUNC(RpcClient, 2, login, usrname, passwd)          \
00157     _FUNC(RpcClient, 0, logout)                         \
00158     _FUNC(RpcClient, 0, hasLogined)                      \
00159     _FUNC(RpcClient, 1, setRequestTimeout, timeout)      \
00160     _FUNC(RpcClient, 1, setExceptionFree, timenableeout) \
00161     _FUNC(RpcClient, 0, errorCode)
00162
00163 #endif
```

13.7 include/agvc_interface/rtde.h File Reference

RPC

#include <string>
#include <vector>
#include <functional>
#include <memory>
#include <map>
#include <unordered_map>
#include "type.h"
Include dependency graph for rtde.h:



Classes

- class agvc_interface::OutputBuilder
- class agvc_interface::InputParser
- class agvc_interface::RtdeClient
  RTDE

Namespaces

- namespace agvc_interface

Macros

- #define AGVC_ABI
- #define RtdeClient_DECLARES

Typedefs

- using agvc_interface::RtdeClientPtr = std::shared_ptr<RtdeClient>

13.7.1 Detailed Description

RPC

Definition in file rtde.h.

13.7.2 Macro Definition Documentation

13.7.2.1 AGVC_ABI

#define AGVC_ABI

Definition at line 7 of file rtde.h.

13.7.2.2 RtdeClient_DECLARES

#define RtdeClient_DECLARES

Value:

```
_FUNC(RtdeClient, 1, setLogHandler, handler)                          \
_FUNC(RtdeClient, 2, connect, ip, port)                              \
_FUNC(RtdeClient, 0, disconnect)                                     \
_FUNC(RtdeClient, 0, hasConnected)                                    \
_FUNC(RtdeClient, 1, hasConnected1, callback)                          \
_FUNC(RtdeClient, 2, login, usrname, passwd)                           \
_FUNC(RtdeClient, 0, logout)                                        \
_FUNC(RtdeClient, 0, hasLogined)                                     \
_FUNC(RtdeClient, 0, getProtocolVersion)                              \
_FUNC(RtdeClient, 0, getInputMaps)                                    \
_FUNC(RtdeClient, 0, getOutputMaps)                                    \
_FUNC(RtdeClient, 4, setTopic, to_server, names, freq, expected_chanel) \
_FUNC(RtdeClient, 2, removeTopic, to_server, chanel)                     \
_FUNC(RtdeClient, 0, getRegisteredInputRecipe)                          \
_FUNC(RtdeClient, 0, getRegisteredOutputRecipe)                          \
_FUNC(RtdeClient, 2, subscribe, chanel, callback)                      \
_FUNC(RtdeClient, 2, publish, chanel, callback)                        \
_FUNC(RtdeClient, 1, setEventHandler, cb)
```

Definition at line 288 of file rtde.h.

13.8 rtde.h

Go to the documentation of this file.

```
00001 /** @file  rtde.h
00002  *  @brief  RPC
00003  */
00004 #ifndef AGVC_SDK_RTDE_H
00005 #define AGVC_SDK_RTDE_H
00006
00007 #define AGVC_ABI
00008
00009 #include <string>
00010 #include <vector>
00011 #include <functional>
00012 #include <memory>
00013 #include <map>
00014 #include <unordered_map>
00015
00016 #include "type.h"
00017
00018 namespace agvc_interface {
00019
00020 class RtdeClient;
00021
```

```
00022 ///
00023 class AGVC_ABI OutputBuilder
00024 {
00025 public:
00026     OutputBuilder();
00027     ~OutputBuilder();
00028
00029     OutputBuilder &push(int val);
00030     OutputBuilder &push(double val);
00031     OutputBuilder &push(const std::vector<double> &val);
00032     OutputBuilder &push(const std::tuple<int, bool> &val);
00033     OutputBuilder &push(int16_t &val);
00034     OutputBuilder &push(const std::vector<int16_t> &val);
00035     OutputBuilder &push(const std::vector<int> &val);
00036     OutputBuilder &push(const std::string &val);
00037     OutputBuilder &push(char val);
00038     OutputBuilder &push(const agvc_interface::RtdeRecipe &val);
00039
00040
00041 private:
00042     friend RtdeClient;
00043     class Impl;
00044     Impl *impl;
00045 };
00046
00047 ///
00048 class AGVC_ABI InputParser
00049 {
00050 public:
00051     InputParser();
00052     ~InputParser();
00053
00054     bool popBool();
00055     int popInt32();
00056     int64_t popInt64();
00057     int16_t popInt16();
00058     double popDouble();
00059     char popChar();
00060     std::string popString();
00061     std::vector<int> popVectorInt();
00062     std::vector<int16_t> popVectorInt16();
00063     std::vector<double> popVectorDouble();
00064     std::vector<std::vector<double» popVectorVectorDouble();
00065     agvc_interface::Pose2d popPose2d();
00066     agvc_interface::RunningInfo popRunningInfo();
00067     agvc_interface::AgvDetails popAgvDetails();
00068     agvc_interface::NavInfo popNavInfo();
00069     std::vector<agvc_interface::Point2d> popVectorPoint2d();
00070     std::vector<agvc_interface::Header> popVectorHeader();
00071
00072 private:
00073     friend RtdeClient;
00074     class Impl;
00075     Impl *impl;
00076 };
00077
00078 /// RTDE
00079 class AGVC_ABI RtdeClient
00080 {
00081 public:
00082     enum Event
00083     {
00084         Connected,
00085         Disconnected,
00086     };
00087
00088     /**
00089      * RTDE
00090      *
00091      * @param mode       0 tcp  1 uds
00092      */
00093     RtdeClient(int mode = 0);
00094     ~RtdeClient();
00095
00096     /**
00097      *
00098      *
00099      *                  \n
00100      * Aubo SDK
00101      *
00102      *              Aubo SDK
00103      *
00104      * @note setLogHandler
00105      *
00106      *
00107      * @param handler       \n
00108      *          : \n
```

```
00109     * void handler(int level, const char* filename, int line, const
00110     * std::string& message) \n
00111     * level       \n
00112     * &amp;nbsp; 0: LOGLEVEL_FATAL      \n
00113     * &amp;nbsp; 1: LOGLEVEL_ERROR        \n
00114     * &amp;nbsp; 2: LOGLEVEL_WARNING      \n
00115     * &amp;nbsp; 3: LOGLEVEL_INFO         \n
00116     * &amp;nbsp; 4: LOGLEVEL_DEBUG        \n
00117     * &amp;nbsp; 5: LOGLEVEL_BACKTRACE    \n
00118     * filename      \n
00119     * line      \n
00120     * message       \n
00121     * @return
00122     */
00123    void setLogHandler(
00124        std::function<void(int /*level*/, const char * /*filename*/, int /*line*/, const std::string & /*message*/)>
00125            handler);
00126
00127    /**
00128     *
00129     *
00130     * @param ip IP
00131     * @param port    RTDE    30110
00132     * @retval 0
00133     * @retval 1
00134     * @retval -1
00135     */
00136    int connect(const std::string &ip = "", int port = 0);
00137
00138    /**
00139     * socket
00140     *
00141     * @retval true   socket
00142     * @retval false   socket
00143     */
00144    bool hasConnected() const;
00145
00146    /**
00147     * socket
00148     *
00149     * @param callback
00150     * @retval true   socket
00151     * @retval false   socket
00152     */
00153    bool hasConnected1(std::function<void(bool)> callback);
00154
00155    /**
00156     *
00157     *
00158     * @param usrname
00159     * @param passwd
00160     * @retval 0
00161     * @retval -1
00162     */
00163    int login(const std::string &usrname, const std::string &passwd);
00164
00165    /**
00166     *
00167     *
00168     * @retval true
00169     * @retval false
00170     */
00171    bool hasLogined();
00172
00173    /**
00174     *
00175     *
00176     * @return 0
00177     */
00178    int logout();
00179
00180    /**
00181     *
00182     *
00183     * @retval 0
00184     * @retval -1
00185     */
00186    int disconnect();
00187
00188    /**
00189     *
00190     *
00191     * @return
00192     */
00193    int getProtocolVersion();
00194
00195    /**
```

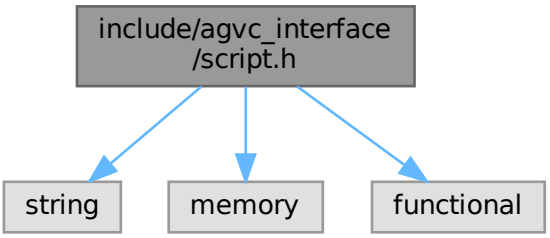
```
00196     *
00197     *
00198     * @return
00199     */
00200    std::map<std::string, int> getInputMaps();
00201
00202    /**
00203     *
00204     *
00205     * @return
00206     */
00207    std::map<std::string, int> getOutputMaps();
00208
00209    /**
00210     *
00211     *
00212     * @param to_server
00213     * true          false
00214     * @param names
00215     * @param freq
00216     * @param expected_chanel
00217     *    0~99
00218     *
00219     * @retval expected_chanel
00220     * @retval -1
00221     */
00222    int setTopic(bool to_server, const std::vector<std::string> &names, double freq, int expected_chanel);
00223
00224    /**
00225     *
00226     *
00227     * @param to_server
00228     * true          false
00229     * @param chanel
00230     * @retval 0
00231     * @retval 1
00232     */
00233    int removeTopic(bool to_server, int chanel);
00234
00235    /**
00236     *
00237     *
00238     * @return
00239     */
00240    std::unordered_map<int, agvc_interface::RtdeRecipe> getRegisteredInputRecipe();
00241
00242    /**
00243     *
00244     *
00245     * @return
00246     */
00247    std::unordered_map<int, agvc_interface::RtdeRecipe> getRegisteredOutputRecipe();
00248
00249    /**
00250     *    subscribe from output
00251     *
00252     * @param chanel
00253     * @param callback             \n
00254     *
00255     * void callback(InputParser &parser)
00256     * @return 0
00257     */
00258    int subscribe(int chanel, std::function<void(InputParser &)> callback);
00259
00260    /**
00261     *    publish to input
00262     *
00263     * @param chanel
00264     * @param callback             \n
00265     *
00266     * void callback(OutputBuilder &builder)
00267     * @retval 0
00268     * @retval -1
00269     */
00270    int publish(int chanel, std::function<void(OutputBuilder &)> callback);
00271
00272    /**
00273     *
00274     *
00275     * @param cb
00276     * @return
00277     */
00278    int setEventHandler(std::function<void(int /*event*/)> cb);
00279
00280 private:
00281    class Impl;
00282    Impl *impl;
```

```
00283 };
00284 using RtdeClientPtr = std::shared_ptr<RtdeClient>;
00285
00286 } // namespace agvc_interface
00287
00288 #define RtdeClient_DECLARES                                                    \
00289     _FUNC(RtdeClient, 1, setLogHandler, handler)                               \
00290     _FUNC(RtdeClient, 2, connect, ip, port)                                    \
00291     _FUNC(RtdeClient, 0, disconnect)                                           \
00292     _FUNC(RtdeClient, 0, hasConnected)                                         \
00293     _FUNC(RtdeClient, 1, hasConnected1, callback)                              \
00294     _FUNC(RtdeClient, 2, login, usrname, passwd)                               \
00295     _FUNC(RtdeClient, 0, logout)                                               \
00296     _FUNC(RtdeClient, 0, hasLogined)                                           \
00297     _FUNC(RtdeClient, 0, getProtocolVersion)                                   \
00298     _FUNC(RtdeClient, 0, getInputMaps)                                         \
00299     _FUNC(RtdeClient, 0, getOutputMaps)                                        \
00300     _FUNC(RtdeClient, 4, setTopic, to_server, names, freq, expected_chanel) \
00301     _FUNC(RtdeClient, 2, removeTopic, to_server, chanel)                       \
00302     _FUNC(RtdeClient, 0, getRegisteredInputRecipe)                             \
00303     _FUNC(RtdeClient, 0, getRegisteredOutputRecipe)                            \
00304     _FUNC(RtdeClient, 2, subscribe, chanel, callback)                          \
00305     _FUNC(RtdeClient, 2, publish, chanel, callback)                            \
00306     _FUNC(RtdeClient, 1, setEventHandler, cb)
00307
00308 #endif
```

13.9 include/agvc_interface/script.h File Reference

SCRIPT

#include <string>
#include <memory>
#include <functional>
Include dependency graph for script.h:



Classes

- class agvc_interface::ScriptWriter
- class agvc_interface::ScriptClient

  SCRIPT

Namespaces

- namespace agvc_interface

Macros

- #define AGVC_ABI

Typedefs

- using agvc_interface::ScriptClientPtr = std::shared_ptr<ScriptClient>

13.9.1 Detailed Description

SCRIPT

Definition in file script.h.

13.9.2 Macro Definition Documentation

13.9.2.1 AGVC_ABI

#define AGVC_ABI

Definition at line 6 of file script.h.

13.10 script.h

Go to the documentation of this file.

```
00001 /** @file  script.h
00002  *  @brief  SCRIPT
00003  */
00004 #ifndef AUBO_SDK_SCRIPT_H
00005 #define AUBO_SDK_SCRIPT_H
00006 #define AGVC_ABI
00007
00008 #include <string>
00009 #include <memory>
00010 #include <functional>
00011
00012 namespace agvc_interface {
00013
00014 class ScriptWriter
00015 {
00016 public:
00017     virtual ~ScriptWriter() = default;
00018
00019     virtual ScriptWriter &append(const std::string &line)     = 0;
00020     virtual ScriptWriter &append(const char *buf, size_t len) = 0;
00021     virtual ScriptWriter &moveJoint()                         = 0;
00022     virtual ScriptWriter &moveLine()                          = 0;
00023     virtual ScriptWriter &ifCondition()                       = 0;
00024     virtual ScriptWriter &elseCondition()                     = 0;
00025     virtual ScriptWriter &elseIfCondition()                   = 0;
00026     virtual ScriptWriter &whileCondition()                    = 0;
00027     virtual ScriptWriter &end()                               = 0;
00028 };
00029
00030 /// SCRIPT
00031 class AGVC_ABI ScriptClient
00032 {
00033 public:
00034     enum Event
00035     {
00036         Connected,
00037         Disconnected,
00038     };
```

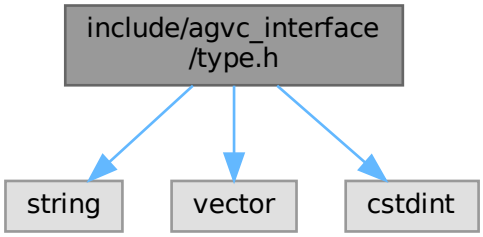
```
00039
00040   ScriptClient(int mode = 0);
00041   ~ScriptClient();
00042
00043   /**
00044    *
00045    *
00046    *                  \n
00047    * Aubo SDK
00048    *
00049    *              Aubo SDK
00050    *
00051    * @note setLogHandler
00052    *
00053    *
00054    * @param handler       \n
00055    *          : \n
00056    * void handler(int level, const char* filename, int line, const
00057    * std::string& message) \n
00058    * level       \n
00059    * &amp;nbsp; 0: LOGLEVEL_FATAL      \n
00060    * &amp;nbsp; 1: LOGLEVEL_ERROR        \n
00061    * &amp;nbsp; 2: LOGLEVEL_WARNING       \n
00062    * &amp;nbsp; 3: LOGLEVEL_INFO         \n
00063    * &amp;nbsp; 4: LOGLEVEL_DEBUG        \n
00064    * &amp;nbsp; 5: LOGLEVEL_BACKTRACE     \n
00065    * filename      \n
00066    * line       \n
00067    * message       \n
00068    * @return
00069    */
00070   void setLogHandler(
00071       std::function<void(int /*level*/, const char * /*filename*/, int /*line*/, const std::string & /*message*/)>
00072           handler);
00073
00074   /**
00075    *
00076    *
00077    * @param ip IP
00078    * @param port    SCRIPT   30004
00079    * @retval 0
00080    * @retval 1
00081    * @retval -1
00082    */
00083   int connect(const std::string &ip = "", int port = 0);
00084
00085   /**
00086    *
00087    *
00088    * @retval true
00089    * @retval false
00090    */
00091   bool hasConnected() const;
00092
00093   /**
00094    *
00095    *
00096    * @param usrname
00097    * @param passwd
00098    * @retval 0
00099    * @retval -1
00100    */
00101   int login(const std::string &usrname, const std::string &passwd);
00102
00103   /**
00104    *
00105    *
00106    * @retval true
00107    * @retval false
00108    */
00109   bool hasLogined();
00110
00111   /**
00112    *
00113    *
00114    * @return 0
00115    */
00116   int logout();
00117
00118   /**
00119    *
00120    *
00121    * @retval 0
00122    * @retval -1
00123    */
00124   int disconnect();
00125
```

```
00126     /**
00127      *
00128      *
00129      *
00130      *
00131      * @param path
00132      * @retval 0
00133      * @retval -1
00134      */
00135     int sendFile(const std::string &path);
00136
00137     /**
00138      *
00139      *
00140      *
00141      *
00142      * @param script
00143      * @retval 0
00144      * @retval -1
00145      */
00146     int sendString(const std::string &script);
00147
00148     /**
00149      *   ScriptWriter
00150      *
00151      * @param chunck_name
00152      * @param cb
00153      * @retval 0
00154      * @retval -1
00155      */
00156     int send(const std::string &chunck_name, std::function<int(ScriptWriter &)> cb);
00157
00158     /**
00159      *
00160      *
00161      * @param cb
00162      * @return 0
00163      */
00164     int subscribeVariableUpdate(std::function<void(const std::string &, const std::string &)> cb);
00165
00166     /**
00167      *
00168      *
00169      * @param cb
00170      * @return 0
00171      */
00172     [[deprecated]] int subscribeScriptError(std::function<void(const std::string &)> cb);
00173
00174     int subscribeScriptError2(std::function<void(const std::string &, const std::string &)> cb);
00175
00176     /**
00177      *
00178      *
00179      * @param cb
00180      * @return
00181      */
00182     int setEventHandler(std::function<void(int /*event*/)> cb);
00183
00184 private:
00185     class Impl;
00186     Impl *impl;
00187 };
00188 using ScriptClientPtr = std::shared_ptr<ScriptClient>;
00189
00190 } // namespace agvc_interface
00191 #endif
```

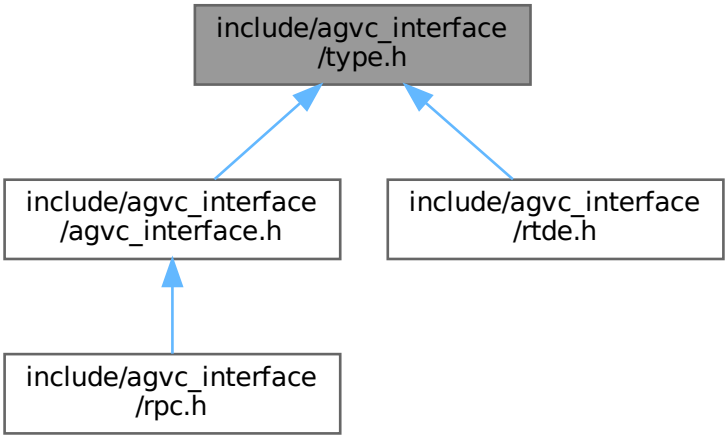
13.11 include/agvc_interface/type.h File Reference

#include <string>
#include <vector>
#include <cstdint>

Include dependency graph for type.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct agvc_interface::Point2d
- struct agvc_interface::Pose2d
- struct agvc_interface::Speed
- struct agvc_interface::Header
- struct agvc_interface::AgvDetails
  agv
- struct agvc_interface::RunningInfo

agv

- struct agvc_interface::SaveMapWorkingStatus

  saveMap
- struct agvc_interface::SwitchMapWorkingStatus

  switchMap
- struct agvc_interface::ChangeModeWorkingStatus

  changeRunningMode
- struct agvc_interface::RelocationWorkingStatus

  Relocation
- struct agvc_interface::SetPngMapAllInfoWorkingStatus

  setPngMapAllInfo
- struct agvc_interface::GetPngMapAllInfoWorkingStatus

  getPngMapAllInfo
- struct agvc_interface::GetGridMapWorkingStatus

  getGridMapFromAgv
- struct agvc_interface::SendGridMapWorkingStatus

  sendGridMapToAgv
- struct agvc_interface::GetPngMapWorkingStatus
- struct agvc_interface::SendPngMapWorkingStatus

  sendBase64PngMapToAgv
- struct agvc_interface::SetGridMapAllInfoWorkingStatus

  setGridMapAllInfo
- struct agvc_interface::GetGridMapAllInfoWorkingStatus

  getGridMapAllInfo
- struct agvc_interface::PreviewPngMapWorkingStatus

  previewPngMapFromAgv
- struct agvc_interface::AutoAlignRailwayWorkingStatus

  autoAlignRailway
- struct agvc_interface::AsyncInterfaceResultStatus

  14 saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, previewPngMapFromAgv, autoAlignRailway
- struct agvc_interface::StationMark
- struct agvc_interface::PathStation
- struct agvc_interface::MapInfo
- struct agvc_interface::MapVirtualArea
- struct agvc_interface::MapAllInfo
- struct agvc_interface::IpAddressInfo

  IP
- struct agvc_interface::AdjustParam

  agv
- struct agvc_interface::NavGoalType

  agv
- struct agvc_interface::NavInfo

  /
- struct agvc_interface::AutoChargingCommand

- struct agvc_interface::FirmwareUpdateParam

- struct agvc_interface::FirmwareUpdateProcessInfo

- struct agvc_interface::CalibrationProcessInfo
- struct agvc_interface::RtdeRecipe
  RTDE
- class agvc_interface::AgvcException

Namespaces

- namespace agvc_interface

Macros

- #define ENUM_AgvcErrorCodes_DECLARES

- #define ENUM_ITEM(c, n, ...)
- #define ENUM_ITEM(n, v, s)
- #define ENUM_ITEM(n, v, s)
- #define ENUM_ITEM(n, v, s)

Typedefs

- typedef FeedbackStatus agvc_interface::NavStatus
  agv
- typedef FeedbackStatus agvc_interface::SaveMapStatus
  saveMap
- typedef FeedbackStatus agvc_interface::SwitchMapStatus
  switchMap
- typedef FeedbackStatus agvc_interface::ChangeRunningModeStatus
  changeRunningMode
- typedef FeedbackStatus agvc_interface::RelocationStatus
  relocation
- typedef FeedbackStatus agvc_interface::SetPngMapAllInfoStatus
  Png setPngMapAllInfo
- typedef FeedbackStatus agvc_interface::GetPngMapAllInfoStatus
  Png getPngMapAllInfo
- typedef FeedbackStatus agvc_interface::GetGridMapStatus
  getGridMapFromAgv
- typedef FeedbackStatus agvc_interface::SendGridMapStatus
  sendGridMapToAgv
- typedef FeedbackStatus agvc_interface::GetPngMapStatus
  png getBase64PngMapFromAgv
- typedef FeedbackStatus agvc_interface::SendPngMapStatus
  png sendBase64PngMapToAgv
- typedef FeedbackStatus agvc_interface::SetGridMapAllInfoStatus
  setGridMapAllInfo
- typedef FeedbackStatus agvc_interface::GetGridMapAllInfoStatus

getGridMapAllInfo

- typedef FeedbackStatus agvc_interface::PreviewPngMapStatus

previewPngMapFromAgv

- typedef FeedbackStatus agvc_interface::AutoAlignRailwayStatus

autoAlignRailway

- typedef FeedbackStatus agvc_interface::CalibrationStatus

- typedef MapInfo agvc_interface::OccupancyGridMap

- typedef MapInfo agvc_interface::Base64PngMap

Base64 png

Enumerations

- enum agvc_interface::AgvcErrorCodes : int { agvc_interface::ENUM_AgvcErrorCodes_DECLARES }
- enum class agvc_interface::RunningMode { agvc_interface::NONE = 0 , agvc_interface::START = 1 , agvc_interface::MAPPING = 2 , agvc_interface::NAVIGATE = 3 , agvc_interface::CHARGING = 4 , agvc_interface::MAINTAIN = 5 , agvc_interface::STOP = 6 }

agv

- enum class agvc_interface::PathShape { agvc_interface::NONE = 0 , agvc_interface::LINE = 1 , agvc_interface::ARC = 2 , agvc_interface::BEZIER = 3 }

- enum class agvc_interface::MapFormat { agvc_interface::NONE = 0 , agvc_interface::OCCUPANCY_GRID = 1 , agvc_interface::BASE64_PNG = 2 }

- enum class agvc_interface::MapVirtualAreaType { agvc_interface::NONE = 0 , agvc_interface::OBSTACLE = 1 , agvc_interface::SLOW_DOWN = 2 , agvc_interface::INFLATE = 3 }

- enum class agvc_interface::MapVirtualAreaShape { agvc_interface::NONE = 0 , agvc_interface::LINE = 1 , agvc_interface::ARC = 2 , agvc_interface::CIRCLE = 3 , agvc_interface::POLYGON = 4 }

- enum class agvc_interface::NavType { agvc_interface::NONE = 0 , agvc_interface::FREE_TO_POSE = 1 , agvc_interface::FREE_TO_STATION = 2 , agvc_interface::PATH_TO_STATION = 3 }

agv

- enum class agvc_interface::AutoChargingType { agvc_interface::NONE = 0 , agvc_interface::LOW_POWER_TO_BOARD = 1 , agvc_interface::HIGH_POWER_ = 2 , agvc_interface::FORCE_TO_BOARD = 3 , agvc_interface::FORCE_LEAVE_BOARD = 4 }

- enum class agvc_interface::FeedbackStatus { agvc_interface::NONE = 0 , agvc_interface::FINISHED = 1 , agvc_interface::FAILED = 2 , agvc_interface::RUNNING = 3 , agvc_interface::PAUSED = 4 , agvc_interface::CANCELED = 5 }

- enum class agvc_interface::FirmwareUpdateMode { agvc_interface::NONE = 0 , agvc_interface::UPDATE = 1 , agvc_interface::FORCED_UPDATE = 2 , agvc_interface::NOT_UPDATE = 3 }

- enum class agvc_interface::ScriptRuntimeState { agvc_interface::RUNNING = 0 , agvc_interface::PAUSED = 1 , agvc_interface::STOPPED = 2 , agvc_interface::ABORTING = 3 }

- enum class agvc_interface::CalibrationType { agvc_interface::NONE = 0 , agvc_interface::LASER_ODOM = 1 }

- enum agvc_interface::error_type { agvc_interface::parse_error = -32700 , agvc_interface::invalid_request = -32600 , agvc_interface::method_not_fou = -32601 , agvc_interface::invalid_params = -32602 , agvc_interface::internal_error = -32603 , agvc_interface::server_error , agvc_interface::invalid }
- enum agvc_interface::ExceptionCode { agvc_interface::EC_DISCONNECTED = -1 , agvc_interface::EC_NOT_LOGINED = -2 , agvc_interface::EC_INVAL_SOCKET = -3 , agvc_interface::EC_REQUEST_BUSY = -4 , agvc_interface::EC_SEND_FAILED = -5 , agvc_interface::EC_RECV_TIMEOUT = -6 , agvc_interface::EC_RECV_ERROR = -7 , agvc_interface::EC_PARSE_ERROR = -8 , agvc_interface::EC_INVALID_REQUEST = -9 , agvc_interface::EC_METHOD_NOT_FOUND = -10 , agvc_interface::EC_INVALID_PARAMS = -11 , agvc_interface::EC_INTERNAL_ERROR = -12 , agvc_interface::EC_SERVER_ERROR = -13 , agvc_interface::EC_INVALID = -14 }

Functions

- const char * agvc_interface::returnValue2Str (int retval)

13.11.1 Macro Definition Documentation

13.11.1.1 ENUM_AgvcErrorCodes_DECLARES

```
#define ENUM_AgvcErrorCodes_DECLARES
```

\+ 0

Definition at line 12 of file type.h.

13.11.1.2 ENUM_ITEM [1/4]

```
#define ENUM_ITEM(
                c,
                n,
                ...)
```

Value:

```
c = n,
```

Definition at line 70 of file type.h.

13.11.1.3 ENUM_ITEM [2/4]

```
#define ENUM_ITEM(
                n,
                v,
                s)
```

Value:

```
if (retval == v)        \
    index = n##_INDEX;
```

Definition at line 70 of file type.h.

13.11.1.4 ENUM_ITEM [3/4]

#define ENUM_ITEM(
n,
v,
s)

Value:
n##_INDEX,

Definition at line 70 of file type.h.

13.11.1.5 ENUM_ITEM [4/4]

#define ENUM_ITEM(
n,
v,
s)

Value:
s,

Definition at line 70 of file type.h.

13.12 type.h

Go to the documentation of this file.

```
00001 #ifndef AGVC_INTERFACE_TYPE_H
00002 #define AGVC_INTERFACE_TYPE_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <cstdint>
00007
00008 namespace agvc_interface {
00009 ///
00010 ///
00011 /// +       0
00012 #define ENUM_AgvcErrorCodes_DECLARES                                    \
00013     ENUM_ITEM(AGVC_OK, 0, "  ")                                  \
00014     ENUM_ITEM(AGVC_BAD_STATE, 1, "   ")                           \
00015     ENUM_ITEM(AGVC_QUEUE_FULL, 2, "    ")                         \
00016     ENUM_ITEM(AGVC_BUSY, 3, "       ")                      \
00017     ENUM_ITEM(AGVC_TIMEOUT, 4, "  ")                             \
00018     ENUM_ITEM(AGVC_INVL_ARGUMENT, 5, "   ")                       \
00019     ENUM_ITEM(AGVC_NOT_IMPLETEMENT, 6, "    ")                     \
00020     ENUM_ITEM(AGVC_NO_ACCESS, 7, "   ")                          \
00021     ENUM_ITEM(AGVC_CONN_REFUSED, 8, "    ")                       \
00022     ENUM_ITEM(AGVC_CONN_RESET, 9, "    ")                         \
00023     ENUM_ITEM(AGVC_INPROGRESS, 10, "    ")                       \
00024     ENUM_ITEM(AGVC_EIO, 11, "input/output error")                   \
00025     ENUM_ITEM(AGVC_NOBUFFS, 12, "")                               \
00026     ENUM_ITEM(AGVC_REQUEST_IGNORE, 13, "     ")                   \
00027     ENUM_ITEM(AGVC_ALGORITHM_PLAN_FAILED, 14, "      ")          \
00028     ENUM_ITEM(AGVC_VERSION_INCOMPAT, 15, "     ")                 \
00029     ENUM_ITEM(AGVC_DIMENSION_ERR, 16, "     ")                   \
00030     ENUM_ITEM(AGVC_SINGULAR_ERR, 17, "       ")                 \
00031     ENUM_ITEM(AGVC_POS_BOUND_ERR, 18, "         ")              \
00032     ENUM_ITEM(AGVC_INIT_POS_ERR, 19, "       ")                 \
00033     ENUM_ITEM(AGVC_ELP_SETTING_ERR, 20, "     ")                  \
00034     ENUM_ITEM(AGVC_TRAJ_GEN_FAIL, 21, "    ")                    \
00035     ENUM_ITEM(AGVC_TRAJ_SELF_COLLISION, 22, "    ")                \
00036     ENUM_ITEM(AGVC_IK_NO_CONVERGE, 23, "         ")              \
00037     ENUM_ITEM(AGVC_IK_OUT_OF_RANGE, 24, "          ")            \
00038     ENUM_ITEM(AGVC_IK_CONFIG_DISMATCH, 25, "        ")           \
00039     ENUM_ITEM(AGVC_IK_JACOBIAN_FAILED, 26, "        ")          \
00040     ENUM_ITEM(AGVC_IK_NO_SOLU, 27, "             ") \
```

```
00041     ENUM_ITEM(AGVC_IK_UNKOWN_ERROR, 28, "       ")           \
00042     ENUM_ITEM(AGVC_ERR_UNKOWN, 99999, "Unkown error occurred.")           \
00043     ENUM_ITEM(STATION_SRV_OK, 100, "    ")                  \
00044     ENUM_ITEM(STATION_SRV_NOT_FOUND, -101, "     ")          \
00045     ENUM_ITEM(STATION_CREAT_FAILED, -102, "    ")            \
00046     ENUM_ITEM(STATION_DELETE_FAILED, -103, "    ")           \
00047     ENUM_ITEM(PATH_SRV_OK, 200, "    ")                     \
00048     ENUM_ITEM(PATH_SRV_NOT_FOUND, -201, "     ")            \
00049     ENUM_ITEM(PATH_CREAT_FAILED, -202, "    ")              \
00050     ENUM_ITEM(PATH_DELETE_FAILED, -203, "    ")             \
00051     ENUM_ITEM(MAP_SRV_OK, 300, "    ")                      \
00052     ENUM_ITEM(MAP_SRV_NOT_FOUND, -301, "     ")             \
00053     ENUM_ITEM(MAP_HEADER_LIST_FAILED, -302, "   header list  ")   \
00054     ENUM_ITEM(MAP_HEADER_FAILED, -303, "   header ")           \
00055     ENUM_ITEM(MAP_SET_FAILED, -304, "    ")                   \
00056     ENUM_ITEM(MAP_SAVE_FAILED, -305, "    ")                  \
00057     ENUM_ITEM(MAP_SWITCH_FAILED, -306, "    ")                \
00058     ENUM_ITEM(MAP_DELETE_FAILED, -307, "    ")                \
00059     ENUM_ITEM(MAP_VIRTUAL_SRV_OK, 301, "    ")                \
00060     ENUM_ITEM(MAP_VIRTUAL_SRV_NOT_FOUND, -308, "     ")      \
00061     ENUM_ITEM(MAP_VIRTUAL_SET_PARAM_WRONG, -309, "     ")    \
00062     ENUM_ITEM(MAP_VIRTUAL_ADD_FAILED, -310, "     ")         \
00063     ENUM_ITEM(MAP_VIRTUAL_DELETE_FAILED, -311, "     ")       \
00064     ENUM_ITEM(IP_SRV_OK, 400, "IP    ")                     \
00065     ENUM_ITEM(IP_SRV_NOT_FOUND, -401, "ip     ")            \
00066     ENUM_ITEM(IP_NETWORK_NULL, -402, "ip     ")             \
00067     ENUM_ITEM(IP_ADDRESS_NULL, -403, "ip    ")              \
00068     ENUM_ITEM(IP_SET_FAILED, -404, "ip    ")
00069
00070 #define ENUM_ITEM(c, n, ...) c = n,
00071 enum AgvcErrorCodes : int
00072 {
00073     ENUM_AgvcErrorCodes_DECLARES
00074 };
00075
00076 /**
00077  * agv
00078  */
00079 enum class RunningMode
00080 {
00081     NONE     = 0,
00082     START    = 1, ///<
00083     MAPPING  = 2, ///<
00084     NAVIGATE = 3, ///<
00085     CHARGING = 4, ///<
00086     MAINTAIN = 5, ///<
00087     STOP     = 6, ///<
00088 };
00089
00090 /**
00091  *
00092  */
00093 enum class PathShape
00094 {
00095     NONE   = 0,
00096     LINE   = 1, ///<
00097     ARC    = 2, ///<
00098     BEZIER = 3, ///<
00099                 ///< B_SPLINE = 4  // B
00100 };
00101
00102 /**
00103  *
00104  */
00105 enum class MapFormat
00106 {
00107     NONE          = 0,
00108     OCCUPANCY_GRID = 1, ///<
00109     BASE64_PNG     = 2, ///< base64  png
00110 };
00111
00112 /**
00113  *
00114  */
00115 enum class MapVirtualAreaType
00116 {
00117     NONE      = 0,
00118     OBSTACLE  = 1, ///<
00119     SLOW_DOWN = 2, ///<
00120     INFLATE   = 3, ///<
00121 };
00122
00123 /**
00124  *
00125  */
00126 enum class MapVirtualAreaShape
00127 {
```

```
00128     NONE    = 0,
00129     LINE    = 1, ///<
00130     ARC     = 2, ///<
00131     CIRCLE  = 3, ///<
00132     POLYGON = 4, ///<
00133 };
00134
00135 /**
00136  * agv
00137  */
00138 enum class NavType
00139 {
00140     NONE           = 0, ///<
00141     FREE_TO_POSE   = 1, ///<
00142     FREE_TO_STATION = 2, ///<
00143     PATH_TO_STATION = 3, ///<
00144 };
00145
00146 /**
00147  * @brief
00148  * @since 0.7.0
00149  * @note
00150  */
00151 enum class AutoChargingType
00152 {
00153     NONE                = 0, ///<
00154     LOW_POWER_TO_BOARD     = 1, ///<
00155     HIGH_POWER_LEAVE_BOARD = 2, ///<
00156     FORCE_TO_BOARD         = 3, ///<
00157     FORCE_LEAVE_BOARD      = 4, ///<
00158 };
00159
00160 /**
00161  *
00162  */
00163 enum class FeedbackStatus
00164 {
00165     NONE     = 0, ///<
00166     FINISHED = 1, ///<
00167     FAILED   = 2, ///<
00168     RUNNING  = 3, ///<
00169     PAUSED   = 4, ///<
00170     CANCELED = 5, ///<
00171 };
00172
00173 /**
00174  *
00175  */
00176 enum class FirmwareUpdateMode
00177 {
00178     NONE         = 0, ///<
00179     UPDATE       = 1, ///<
00180     FORCED_UPDATE = 2, ///<
00181     NOT_UPDATE   = 3, ///<
00182 };
00183
00184 //
00185 enum class ScriptRuntimeState
00186 {
00187     RUNNING  = 0, //
00188     PAUSED   = 1, //
00189     STOPPED  = 2, //
00190     ABORTING = 3, //
00191 };
00192
00193 /**
00194  *
00195  */
00196 enum class CalibrationType
00197 {
00198     NONE      = 0, ///<
00199     LASER_ODOM = 1, ///<  -
00200 };
00201
00202 /**
00203  * @brief agv
00204  * @details
00205  * NONE     = 0, //  \n
00206  * FINISHED = 1, //      \n
00207  * FAILED   = 2, //      \n
00208  * RUNNING  = 3, //      \n
00209  * PAUSED   = 4, //      \n
00210  * CANCELED = 5, //
00211  */
00212 typedef FeedbackStatus NavStatus;
00213
00214 /**
```

```
00215  * @brief            saveMap
00216  * @details
00217  * NONE     = 0, //       \n
00218  * FINISHED = 1, //        \n
00219  * FAILED   = 2, //   \n
00220  * RUNNING  = 3, //    \n
00221  * PAUSED   = 4, //   \n
00222  * CANCELED = 5, //
00223  */
00224 typedef FeedbackStatus SaveMapStatus;
00225
00226 /**
00227  * @brief            switchMap
00228  * @details
00229  * NONE     = 0, //       \n
00230  * FINISHED = 1, //        \n
00231  * FAILED   = 2, //   \n
00232  * RUNNING  = 3, //    \n
00233  * PAUSED   = 4, //   \n
00234  * CANCELED = 5, //
00235  */
00236 typedef FeedbackStatus SwitchMapStatus;
00237
00238 /**
00239  * @brief             changeRunningMode
00240  * @details
00241  * NONE     = 0, //        \n
00242  * FINISHED = 1, //         \n
00243  * FAILED   = 2, //   \n
00244  * RUNNING  = 3, //     \n
00245  * PAUSED   = 4, //   \n
00246  * CANCELED = 5, //
00247  */
00248 typedef FeedbackStatus ChangeRunningModeStatus;
00249
00250 /**
00251  * @brief         relocation
00252  * @details
00253  * NONE     = 0, //      \n
00254  * FINISHED = 1, //       \n
00255  * FAILED   = 2, //   \n
00256  * RUNNING  = 3, //    \n
00257  * PAUSED   = 4, //   \n
00258  * CANCELED = 5, //
00259  */
00260 typedef FeedbackStatus RelocationStatus;
00261
00262 /**
00263  * @brief   Png           setPngMapAllInfo
00264  * @details
00265  * NONE     = 0, //       \n
00266  * FINISHED = 1, //         \n
00267  * FAILED   = 2, //   \n
00268  * RUNNING  = 3, //     \n
00269  * PAUSED   = 4, //   \n
00270  * CANCELED = 5, //
00271  */
00272 typedef FeedbackStatus SetPngMapAllInfoStatus;
00273
00274 /**
00275  * @brief   Png           getPngMapAllInfo
00276  * @details
00277  * NONE     = 0, //       \n
00278  * FINISHED = 1, //         \n
00279  * FAILED   = 2, //   \n
00280  * RUNNING  = 3, //     \n
00281  * PAUSED   = 4, //   \n
00282  * CANCELED = 5, //
00283  */
00284 typedef FeedbackStatus GetPngMapAllInfoStatus;
00285
00286 /**
00287  * @brief            getGridMapFromAgv
00288  * @details
00289  * NONE     = 0, //       \n
00290  * FINISHED = 1, //         \n
00291  * FAILED   = 2, //   \n
00292  * RUNNING  = 3, //     \n
00293  * PAUSED   = 4, //   \n
00294  * CANCELED = 5, //
00295  */
00296 typedef FeedbackStatus GetGridMapStatus;
00297
00298 /**
00299  * @brief            sendGridMapToAgv
00300  * @details
00301  * NONE     = 0, //       \n
```

```
00302  * FINISHED = 1, //         \n
00303  * FAILED   = 2, //   \n
00304  * RUNNING  = 3, //      \n
00305  * PAUSED   = 4, //   \n
00306  * CANCELED = 5, //
00307  */
00308 typedef FeedbackStatus SendGridMapStatus;
00309
00310 /**
00311  * @brief  png        getBase64PngMapFromAgv
00312  * @details
00313  * NONE     = 0, //  png     \n
00314  * FINISHED = 1, //  png      \n
00315  * FAILED   = 2, //   \n
00316  * RUNNING  = 3, //  png  \n
00317  * PAUSED   = 4, //   \n
00318  * CANCELED = 5, //
00319  */
00320 typedef FeedbackStatus GetPngMapStatus;
00321
00322 /**
00323  * @brief  png        sendBase64PngMapToAgv
00324  * @details
00325  * NONE     = 0, //  png     \n
00326  * FINISHED = 1, //  png      \n
00327  * FAILED   = 2, //   \n
00328  * RUNNING  = 3, // png   \n
00329  * PAUSED   = 4, //   \n
00330  * CANCELED = 5, //
00331  */
00332 typedef FeedbackStatus SendPngMapStatus;
00333
00334 /**
00335  * @brief              setGridMapAllInfo
00336  * @details
00337  * NONE     = 0, //          \n
00338  * FINISHED = 1, //           \n
00339  * FAILED   = 2, //   \n
00340  * RUNNING  = 3, //      \n
00341  * PAUSED   = 4, //   \n
00342  * CANCELED = 5, //
00343  */
00344 typedef FeedbackStatus SetGridMapAllInfoStatus;
00345
00346 /**
00347  * @brief              getGridMapAllInfo
00348  * @details
00349  * NONE     = 0, //          \n
00350  * FINISHED = 1, //           \n
00351  * FAILED   = 2, //   \n
00352  * RUNNING  = 3, //         \n
00353  * PAUSED   = 4, //   \n
00354  * CANCELED = 5, //
00355  */
00356 typedef FeedbackStatus GetGridMapAllInfoStatus;
00357
00358 /**
00359  * @brief              previewPngMapFromAgv
00360  * @details
00361  * NONE     = 0, //       \n
00362  * FINISHED = 1, //        \n
00363  * FAILED   = 2, //   \n
00364  * RUNNING  = 3, //       \n
00365  * PAUSED   = 4, //   \n
00366  * CANCELED = 5, //
00367  */
00368 typedef FeedbackStatus PreviewPngMapStatus;
00369
00370 /**
00371  * @brief           autoAlignRailway
00372  * @details
00373  * NONE     = 0, //       \n
00374  * FINISHED = 1, //        \n
00375  * FAILED   = 2, //   \n
00376  * RUNNING  = 3, //       \n
00377  * PAUSED   = 4, //   \n
00378  * CANCELED = 5, //
00379  */
00380 typedef FeedbackStatus AutoAlignRailwayStatus;
00381
00382 /**
00383  * @brief
00384  * @details
00385  * NONE     = 0, // \n
00386  * FINISHED = 1, //   \n
00387  * FAILED   = 2, //   \n
00388  * RUNNING  = 3, //     \n
```

```
00389  * PAUSED   = 4, //   \n
00390  * CANCELED = 5, //
00391  */
00392 typedef FeedbackStatus CalibrationStatus;
00393
00394 /**
00395  *
00396  */
00397 struct Point2d
00398 {
00399     float x; ///<    m
00400     float y; ///<    m
00401 };
00402
00403 /**
00404  *
00405  */
00406 struct Pose2d
00407 {
00408     double x;   ///<    m
00409     double y;   ///<    m
00410     double yaw; ///<    rad
00411 };
00412
00413 /**
00414  *
00415  */
00416 struct Speed
00417 {
00418     double v; ///<    m/s
00419     double w; ///<    rad/s
00420 };
00421
00422 /**
00423  *
00424  */
00425 struct Header
00426 {
00427     std::string id;      ///< uuid
00428     std::string name;    ///<
00429     int64_t stamp;       ///<    UTC  (1970 01 01 00 00 00 ~ )
00430     std::string map_id;  ///<     id,  uuid  a5e7c6de-4463-443e-8b25-ccbfd6a27aee,
00431                          ///<
00432     int error_code = -1; ///<
00433 };
00434
00435 /**
00436  * agv
00437  */
00438 struct AgvDetails
00439 {
00440     std::string name;                ///< agv
00441     std::string version;             ///<
00442     std::string MFD;                 ///<
00443     std::string SN;                  ///< S/N
00444     int communication_version;       ///< bridge
00445     int battery_version;             ///<
00446     int loop_times;                  ///<
00447     float surplus_capacity;          ///<
00448     int left_motor_firmware_version;   ///<
00449     int right_motor_firmware_version;  ///<
00450     int master_board_firmware_version; ///<
00451     int hardware_abstraction_version;  ///<
00452     int error_code_version;          ///<
00453     float max_speed;                 ///< agv
00454     float max_angular;               ///< agv
00455 };
00456
00457 /**
00458  * agv
00459  */
00460 struct RunningInfo
00461 {
00462     Header header;              ///< agv SN  agv         id
00463     RunningMode running_mode;   ///< agv
00464     Speed current_speed;        ///<      {v w}   {m/s rad/s}
00465     int8_t location_score;      ///<      [0-100]
00466     double total_dist;          ///<       m
00467     double current_dist;        ///<         m
00468     double total_running_time;  ///<        s
00469     double current_running_time; ///<        s
00470     double battery_voltage;     ///<       V
00471     double battery_current;     ///<       A
00472     int8_t remaining_voltage;   ///<        % [0-100]
00473     int8_t battery_status;      ///<    0- ,1-  2-  3-  4-
00474     int8_t safety_edge_status;  ///<      0- ,1-    2-    3-    4-
00475     bool low_battery;           ///<
```

```
00476     bool pause;                 ///<
00477     bool emergency_stop;          ///<
00478     bool blocked;               ///<
00479     int8_t obstacle_level;        ///<    0-  1-   2-    3-
00480     bool localization_loss;       ///<
00481     bool fault_alarm;             ///<              ,
00482     bool break_release_flag;      ///<      true    false
00483 };
00484
00485 /**
00486  *  saveMap
00487  */
00488 struct SaveMapWorkingStatus
00489 {
00490     Header header;        ///< header.id    saveMap  id
00491     SaveMapStatus status; ///<  SaveMap    NONE -    RUNNING-  FINISH-
00492 };
00493
00494 /**
00495  *  switchMap
00496  */
00497 struct SwitchMapWorkingStatus
00498 {
00499     Header header;          ///< header.id    switchMap  id
00500     SwitchMapStatus status; ///<  switchMap    NONE -    RUNNING-  FINISH-
00501 };
00502
00503 /**
00504  *  changeRunningMode
00505  */
00506 struct ChangeModeWorkingStatus
00507 {
00508     Header header;                ///< header.id    changeRunningMode  id
00509     ChangeRunningModeStatus status; ///<  changeRunningMode
00510                                   ///< NONE -    RUNNING-  FINISH-
00511 };
00512
00513 /**
00514  *  Relocation
00515  */
00516 struct RelocationWorkingStatus
00517 {
00518     Header header;          ///< header.id    Relocation  id
00519     RelocationStatus status; ///<  Relocation    NONE -    RUNNING-  FINISH-
00520 };
00521
00522 /**
00523  *  setPngMapAllInfo
00524  */
00525 struct SetPngMapAllInfoWorkingStatus
00526 {
00527     Header header;                ///< header.id    setPngMapAllInfo  id
00528     SetPngMapAllInfoStatus status; ///<  setPngMapAllInfo    NONE -    RUNNING-  FINISH-
00529 };
00530
00531 /**
00532  *  getPngMapAllInfo
00533  */
00534 struct GetPngMapAllInfoWorkingStatus
00535 {
00536     Header header;                ///< header.id    getPngMapAllInfo  id
00537     GetPngMapAllInfoStatus status; ///<  getPngMapAllInfo    NONE -    RUNNING-  FINISH-
00538 };
00539
00540 /**
00541  *  getGridMapFromAgv
00542  */
00543 struct GetGridMapWorkingStatus
00544 {
00545     Header header;          ///< header.id    getGridMapFromAgv  id
00546     GetGridMapStatus status; ///<  getGridMapFromAgv    NONE -    RUNNING-  FINISH-
00547 };
00548
00549 /**
00550  *  sendGridMapToAgv
00551  */
00552 struct SendGridMapWorkingStatus
00553 {
00554     Header header;          ///< header.id    sendGridMapToAgv  id
00555     SendGridMapStatus status; ///<  sendGridMapToAgv    NONE -    RUNNING-  FINISH-
00556 };
00557
00558 struct GetPngMapWorkingStatus
00559 {
00560     Header header;          ///< header.id    getBase64PngMapFromAgv  id
00561     GetPngMapStatus status; ///<  getBase64PngMapFromAgv    NONE -    RUNNING-  FINISH-
00562 };
```

```
00563
00564 /**
00565  *  sendBase64PngMapToAgv
00566  */
00567 struct SendPngMapWorkingStatus
00568 {
00569     Header header;          ///< header.id     sendBase64PngMapToAgv  id
00570     SendPngMapStatus status; ///<  sendBase64PngMapToAgv     NONE -     RUNNING-   FINISH-
00571 };
00572
00573 /**
00574  *  setGridMapAllInfo
00575  */
00576 struct SetGridMapAllInfoWorkingStatus
00577 {
00578     Header header;                ///< header.id     setGridMapAllInfo  id
00579     SetGridMapAllInfoStatus status; ///<  setGridMapAllInfo
00580                                   ///< NONE -     RUNNING-   FINISH-
00581 };
00582
00583 /**
00584  *  getGridMapAllInfo
00585  */
00586 struct GetGridMapAllInfoWorkingStatus
00587 {
00588     Header header;                ///< header.id     getGridMapAllInfo  id
00589     GetGridMapAllInfoStatus status; ///<  getGridMapAllInfo
00590                                   ///< NONE -     RUNNING-   FINISH-
00591 };
00592
00593 /**
00594  *  previewPngMapFromAgv
00595  */
00596 struct PreviewPngMapWorkingStatus
00597 {
00598     Header header;            ///< header.id     previewPngMapFromAgv  id
00599     PreviewPngMapStatus status; ///<  previewPngMapFromAgv
00600                               ///< NONE -     RUNNING-   FINISH-
00601 };
00602
00603 /**
00604  *  autoAlignRailway
00605  */
00606 struct AutoAlignRailwayWorkingStatus
00607 {
00608     Header header;                ///< header.id     autoAlignRailway  id
00609     AutoAlignRailwayStatus status; ///<  autoAlignRailway
00610                                   ///< NONE -     RUNNING-   FINISH-
00611 };
00612
00613 /**
00614  *
00615  *    14     saveMap, switchMap, changeRunningMode, relocation, sendBase64PngMapToAgv
00616  * getGridMapAllInfo, setGridMapAllInfo, sendGridMapToAgv, getGridMapFromAgv
00617  * getPngMapAllInfo, setPngMapAllInfo, getBase64PngMapFromAgv, previewPngMapFromAgv, autoAlignRailway
00618  */
00619 struct AsyncInterfaceResultStatus
00620 {
00621     Header header;                                  ///< agv SN  agv         id
00622     SaveMapWorkingStatus save_map_status;             ///<    (saveMap)
00623     SwitchMapWorkingStatus switch_map_status;          ///<    (switchMap)
00624     ChangeModeWorkingStatus change_running_mode_status; ///<      (changeRunningMode)
00625     RelocationWorkingStatus relocation_status;         ///<    (relocation)
00626     GetGridMapWorkingStatus get_grid_map_status;       ///<      (getGridMapFromAgv)
00627     SendGridMapWorkingStatus send_grid_map_status;      ///<      (sendGridMapToAgv)
00628     GetPngMapWorkingStatus get_png_map_status;          ///<   png  (getBase64PngMapFromAgv)
00629     SendPngMapWorkingStatus send_png_map_status;        ///<   png  (sendBase64PngMapToAgv)
00630     SetPngMapAllInfoWorkingStatus set_png_all_status;   ///<   png     (setPngMapAllInfo)
00631     GetPngMapAllInfoWorkingStatus get_png_all_status;   ///<   png     (getPngMapAllInfo)
00632     SetGridMapAllInfoWorkingStatus set_grid_all_status; ///<         (setGridMapAllInfo)
00633     GetGridMapAllInfoWorkingStatus get_grid_all_status; ///<         (getGridMapAllInfo)
00634     PreviewPngMapWorkingStatus preview_png_map_status;  ///<   png  (previewPngMapFromAgv)
00635     AutoAlignRailwayWorkingStatus align_railway_status; ///<    (autoAlignRailway)
00636 };
00637
00638 /**
00639  *
00640  */
00641 struct StationMark
00642 {
00643     Header header;    ///<  id            id
00644     std::string type; ///<     "ST",   "CP"
00645     Pose2d pose;      ///<     {x(m) y(m) theta(rad)}
00646 };
00647
00648 /**
00649  *
```

```
00650 */
00651 struct PathStation
00652 {
00653     Header header;                    ///<  id            id
00654     PathShape shape;                   ///<
00655     bool use_direction;                ///<
00656     double max_speed;                  ///< agv
00657     std::string start_station_id;      ///<     id
00658     std::string end_station_id;        ///<     id
00659     std::vector<Point2d> control_points; ///<      {{x,y},{}, ...}
00660 };
00661
00662 /**
00663 *
00664 */
00665 struct MapInfo
00666 {
00667     Header header;           ///<  id       map_id    id
00668     double resolution;       ///<        /       m
00669     uint32_t width;          ///<        /
00670     uint32_t height;         ///<        /
00671     Pose2d origin;           ///< {x(m) y(m) theta(rad)}
00672                             ///<
00673                             ///<
00674     MapFormat format;        ///<
00675     std::vector<int8_t> data; ///<
00676 };
00677
00678 /**
00679 *  @brief
00680 *  @details
00681 *  Header header;           //  id       map_id    id\n
00682 *  double resolution;       //            m\n
00683 *  uint32_t width;          //        \n
00684 *  uint32_t height;         //        \n
00685 *  Pose2d origin;           // {x(m) y(m) theta(rad)}
00686 *                          //
00687 *                          //           \n
00688 *  MapFormat format;        //    =MapFormat::OCCUPANCY_GRID\n
00689 *  std::vector<int8_t> data; //     0     100     -1
00690 */
00691 typedef MapInfo OccupancyGridMap;
00692
00693 /**
00694 *  @brief Base64  png
00695 *  @details
00696 *  Header header;           //  id       map_id    id\n
00697 *  double resolution;       //            m\n
00698 *  uint32_t width;          //        \n
00699 *  uint32_t height;         //        \n
00700 *  Pose2d origin;           // {x(m) y(m) theta(rad)}
00701 *                          //
00702 *                          //          \n
00703 *  MapFormat format;        //    =MapFormat::BASE64_PNG\n
00704 *  std::vector<int8_t> data; //     Base64  png     data    string
00705 */
00706 typedef MapInfo Base64PngMap;
00707
00708 /**
00709 *
00710 */
00711 struct MapVirtualArea
00712 {
00713     Header header;             ///<    id              id
00714     MapVirtualAreaType type;    ///<
00715     MapVirtualAreaShape shape;  ///<
00716     double width_speed;        ///<    /   /
00717     std::vector<Point2d> vertex; ///<      {{x(m) y(m)},{}, ...}
00718                             ///<              vertex
00719                             ///<
00720                             ///<             ( 3 )
00721 };
00722
00723 /**
00724 *
00725 */
00726 struct MapAllInfo
00727 {
00728     MapInfo map;                        ///<
00729     std::vector<StationMark> stations;       ///<
00730     std::vector<PathStation> paths;          ///<
00731     std::vector<MapVirtualArea> virtual_areas; ///<
00732 };
00733
00734 /**
00735 * IP
00736 */
```

```
00737 struct IpAddressInfo
00738 {
00739     std::string name;         ///<
00740     std::string type;         ///<
00741     std::string device;       ///<
00742     std::string ipv4_ip;      ///< IPv4  CIDR
00743     std::string ipv4_gateway; ///<
00744     std::string ipv4_method;  ///<            auto, manual
00745     std::string ipv4_dns;     ///<   DNS
00746     std::string ipv6_ip;      ///< IPv6  CIDR
00747     std::string ipv6_gateway;
00748 };
00749 /**
00750  * agv
00751  */
00752 struct AdjustParam
00753 {
00754     bool enable;         ///<      agv  true-  false
00755     bool forward_flag;   ///< agv    true      false
00756     double max_distance; ///< agv
00757     double max_angle;    ///< agv
00758 };
00759 /**
00760  *  agv
00761  */
00762 struct NavGoalType
00763 {
00764     Header header;                      ///<  id         id
00765     NavType type;                       ///<
00766     Pose2d target_pose;                 ///<     {x(m) y(m) theta(rad)}
00767                                         ///<
00768     std::vector<std::string> stations_id; ///<      id,
00769                                         ///<
00770     bool forward_flag;                  ///< agv
00771     bool goal_end_rotate;               ///<            true
00772                                         ///< false
00773     AdjustParam secondary_adjustment;     ///<               , true
00774                                         ///< false
00775     Speed max_vel;                      ///<    v,w
00776 };
00777
00778 /**
00779  * @brief       /
00780  * @details
00781  *          \n
00782  *             type current_trajectory
00783  *  */
00784 struct NavInfo
00785 {
00786     Header header;    ///<  id         id
00787     NavStatus status; ///<
00788     [[deprecated("Since 0.7.0: use nav_type instead.")]]
00789     NavType type;     ///< @if Chinese @deprecated  0.7.0      nav_type    @endif @if English @deprecated Deprecated
        since 0.7.0. Use nav_type instead. @endif
00790     NavType nav_type;                   ///<
00791     AutoChargingType auto_charging_type;     ///<
00792     StationMark goal_station;           ///< agv
00793     std::vector<Point2d> current_trajectory; ///<       {{x(m) y(m)},{}, ...}
00794 };
00795
00796 /**
00797  * @brief
00798  * @since 0.7.0
00799  */
00800 struct AutoChargingCommand
00801 {
00802     Header header;                      ///<  id         id
00803     AutoChargingType auto_charging_type; ///<
00804     std::string cp_station_id;          ///<  id
00805     std::string leave_to_station_id;    ///<  id
00806     Speed max_speed;                    ///<
00807 };
00808
00809 /**
00810  *
00811  */
00812 struct FirmwareUpdateParam
00813 {
00814     FirmwareUpdateMode left_motor;   ///<
00815     FirmwareUpdateMode right_motor;  ///<
00816     FirmwareUpdateMode master_board; ///<
00817     std::string firmware_file_path;  ///<         "/root/update.firm"
00818 };
00819
00820 /**
00821  *
00822  */
```

```
00823 struct FirmwareUpdateProcessInfo
00824 {
00825     std::string update_info; ///<          failed     ,
00826                              ///< none
00827     double update_progress;  ///<        1.0
00828 };
00829
00830 struct CalibrationProcessInfo
00831 {
00832     Header header;               ///<  id         id
00833     CalibrationType type;          ///<
00834     CalibrationStatus status;      ///<
00835     uint32_t collected_data_count; ///<
00836 };
00837
00838 /**
00839  * RTDE
00840  */
00841 struct RtdeRecipe
00842 {
00843     bool to_server;                ///<   /
00844     int chanel;                   ///<
00845     double frequency;               ///<
00846     int trigger;                  ///<    (     ): 0 -   ; 1 -
00847     std::vector<std::string> segments; ///<
00848 };
00849
00850 enum error_type
00851 {
00852     parse_error      = -32700, ///<
00853     invalid_request  = -32600, ///<
00854     method_not_found = -32601, ///<
00855     invalid_params   = -32602, ///<
00856     internal_error   = -32603, ///<
00857     server_error,              ///<
00858     invalid                   ///<
00859 };
00860
00861 enum ExceptionCode
00862 {
00863     EC_DISCONNECTED     = -1,  ///<
00864     EC_NOT_LOGINED      = -2,  ///<
00865     EC_INVAL_SOCKET     = -3,  ///<
00866     EC_REQUEST_BUSY     = -4,  ///<
00867     EC_SEND_FAILED      = -5,  ///<
00868     EC_RECV_TIMEOUT     = -6,  ///<
00869     EC_RECV_ERROR       = -7,  ///<
00870     EC_PARSE_ERROR      = -8,  ///<
00871     EC_INVALID_REQUEST  = -9,  ///<
00872     EC_METHOD_NOT_FOUND = -10, ///<
00873     EC_INVALID_PARAMS   = -11, ///<
00874     EC_INTERNAL_ERROR   = -12, ///<
00875     EC_SERVER_ERROR     = -13, ///<
00876     EC_INVALID          = -14  ///<
00877 };
00878
00879 class AgvcException : public std::exception
00880 {
00881 public:
00882     AgvcException(int code, const std::string &prefix, const std::string &message) noexcept
00883         : code_(code), message_(prefix + "-" + message)
00884     {
00885     }
00886
00887     AgvcException(int code, const std::string &message) noexcept : code_(code), message_(message) {}
00888
00889     error_type type() const
00890     {
00891         if (code_ >= -32603 && code_ <= -32600) {
00892             return static_cast<error_type>(code_);
00893         } else if (code_ >= -32099 && code_ <= -32000) {
00894             return server_error;
00895         } else if (code_ == -32700) {
00896             return parse_error;
00897         }
00898         return invalid;
00899     }
00900
00901     int code() const { return code_; }
00902     const char *what() const noexcept override { return message_.c_str(); }
00903
00904 private:
00905     int code_;
00906     std::string message_;
00907 };
00908
00909 inline const char *returnValue2Str(int retval)
```

```
00910 {
00911     static const char *retval_str[] = {
00912 #define ENUM_ITEM(n, v, s) s,
00913         ENUM_AgvcErrorCodes_DECLARES
00914 #undef ENUM_ITEM
00915     };
00916 
00917     enum agvc_index
00918     {
00919 #define ENUM_ITEM(n, v, s) n##_INDEX,
00920         ENUM_AgvcErrorCodes_DECLARES
00921 #undef ENUM_ITEM
00922     };
00923 
00924     int index = -1;
00925 
00926 #define ENUM_ITEM(n, v, s) \
00927     if (retval == v)       \
00928         index = n##_INDEX;
00929     ENUM_AgvcErrorCodes_DECLARES
00930 #undef ENUM_ITEM
00931 
00932         if (index == -1)
00933     {
00934         index = AGVC_ERR_UNKOWN;
00935     }
00936 
00937     return retval_str[(unsigned)index];
00938 }
00939 
00940 } // namespace agvc_interface
00941 
00942 #if defined ENABLE_JSON_TYPES
00943 #include "bindings/jsonrpc/json_types.h"
00944 #endif
00945 
00946 #endif
```

Index