

# ROBIL-2

## *Interface Control Document*

*( ICD )*

Version: 0.83

**Produced by:**

**CogniTeam Ltd.**

**Shoham 7 St, Petah Tikva, Israel**

## Revision History

Date	Version	Description	Author
12/02/2014	0.1	Initial draft	Dan Erusalimchik
16/02/2014	0.2	All diagnostic messages and events messages taken together.	Dan Erusalimchik
18/02/2014	0.3	Update	Dan Erusalimchik Eliya Shaviv (IAI)
18/02/2014	0.4	Components Acronym description	Ari Yakir
25/02/2014	0.5	Update	Dan Erusalimchik
27/02/2014	0.6	Assign Manipulator Task is updated. BaldePose type is updated.	Eliya Shaviv (IAI) Dan Erusalimchik
09/03/2014	0.7	MultiLaserScan added	Dan Erusalimchik
27/04/2014	0.7.1	/SENSORS/SICK/2 added	Dan Erusalimchik Oded Yahiel (BGU)
13/05/2014	0.7.2	update of events for PP, WPD, LLC	Dan Erusalimchik
13/05/2014	0.8	Transformations : - Tree - Frames: world, odom, base_link, map, min-map	Dan Erusalimchik
15/05/2014	0.81	Transformations - body subtree	Eliya Shaviv (IAI)
20/05/2014	0.82	Transformations - base_arm subtree	Eliya Shaviv (IAI)
01/06/2014	0.83	update of WSM Velocity message	Dan Erusalimchik

# Table of Contents

## 1. Introduction

Interface Control Document Objectives

Acronyms used in the document

## 2. Messages

Components Control Events

Components Diagnostic Messages

IED Detection Event

Custom IED Object

Position Update

OCU Teleoperation Control

Assign Navigation Task

Assign Manipulator Task

Assign Mission

Set / Get ABL

Global Path

Set / Get Max Velocity

Work Sequence Data

Mission Acceptance

Local Path

WPD Velocity

WSM Velocity

GPS/INS data

Blade Position

Blade Position Command

Map

Mini-Map

VO Estimation

Percepted Velocity

Location

Sensor INS

Sensor GPS

Sensor Camera

Sensor Wire

Sensor Laser SICK

Sensor Laser IBEO

Platform Efforts

Heartbeat

## 3. Components Diagnostic Messages

Decision Making Control Flow

Decision Making

OCU

SMME

Mission Progress

ABL Reactions

[SSM](#)  
[IEDSIM](#)  
[WSM](#)  
[PP](#)  
[WPD](#)  
[PER](#)  
[LOC](#)  
[LLC](#)

#### [4. Components Control Events](#)

[/Teleoperation](#)  
[/Autonomy](#)  
[SMME Mission and Component control](#)  
[/E-Stop](#)  
[WPD Component control](#)  
[PP Component control](#)  
[LLC Component control](#)  
[WSM Component control](#)  
[Monitoring Events](#)  
[/IEDDetected](#)  
[ABL Events](#)  
[/NoPathFound](#)  
[/CommFail](#)  
[/ObstacleDetected](#)  
[/RoadDetected](#)  
[/Turn-over](#)  
[/Collision](#)  
[/NoGPS](#)  
[/AssistanceRequired](#)

#### [5. Types](#)

[std\\_msgs](#)  
[std\\_msgs/String](#)  
[std\\_msgs/Int32](#)  
[std\\_msgs/Float64](#)  
[std\\_msgs/Header](#)  
[sensor\\_msgs](#)  
[sensor\\_msgs/NavSatFix](#)  
[sensor\\_msgs/Imu](#)  
[sensor\\_msgs/LaserScan](#)  
[sensor\\_msgs/Image](#)  
[sensor\\_msgs/JointState](#)  
[geometry\\_msgs](#)  
[geometry\\_msgs/PoseStamped](#)  
[geometry\\_msgs/TwistStamped](#)  
[geometry\\_msgs/PoseWithCovariance](#)  
[geometry\\_msgs/TwistWithCovariance](#)  
[geometry\\_msgs/PoseWithCovarianceStamped](#)

## Nav msgs

nav msgs/Path waypoints

nav msgs/Odometry

## diagnostic msgs

diagnostic msgs/DiagnosticStatus

## robin msgs

robin msgs/AssignNavTask

robin msgs/AssignManipulatorTask

robin msgs/AssignMission

robin msgs/Path

robin msgs/MissionAcceptance

robin msgs/IEDLocation

robin msgs/Map

robin msgs/MultiLaserScan

## 6. Transformations (TF)

### Transformations Tree

### Coordinate Frames Specification

world

odom

base link

occupancy map

mini-map

body

Ibeo

cameraL frame

cameraR frame

front sick

gps ins

ibeo

left sick

right sick

base arm

main arm

loader

brackets

# 1. Introduction

This section introduces the ICD to the reader.

## Interface Control Document Objectives

This ICD describes the interface between ROBIL components.

## Acronyms used in the document

- Components
  - SMME - System Management and Mission Execution
  - SSM - System and Safety Monitoring
  - LLC - Low Level Controller
  - PP - Path Planning
  - WPD - Way Point Driver
  - WSM - Work Sequence Management (of the loader)
  - PER - Perception
  - IEDSIM - IED (Improvised Explosive Device) Simulation
  - OCU - Operator Control Unit
  - LOC - Localization
- General terms
  - ABL - Autonomous Behavior level ( autonomy reaction configuration)

## 2. Messages

This section describes the messages, types and data flow.

### Components Control Events

Events to/from FSMs control component activity

**From:** IEDSIM, LLC, OCU, PER, PP, SMME, SSM, WPD, WSM

**To:** IEDSIM, LLC, OCU, PER, PP, SMME, SSM, WPD, WSM

**Topic name:** /decision\_making/events

**Message Type:** std\_msgs/String

**See:** Section 4

### Components Diagnostic Messages

Notification about component running status changes.

**From:** IEDSIM, LLC, OCU, PER, PP, SMME, SSM, WPD, WSM

**To:** SSM

**Topic name:** /diagnostics

**Message Type:** diagnostic\_msgs/DiagnosticStatus

**Inner Struct:** each component defines relevant key-value records.

**See:** Section 3

### IED Detection Event

Notification about IED detection.

**From:** IEDSIM

**To:** SMME, OCU

**Topic name:** /IED/Location

**Message Type:** robil\_msgs/IEDLocation

### Custom IED Object

Custom definition of IED object location.

**From:** OCU

**To:** IEDSIM

**Topic name:** /OCU/IED/Location

**Message Type:** robil\_msgs/IEDLocation

### Position Update

Correction of robot localization

**From:** OCU

**To:** LOC

**Topic name:** /OCU/PositionUpdate

**Message Type:** geometry\_msgs/PoseStamped

## OCU Teleoperation Control

Teleoperation control

**From:** OCU

**To:** Platform

**Over:** non ROS communication

## Assign Navigation Task

Add navigation task to tasks collection.

**From:** OCU

**To:** SMME

**Topic name:** /OCU/SMME/NavigationTask

**Message Type:** robil\_msgs/AssignNavTask

## Assign Manipulator Task

Add manipulator task to tasks collection

**From:** OCU

**To:** SMME

**Topic name:** /OCU/SMME/ManipulationTask

**Message Type:** robil\_msgs/AssignManipulatorTask

## Assign Mission

Add mission to missions collection

**From:** OCU

**To:** SMME

**Topic name:** /OCU/SMME/MissionPlan

**Message Type:** robil\_msgs/AssignMission

## Set / Get ABL

Define policy of reactions on navigation problems.

**From:** OCU

**To:** SMME

**Parameter Server:** /ABL/Events/\*

**Key:** event name (trigger) as String

**Value:** reaction behavior ID as String

## Global Path

Global Path from current robot state to final goal.

May be, it's defined in low resolution and without obstacle avoidance.

**From:** SMME

**To:** PP

**Topic name:** /SMME/GlobalPath

**Message Type:** robil\_msgs/Path



## Set / Get Max Velocity

Limitations of speed

**From:** SMME

**To:** PP, WPD

**Parameter Server:**

/NAVIGATION/MaxSpeed/Linear/x as Double [m/sec]

/NAVIGATION/MaxSpeed/Linear/y as Double [m/sec]

/NAVIGATION/MaxSpeed/Linear/z as Double [m/sec]

/NAVIGATION/MaxSpeed/Angular/x as Double [rad/sec]

/NAVIGATION/MaxSpeed/Angular/y as Double [rad/sec]

/NAVIGATION/MaxSpeed/Angular/z as Double [rad/sec]

## Work Sequence Data

Work sequence task data.

**From:** SMME

**To:** WSM

**Topic name:** /SMME/WSM/Task

**Message Type:** robil\_msgs/AssignManipulatorTask

## Mission Acceptance

Notification about mission acceptance or rejection.

**From:** SMME

**To:** OCU

**Topic name:** /SMME/OCU/MissionAcceptance

**Message Type:** robil\_msgs/MissionAcceptance

## Local Path

Local path plan. Defined with normal resolution and with static obstacles avoidance.

**From:** PP

**To:** WPD, OCU

**Topic name:** /PP/Path

**Message Type:** robil\_msgs/Path

## WPD Velocity

Steering

**From:** WPD

**To:** LLC

**Topic name:** /WPD/Speed

**Message Type:** geometry\_msgs/TwistStamped

## WSM Velocity

Steering

**From:** WSM  
**To:** LLC  
**Topic name:** /WSM/Speed  
**Message Type:** geometry\_msgs/TwistStamped

## GPS/INS data

GPS and INS data  
**From:** PER  
**To:** LOC  
**Topic name:**  
1. /PER/GPS  
2. /PER/INS  
**Message Type:**  
1. sensor\_msgs/NavSatFix - for GPS data  
2. sensor\_msgs/Imu - for INS data

## Blade Position

Position of robot's blade  
**From:** PER  
**To:** WSM, SSM, PP, SMME, OCU  
**Topic name:** /PER/BladPosition  
**Message Type:** sensor\_msgs/JointState

## Blade Position Command

Position of robot's blade  
**From:** WSM  
**To:** LLC  
**Topic name:** /WSM/BladePosition  
**Message Type:** sensor\_msgs/JointState

## Map

Occupancy Grid with altitude and tags on each cell.  
**From:** PER  
**To:** PP, OCU  
**Topic name:** /PER/Map  
**Message Type:** robil\_msgs/Map

## Mini-Map

Occupancy Grid with altitude and tags on each cell.  
**From:** PER  
**To:** WPD  
**Topic name:** /PER/MiniMap  
**Message Type:** robil\_msgs/Map

## VO Estimation

Visual odometry estimation of localization pose.

**From:** PER

**To:** LOC

**Topic name:** /PER/VO

**Message Type:** nav\_msgs/Odometry

## Perceived Velocity

Robot speed perception

**From:** LOC

**To:** PER, SSM, OCU, LLC

**Topic name:** /LOC/Velocity

**Message Type:** geometry\_msgs/TwistStamped

## Location

Robot location perception

**From:** LOC

**To:** PER, SSM, OCU, IEDSIM, SMME, WSM, LLC, WPD

**Topic name:** /LOC/Pose

**Message Type:** geometry\_msgs/PoseWithCovarianceStamped

## Sensor INS

IMU raw data

**From:** SENSORS

**To:** PER

**Topic name:** /SENSORS/IMU

**Message Type:** sensor\_msgs/Imu

## Sensor GPS

GPS raw data

**From:** SENSORS

**To:** PER

**Topic name:** /SENSORS/GPS

**Message Type:** sensor\_msgs/NavSatFix

## Sensor Camera

Stream of frames from camera

**From:** SENSORS

**To:** PER

**Topic name:**

/SENSORS/CAM/R -- right camera

/SENSORS/CAM/L -- left camera

**Message Type:** sensor\_msgs/Image

## Sensor Wire

Wire length

**From:** SENSORS

**To:** PER

**Topic name:** /SENSORS/WIRE

**Message Type:** std\_msgs/Float64

**Units:** meters

## Sensor Laser SICK

SICK laser raw data

**From:** SENSORS

**To:** PER

**Topic name:**

/SENSORS/SICK/1

/SENSORS/SICK/2

**Message Type:** sensor\_msgs/LaserScan

## Sensor Laser IBEO

IBEO laser raw data, divided by levels of scans.

**From:** SENSORS

**To:** PER

**Topic name:** /SENSORS/IBEO/1

**Message Type:** robil\_msgs/MultiLaserScan

## Platform Efforts

Efforts of platform actuators

**From:** LLC

**To:** PLATFORM , PER

**Topic name:**

/LLC/EFFORTS/Throttle

/LLC/EFFORTS/Steering

/LLC/EFFORTS/Joints

**Message Type:** std\_msgs/Int32

**Units:** [-100%,100%]

## Heartbeat

Operation status notification

**From:** IEDSIM, LLC, OCU, PER, PP, SMME, SSM, WPD, WSM

**To:** SSM

**Topic name:** /heartbeat

**Message Type:** std\_msgs/String

**Data:** name of component

### 3. Components Diagnostic Messages

This section describes key-value structs of DiagnosticsStatus.

For all diagnostics messages field “*message*” is a short description of reported change and field “*name*” ( not a part of key-value list ) is a message type.

#### Decision Making Control Flow

these are general diagnostic messages for all components and all logic controllers. In other words, all FSMs of all components report their states using this format.

##### Decision Making

Key	Value	Type
name	full name of element (id), contains name of component and FSM state.	string
type	type of change : started/stopped	string
status	in case of finish, result Success/Failure	string
node_name	name of controlled node	string
node_exe_file	executable command	string
node_exe_dir	directory of executable file	string
node_run_dir	unique id of execution. It's changed when node restarts.	string

#### OCU

No diagnostic messages

#### SMME

##### Mission Progress

Key	Value	Type
mission	Id of mission	string
task	Id of task	string
type	type of change : started/finished	string
status	cause of stopping : Success/Failure	string

## ABL Reactions

Key	Value	Type
event	id of event	string
reaction	Id of reaction	string
type	type of report : started/finished	string

## SSM

### Platform Status

Key	Value	Type
status	status of platform : OK,FAIL,etc	string
???	???	???

### Software Status

Key	Value	Type
component_id	ID of component	???
status	status of component	string
?????	?????	?????

### Sensors Status

Key	Value	Type
sensor_id	ID of sensor	???
status	status of sensor	string
confidence	confidence of sensor data	integer [0-100%]
?????	?????	?????

## IEDSIM

No diagnostic messages

## WSM

### WSM Progress Report

Key	Value	Type
step_id	ID of step	string
status	Started / Timeout / Success / Pause	string

## PP

### Path Planner

Key	Value	Type
start	Current position of robot	string
goal	Goal for planning	string
status	progress/no_solution/planned	string

## WPD

### Waypoint Driver

Key	Value	Type
start	start position of gotten plan	string
finish	end position of gotten plan	string
robot	current position of robot	string
length	number of waypoints	int
current_index	target waypoint index	int
current_waypoint	target waypoint position	string
status	selected / finished / event	string
event	event description	string

## PER

### Sensor State and Status

Key	Value	Type
sensor	sensor id	string
status	status of sensor	string

confidence	reliability of sensor data	integer [0-100%]
------------	----------------------------	---------------------

**if** sensor\_id = communication  
status := { ok, fail, high\_latency }

## LOC

### Localization Source

Key	Value	Type
source	Comma separated list of sources (GPS/Odometry/Visual/etc)	string

### Localization Confidence

Key	Value	Type
confidence	confidence of localization : 0% - 100%	integer

## LLC

### Platform State and Status

Key	Value	Type
???	???	???

### Safety Status

Key	Value	Type
???	???	???



## 4. Components Control Events

This section describes the system events data flow.

**All messages below are sent over topic /decision\_making/events and are of message type std\_msgs/String**

### **/Teleoperation**

Switch to Teleoperation mode request

**From:** OCU

**To:** LLC, PER, SMME

### **/Autonomy**

Switch to Autonomy mode request

**From:** OCU

**To:** LLC, PER, SMME

### **SMME Mission and Component control**

/SMME/Start

/SMME/Pause

/SMME/Abort

/SMME/Resume

/SMME/Complete

/SMME/Delete

**From:** OCU

**To:** SMME

### **/E-Stop**

Emergency stop

**From:** SMME

**To:** LLC

### **WPD Component control**

/wpd/Start

/wpd/Stop

/wpd/Standby

/wpd/Resume

**From:** SMME

**To:** WPD

## **PP Component control**

/pp/Start  
/pp/Stop  
/pp/Standby  
/pp/Resume  
**From:** SMME  
**To:** PP

## **LLC Component control**

/llc/Start  
/llc/Stop  
/llc/Standby  
/llc/Resume  
**From:** SMME  
**To:** PP

## **WSM Component control**

/WSM/Start  
/WSM/Stop  
**From:** SMME  
**To:** WSM

## **Monitoring Events**

Problems detection events  
/XXX  
/XXX  
....  
**From:** SSM  
**To:** SMME, OCU

## **/IEDDetected**

Notification about IED detection  
**From:** IEDSIM  
**To:** SMME

## **ABL Events**

**/NoPathFound**  
No path planning solution found  
**Compliment:** /PathFound  
**From:** PP  
**To:** SMME, OCU

**/CommFail**

Communication problem

**Compliment:** /CommOK

**From:** SSM

**To:** OCU

**/ObstacleDetected**

Obstacle detected

**Compliment:** /AllClear

**From:** PP

**To:** SMME, OCU

**/RoadDetected**

Cells selected as Road detected near Global Path waypoints

**Compliment:** /OpenSpace

**From:** PP

**To:** SMME, OCU

**/Turn-over**

Prediction about Turn-Over of robot

**Compliment:** /StablePosition

**From:** SSM

**To:** SMME, OCU

**/Collision**

Collision detection

**Compliment:** /NoCollisions

**From:** LLC

**To:** SMME, OCU

**/NoGPS**

No GPS signal

**Compliment:** /GPSOK

**From:** LOC

**To:** SMME, OCU

**/AssistanceRequired**

Ask for assistance. General, not predefined problem.

**Compliment:** ---

**From:** SMME

**To:** OCU

## 5. Types

### std\_msgs

General standard messages.

#### std\_msgs/String

- **string data** : vector of chars represents text data

#### std\_msgs/Int32

- **int32 data** : integer number, 4 bytes

#### std\_msgs/Float64

- **float64 data** : float number, 8 bytes

#### std\_msgs/Header

Standard metadata for higher-level stamped data types.

This is generally used to communicate timestamped data in a particular coordinate frame.

- **uint32 seq : sequence ID: consecutively increasing ID**
- **time stamp** :  
Two-integer timestamp that is expressed as:
  - stamp.sec: seconds (stamp\_secs) since epoch
  - stamp.nsec: nanoseconds since stamp\_secstime-handling sugar is provided by the client library
- **string frame\_id** :  
Frame this data is associated with
  - 0: no frame
  - 1: global frame

### sensor\_msgs

Sensor messages

#### sensor\_msgs/NavSatFix

Navigation Satellite fix for any Global Navigation Satellite System

Specified using the WGS 84 reference ellipsoid

header.stamp specifies the ROS time for this measurement (the corresponding satellite time may be reported using the sensor\_msgs/TimeReference message).

header.frame\_id is the frame of reference reported by the satellite receiver, usually the location of the antenna. This is a Euclidean frame relative to the vehicle, not a reference ellipsoid.

- **std\_msgs/Header header**

- **sensor\_msgs/NavSatStatus status** : satellite fix status information  
Navigation Satellite fix status for any Global Navigation Satellite System. Whether to output an augmented fix is determined by both the fix type and the last time differential corrections were received. A fix is valid when status >= STATUS\_FIX.
  - **int8 status**
    - int8 STATUS\_NO\_FIX = -1 : unable to fix position
    - int8 STATUS\_FIX = 0 : unaugmented fix
    - int8 STATUS\_SBAS\_FIX = 1 : with satellite-based augmentation
    - int8 STATUS\_GBAS\_FIX = 2 : with ground-based augmentation
  - **uint16 service** : Bits defining which Global Navigation Satellite System signals were used by the receiver.
    - uint16 SERVICE\_GPS = 1
    - uint16 SERVICE\_GLONASS = 2
    - uint16 SERVICE\_COMPASS = 4 : includes BeiDou.
    - uint16 SERVICE\_GALILEO = 8
- **float64 latitude** : Latitude [degrees]. Positive is north of equator; negative is south
- **float64 longitude** : Longitude [degrees]. Positive is east of prime meridian; negative is west.
- **float64 altitude** : Altitude [m]. Positive is above the WGS 84 ellipsoid
- # (quiet NaN if no altitude is available).
- **float64[9] position\_covariance** : Position covariance [m<sup>2</sup>] defined relative to a tangential plane through the reported position. The components are East, North, and Up (ENU), in row-major order.  
Beware: this coordinate system exhibits singularities at the poles.
- **uint8 position\_covariance\_type** : If the covariance of the fix is known, fill it in completely. If the GPS receiver provides the variance of each measurement, put them along the diagonal. If only Dilution of Precision is available, estimate an approximate covariance from that.
  - uint8 COVARIANCE\_TYPE\_UNKNOWN = 0
  - uint8 COVARIANCE\_TYPE\_APPROXIMATED = 1
  - uint8 COVARIANCE\_TYPE\_DIAGONAL\_KNOWN = 2
  - uint8 COVARIANCE\_TYPE\_KNOWN = 3

## sensor\_msgs/Imu

This is a message to hold data from an IMU (Inertial Measurement Unit)

Accelerations should be in m/s<sup>2</sup> (not in g's), and rotational velocity should be in rad/sec. If the covariance of the measurement is known, it should be filled in (if all you know is the variance of each measurement, e.g. from the datasheet, just put those along the diagonal). A covariance matrix of all zeros will be interpreted as "covariance unknown", and to use the data a covariance will have to be assumed or gotten from some other source. If you have no estimate for one of the data elements (e.g. your IMU doesn't produce an orientation estimate), please set element 0 of the associated covariance matrix to -1. If you are interpreting this message, please check

for a value of -1 in the first element of each covariance matrix, and disregard the associated estimate.

- **std\_msgs/Header header**
- **geometry\_msgs/Quaternion orientation :**  
This represents an orientation in free space in quaternion form
  - **float64 x**
  - **float64 y**
  - **float64 z**
  - **float64 w**
- **float64[9] orientation\_covariance :** Row major about x, y, z axes
- **geometry\_msgs/Vector3 angular\_velocity :**  
This represents a vector in free space.
  - **float64 x**
  - **float64 y**
  - **float64 z**
- **float64[9] angular\_velocity\_covariance :** Row major about x, y, z axes
- **geometry\_msgs/Vector3 linear\_acceleration :**  
This represents a vector in free space.
  - **float64 x**
  - **float64 y**
  - **float64 z**
- **float64[9] linear\_acceleration\_covariance :** Row major about x, y, z

### **sensor\_msgs/LaserScan**

Single scan from a planar laser range-finder

If you have another ranging device with different behavior (e.g. a sonar array), please find or create a different message, since applications will make fairly laser-specific assumptions about this data

- **std\_msgs/Header header**  
timestamp in the header is the acquisition time of the first ray in the scan.  
in frame frame\_id, angles are measured around the positive Z axis (counterclockwise, if Z is up) with zero angle being forward along the x axis
- **float32 angle\_min :** start angle of the scan [rad]
- **float32 angle\_max :** end angle of the scan [rad]
- **float32 angle\_increment :** angular distance between measurements [rad]
- **float32 time\_increment :** time between measurements [seconds] - if your scanner is moving, this will be used in interpolating position of 3d points
- **float32 scan\_time :** time between scans [seconds]
- **float32 range\_min :** minimum range value [m]
- **float32 range\_max :** maximum range value [m]
- **float32[] ranges :** range data [m] (Note: values < range\_min or > range\_max should be discarded)
- **float32[] intensities :** intensity data [device-specific units]. If your device does not provide intensities, please leave the array empty.

## sensor\_msgs/Image

This message contains an uncompressed image (0, 0) is at top-left corner of image

- **std\_msgs/Header header**  
Header timestamp should be acquisition time of image  
Header frame\_id should be optical frame of camera origin of frame should be optical center of camera  
+x should point to the right in the image  
+y should point down in the image  
+z should point into to plane of the image  
If the frame\_id here and the frame\_id of the CameraInfo message associated with the image conflict the behavior is undefined
- **uint32 height**  
image height, that is, number of rows
- **uint32 width**  
image width, that is, number of columns
- **string encoding**  
The legal values for encoding are in file src/image\_encodings.cpp  
If you want to standardize a new string format, join  
ros-users@lists.sourceforge.net and send an email proposing a new encoding.  
Encoding of pixels -- channel meaning, ordering, size  
taken from the list of strings in include/sensor\_msgs/image\_encodings.h
- **uint8 is\_bigendian** : is this data bigendian?
- **uint32 step** : Full row length in bytes
- **uint8[] data** : actual matrix data, size is (step \* rows)

## sensor\_msgs/JointState

This is a message that holds data to describe the state of a set of torque controlled joints.

The state of each joint (revolute or prismatic) is defined by:

- \* the position of the joint (rad or m),
- \* the velocity of the joint (rad/s or m/s) and
- \* the effort that is applied in the joint (Nm or N).

Each joint is uniquely identified by its name

This message consists of a multiple arrays, one for each part of the joint state.

The goal is to make each of the fields optional. When e.g. your joints have no effort associated with them, you can leave the effort array empty.

All arrays in this message should have the same size, or be empty.

This is the only way to uniquely associate the joint name with the correct states.

- **Header header** :  
The header specifies the time at which the joint states were recorded. All the joint states in one message have to be recorded at the same time.
- **string[] name** : list of joints identifiers
- **float64[] position** : list of joints positions [rad]
- **float64[] velocity** : list of joints velocity [rad/s]
- **float64[] effort** : list of joints efforts

## geometry\_msgs

Geometry messages

### geometry\_msgs/PoseStamped

A Pose with reference coordinate frame and timestamp

- **std\_msgs/Header header**
- **geometry\_msgs/Pose pose**

A representation of pose in free space, composed of position and orientation.

- **geometry\_msgs/Point position**  
position of robot (meters)
  - float64 x**
  - float64 y**
  - float64 z**
- **geometry\_msgs/Quaternion orientation**
  - float64 x**
  - float64 y**
  - float64 z**
  - float64 w**

### geometry\_msgs/TwistStamped

A twist with reference coordinate frame and timestamp

- **std\_msgs/Header header**
- **geometry\_msgs/Twist twist**

This expresses velocity in free space broken into its linear and angular parts.

- **geometry\_msgs/Vector3 linear**  
This represents a vector in free space. (m/sec)
  - float64 x**
  - float64 y**
  - float64 z**
- **geometry\_msgs/Vector3 angular**  
This represents a vector in free space. (rad/sec)
  - float64 x**
  - float64 y**
  - float64 z**

### geometry\_msgs/PoseWithCovariance

This represents a pose in free space with uncertainty.

- **geometry\_msgs/Pose pose**  
A representation of pose in free space, composed of position and orientation.
  - **geometry\_msgs/Point position**  
position of robot (meters)
    - float64 x**



- float64 y
  - float64 z
- **geometry\_msgs/Quaternion orientation**
  - float64 x
  - float64 y
  - float64 z
  - float64 w
- **float64[36] covariance**

Row-major representation of the 6x6 covariance matrix. The orientation parameters use a fixed-axis representation.

In order, the parameters are:

(x, y, z, rotation about X axis, rotation about Y axis, rotation about Z axis)

### **geometry\_msgs/TwistWithCovariance**

This expresses velocity in free space with uncertainty

- **geometry\_msgs/Twist twist**

This expresses velocity in free space broken into its linear and angular parts.

  - **geometry\_msgs/Vector3 linear**

This represents a vector in free space. (m/sec)

    - float64 x
    - float64 y
    - float64 z
  - **geometry\_msgs/Vector3 angular**

This represents a vector in free space. (rad/sec)

    - float64 x
    - float64 y
    - float64 z
- **float64[36] covariance**

Row-major representation of the 6x6 covariance matrix

The orientation parameters use a fixed-axis representation.

In order, the parameters are:

(x, y, z, rotation about X axis, rotation about Y axis, rotation about Z axis)

### **geometry\_msgs/PoseWithCovarianceStamped**

This expresses an estimated pose with a reference coordinate frame and timestamp

- **Header header**
- **PoseWithCovariance pose**

## **Nav\_msgs**

Navigation messages

### **nav\_msgs/Path waypoints**

An array of poses that represents a Path for a robot to follow

- **std\_msgs/Header header**

- **geometry\_msgs/PoseStamped[] poses** : waypoints list, defined by target position and orientation.

### **nav\_msgs/Odometry**

This represents an estimate of a position and velocity in free space.

- **std\_msgs/Header header**
- **string child\_frame\_id**
- **geometry\_msgs/PoseWithCovariance pose**  
The pose in this message should be specified in the coordinate frame given by header.frame\_id.
- **geometry\_msgs/TwistWithCovariance twist**  
The twist in this message should be specified in the coordinate frame given by the child\_frame\_id

## **diagnostic\_msgs**

Diagnostic messages

### **diagnostic\_msgs/DiagnosticStatus**

This message holds the status of an individual component of the robot.

- **byte level** : level of operation enumerated above  
Possible levels of operations:  
byte OK=0  
byte WARN=1  
byte ERROR=2
- **string name** : a description of the test/component reporting
- **string message** : a description of the status
- **string hardware\_id** : a hardware unique string
- **diagnostic\_msgs/KeyValue[] values** :  
an array of values associated with the status
  - **string key** : what to label this value when viewing
  - **string value** : a value to track over time

## **robil\_msgs**

ROBIL specific messages

### **robil\_msgs/AssignNavTask**

Navigation task definition. Task is a global path with target heading at the last position.

- **std\_msgs/Header header** : time and frame information
- **string task\_id** : Task ID
- **string task\_description** : textual description of task
- **float32 heading\_at\_last\_point** : heading at last point of path (radians)
- **nav\_msgs/Odometry[] waypoints** :

waypoints with speed limitation of global path

- **std\_msgs/Header header** : time and frame information
- **geometry\_msgs/PoseWithCovariance pose** : target pose of waypoint
- **geometry\_msgs/TwistWithCovariance twist** : speed limit of segment from the waypoint up to next waypoint in the path.

## **robin\_msgs/AssignManipulatorTask**

Manipulator task definition. Task is a list of steps. Step is a manipulator command.

- **std\_msgs/Header header** : time and frame information
- **string task\_id** : Task Id
- **string task\_description** : textual description of task
- **robin\_msgs/AssignManipulatorTaskStep[] steps** :

List of manipulator steps commands

- **int32 id** : Unique step id
- **int8 type** : Type of step
  - int8 type\_unknown=0
  - int8 type\_blade\_height=1
  - int8 type\_blade\_angle=2
  - int8 type\_clamp=3
  - int8 type\_advance=4
  - int8 type\_turn=5
- **float32 value** : target angle of step command (radians)
  - Variants of data represented by this value:
    - Height [m]
    - Angle [deg]
    - Clamp [0-100%]
    - Advance [m]
    - Turn [deg]
- **int8 blade\_relativity** :
  - is actual for type = *blade\_height* or *advance* or *turn*
  - int8 blade\_relativity\_absolute=0 - Absolute height
  - int8 blade\_relativity\_graud=1 - Height relative to ground
- **int64 success\_timeout** : time limit of step (seconds)
- **int64 duration\_at\_end** : pause duration at end (seconds)

## **robin\_msgs/AssignMission**

Mission definition. Mission is a list of Tasks (of both types: nav. and manip.)

- **std\_msgs/Header header** : time and frame information
- **string mission\_id** : Mission ID
- **string mission\_description** : textual description of mission
- **robin\_msgs/AssignMissionTask[] tasks** : list of tasks
  - **string task\_id** : Task ID

## **robin\_msgs/Path**

An array of poses that represents a Path for a robot to follow

- **nav\_msgs/Path waypoints** : waypoints defined as standard ros path
- **bool is\_ip\_defined** : *True* if Interested Point defined, *Else* in other case.
- **geometry\_msgs/Point ip** : if *is\_ip\_defined* is *True*, location of interested points. The heading of the robot after the path following has to be directed to this point.
- **bool is\_heading\_defined** : *True* if heading of last point defined, *Else* in other case.
- **float32 heading** : If *is\_heading\_defined* is *True*, this is a target heading at the last path waypoint.

## **robil\_msgs/MissionAcceptance**

Report message about mission acceptance

- **string mission\_id** : Unique ID of mission
- **time mission\_assign\_stamp** : time from header of AssignMission message for message with *mission\_id*.
- **int32 status** :  
if >0 then mission accepted  
else mission rejected and value of status is a error code.

## **robil\_msgs/IEDLocation**

IED detection event

- **bool is\_detected** : *True* if IED detected, *False* in other case
- **geometry\_msgs/Point** : Position of IED object.

## **robil\_msgs/Map**

This represents a 2-D grid map, in which each cell represents the type of occupancy and detected objects.

- **std\_msgs/Header header**
- **nav\_msgs/MapMetaData info** : MetaData for the map.  
This hold basic information about the characterists of the Map
  - **time map\_load\_time** : The time at which the map was loaded
  - **float32 resolution** : The map resolution [m/cell]
  - **uint32 width** : Map width [cells]
  - **uint32 height** : Map height [cells]
  - **geometry\_msgs/Pose origin** : The origin of the map [m, m, rad]. This is the real-world pose of the cell (0,0) in the map.
- **robil\_msgs/MapCell[] data** :  
The map data, in row-major order, starting with (0,0)  
Cell description:
  - **float32 height** : average altitude of the cell
  - **int8 type** : type occupancy of the cell  
int8 type\_unscanned=0  
int8 type\_clear=1  
int8 type\_obstacle=2
  - **int32 feature** : features detected in the cell

```
int32 feature_unknown=0
int32 feature_road=1
int32 feature_object=2
```

## **rohil\_msgs/MultiLaserScan**

Scan of multilayer laser (IBEO).

- **std\_msgs/Header header**
- **float32 angle\_min\_t** : start angle of the top planes scan [rad]
- **float32 angle\_max\_t** : end angle of the top planes scan [rad]
- **float32 angle\_min\_b** : start angle of the bottom planes scan [rad]
- **float32 angle\_max\_b** : end angle of the bottom planes scan [rad]
- **float32 angle\_increment** : angular distance between measurements [rad]
- **float32 angle\_t1** : vertical angle of top plane 1
- **float32 angle\_t2** : vertical angle of top plane 2
- **float32 angle\_b1** : vertical angle of bottom plane 1
- **float32 angle\_b2** : vertical angle of bottom plane 2
- **float32 time\_increment** : time between measurements [seconds]  
if your scanner is moving, this will be used in interpolating position of 3d points
- **float32 scan\_time** : time between scans [seconds]
- **float32 range\_min** : minimum range value [m]
- **float32 range\_max** : maximum range value [m]
- **float32[] ranges\_t1** : range data [m]  
values < range\_min or > range\_max should be discarded
- **float32[] ranges\_t2** : range data [m]  
values < range\_min or > range\_max should be discarded
- **float32[] ranges\_b1** : range data [m]  
values < range\_min or > range\_max should be discarded
- **float32[] ranges\_b2** : range data [m]  
values < range\_min or > range\_max should be discarded
- **float32[] intensities** : intensity data [devicespecific units].  
If your device does not provide intensities, please leave the array empty.

## 6. Transformations (TF)

### Transformations Tree

- world
  - map
    - occupancy\_map
    - mini\_map
    - odom
      - base\_link
        - body
          - cabin
            - Ibeo
          - cameraL\_frame
          - cameraR\_frame
          - front\_sick
          - gps\_ins
          - ibeo
          - left\_sick
          - right\_sick
          - base\_arm
            - main\_arm
              - loader
                - brackets

### Coordinate Frames Specification

#### **world**

The coordinate frame is a start coordinates fixed frame, with its Z-axis pointing upwards.

#### **map**

The coordinate frame is a world fixed frame, with its Z-axis pointing upwards. The pose of a mobile platform, relative to the map frame, should not significantly drift over time. The map frame is not continuous, meaning the pose of a mobile platform in the map frame can change in discrete jumps at any time.

In a typical setup, a localization component constantly re-computes the robot pose in

the world frame based on sensor observations, therefore eliminating drift, but causing discrete jumps when new sensor information arrives.

The map frame is useful as a long-term global reference, but discrete jumps make it a poor reference frame for local sensing and acting.

### **odom**

The coordinate frame called odom is a world-fixed frame. The pose of a mobile platform in the odom frame can drift over time, without any bounds. This drift makes the odom frame useless as a long-term global reference. However, the pose of a robot in the odom frame is guaranteed to be continuous, meaning that the pose of a mobile platform in the odom frame always evolves in a smooth way, without discrete jumps. In a typical setup the odom frame is computed based on an odometry source, such as wheel odometry, visual odometry or an inertial measurement unit.

The odom frame is useful as an accurate, short-term local reference, but drift makes it a poor frame for long-term reference.

### **base\_link**

The coordinate frame called base\_link is rigidly attached to the mobile robot base.

The base\_link can be attached to the base in any arbitrary position or orientation; for every hardware platform there will be a different place on the base that provides an obvious point of reference.

### **occupancy\_map**

The map (0,0) coordinates (the main occupancy grid offset and orientation)

The coordinate frame is rigidly attached to the (0,0) point of map (start of occupancy grid).

### **mini-map**

The coordinate frame is rigidly attached to the (0,0) point of mini-map (location of robot on mini-map).

### **body**

This is the frame of the mobile robot body and all other hardware connects to it.

### **Ibeo**

This frame is of the static link to the ibeo sensor which is rigidly attached to the cabin.

### **cameraL\_frame**

this is the coordinate frame of the Flea3 left camera. the camera is rigidly attached to the body and may be positioned differently

### **cameraR\_frame**

this is the coordinate frame of the Flea3 right camera. the camera is rigidly attached to the body and may be positioned differently

### **front\_sick**

this is the coordinate frame of the front sick scanner. It is rigidly attached to the body.

**gps\_ins**

this is the coordinate frame of the GPS/INS sensor. It is rigidly attached to the body.

**ibeo**

this is the coordinate frame of the ibeo sensor. It is rigidly attached to the body. (it should be attached to the ibeo link)

**left\_sick**

this is the coordinate frame of the left sick scanner. It is rigidly attached to the body.

**right\_sick**

this is the coordinate frame of the right sick scanner. It is rigidly attached to the body.

**base\_arm**

the frame of the base\_arm represents the virtual joint between the body and base-arm link, located in the middle between the 2 parallel base\_arm links. The frame origin is in a fixed position related to the body frame and can rotate around the Y axis. The angle of rotation is a function of the wire sensor's 1st DOF state.

**main\_arm**

the frame of the main\_arm represents the virtual joint between the base\_arm and main-arm links, located in the middle between the 2 parallel main\_arm links. The frame origin is in a fixed position related to the base\_arm frame and can rotate around the Y axis. The angle of rotation is a function of the wire sensor's 1st DOF state.

**loader**

the loader frame is positioned on the virtual joint between the loader and main\_arm links and can rotate around the Y axis according to loader rotation commands. its rotation is a function of the 2nd DOF of the wire sensor.

**brackets**

the brackets frame is positioned on the virtual joint between the 2 brackets and the loader links and can rotate around the Y axis according to brackets open/close commands. its rotation is a function of the 3rd DOF of the wire sensor.