;; Auto-generated. Do not edit!

(when (boundp 'robotnik\_msgs\_new::Pose2DStamped)

(if (not (find-package "ROBOTNIK\_MSGS\_NEW"))

(make-package "ROBOTNIK\_MSGS\_NEW"))

(shadow 'Pose2DStamped (find-package "ROBOTNIK\_MSGS\_NEW")))

(unless (find-package "ROBOTNIK\_MSGS\_NEW::POSE2DSTAMPED")

(make-package "ROBOTNIK\_MSGS\_NEW::POSE2DSTAMPED"))

(in-package "ROS")

;;//! \htmlinclude Pose2DStamped.msg.html

(if (not (find-package "GEOMETRY\_MSGS"))

(ros::roseus-add-msgs "geometry\_msgs"))

(if (not (find-package "STD\_MSGS"))

(ros::roseus-add-msgs "std\_msgs"))

(defclass robotnik\_msgs\_new::Pose2DStamped

:super ros::object

:slots (\_header \_pose ))

(defmethod robotnik\_msgs\_new::Pose2DStamped

(:init

(&key

((:header \_\_header) (instance std\_msgs::Header :init))

((:pose \_\_pose) (instance geometry\_msgs::Pose2D :init))

)

(send-super :init)

(setq \_header \_\_header)

(setq \_pose \_\_pose)

self)

(:header

(&rest \_\_header)

(if (keywordp (car \_\_header))

(send\* \_header \_\_header)

(progn

(if \_\_header (setq \_header (car \_\_header)))

\_header)))

(:pose

(&rest \_\_pose)

(if (keywordp (car \_\_pose))

(send\* \_pose \_\_pose)

(progn

(if \_\_pose (setq \_pose (car \_\_pose)))

\_pose)))

(:serialization-length

()

(+

;; std\_msgs/Header \_header

(send \_header :serialization-length)

;; geometry\_msgs/Pose2D \_pose

(send \_pose :serialization-length)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

(make-string-output-stream (send self :serialization-length)))))

;; std\_msgs/Header \_header

(send \_header :serialize s)

;; geometry\_msgs/Pose2D \_pose

(send \_pose :serialize s)

;;

(if (null strm) (get-output-stream-string s))))

(:deserialize

(buf &optional (ptr- 0))

;; std\_msgs/Header \_header

(send \_header :deserialize buf ptr-) (incf ptr- (send \_header :serialization-length))

;; geometry\_msgs/Pose2D \_pose

(send \_pose :deserialize buf ptr-) (incf ptr- (send \_pose :serialization-length))

;;

self)

)

(setf (get robotnik\_msgs\_new::Pose2DStamped :md5sum-) "b5f1e28823201bc5ea7e310fc49d253f")

(setf (get robotnik\_msgs\_new::Pose2DStamped :datatype-) "robotnik\_msgs\_new/Pose2DStamped")

(setf (get robotnik\_msgs\_new::Pose2DStamped :definition-)

"Header header

geometry\_msgs/Pose2D pose

================================================================================

MSG: std\_msgs/Header

# Standard metadata for higher-level stamped data types.

# This is generally used to communicate timestamped data

# in a particular coordinate frame.

#

# sequence ID: consecutively increasing ID

uint32 seq

#Two-integer timestamp that is expressed as:

# \* stamp.sec: seconds (stamp\_secs) since epoch (in Python the variable is called 'secs')

# \* stamp.nsec: nanoseconds since stamp\_secs (in Python the variable is called 'nsecs')

# time-handling sugar is provided by the client library

time stamp

#Frame this data is associated with

string frame\_id

================================================================================

MSG: geometry\_msgs/Pose2D

# Deprecated

# Please use the full 3D pose.

# In general our recommendation is to use a full 3D representation of everything and for 2D specific applications make the appropriate projections into the plane for their calculations but optimally will preserve the 3D information during processing.

# If we have parallel copies of 2D datatypes every UI and other pipeline will end up needing to have dual interfaces to plot everything. And you will end up with not being able to use 3D tools for 2D use cases even if they're completely valid, as you'd have to reimplement it with different inputs and outputs. It's not particularly hard to plot the 2D pose or compute the yaw error for the Pose message and there are already tools and libraries that can do this for you.

# This expresses a position and orientation on a 2D manifold.

float64 x

float64 y

float64 theta

")

(provide :robotnik\_msgs\_new/Pose2DStamped "b5f1e28823201bc5ea7e310fc49d253f")