;; Auto-generated. Do not edit!

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

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

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

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

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

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

(in-package "ROS")

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

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

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

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

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

(defclass robotnik\_msgs\_new::SetElevatorActionFeedback

:super ros::object

:slots (\_header \_status \_feedback ))

(defmethod robotnik\_msgs\_new::SetElevatorActionFeedback

(:init

(&key

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

((:status \_\_status) (instance actionlib\_msgs::GoalStatus :init))

((:feedback \_\_feedback) (instance robotnik\_msgs\_new::SetElevatorFeedback :init))

)

(send-super :init)

(setq \_header \_\_header)

(setq \_status \_\_status)

(setq \_feedback \_\_feedback)

self)

(:header

(&rest \_\_header)

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

(send\* \_header \_\_header)

(progn

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

\_header)))

(:status

(&rest \_\_status)

(if (keywordp (car \_\_status))

(send\* \_status \_\_status)

(progn

(if \_\_status (setq \_status (car \_\_status)))

\_status)))

(:feedback

(&rest \_\_feedback)

(if (keywordp (car \_\_feedback))

(send\* \_feedback \_\_feedback)

(progn

(if \_\_feedback (setq \_feedback (car \_\_feedback)))

\_feedback)))

(:serialization-length

()

(+

;; std\_msgs/Header \_header

(send \_header :serialization-length)

;; actionlib\_msgs/GoalStatus \_status

(send \_status :serialization-length)

;; robotnik\_msgs\_new/SetElevatorFeedback \_feedback

(send \_feedback :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)

;; actionlib\_msgs/GoalStatus \_status

(send \_status :serialize s)

;; robotnik\_msgs\_new/SetElevatorFeedback \_feedback

(send \_feedback :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))

;; actionlib\_msgs/GoalStatus \_status

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

;; robotnik\_msgs\_new/SetElevatorFeedback \_feedback

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

;;

self)

)

(setf (get robotnik\_msgs\_new::SetElevatorActionFeedback :md5sum-) "d977e6a7039669a4be1509f51814113f")

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

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

"# ====== DO NOT MODIFY! AUTOGENERATED FROM AN ACTION DEFINITION ======

Header header

actionlib\_msgs/GoalStatus status

SetElevatorFeedback feedback

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

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: actionlib\_msgs/GoalStatus

GoalID goal\_id

uint8 status

uint8 PENDING = 0 # The goal has yet to be processed by the action server

uint8 ACTIVE = 1 # The goal is currently being processed by the action server

uint8 PREEMPTED = 2 # The goal received a cancel request after it started executing

# and has since completed its execution (Terminal State)

uint8 SUCCEEDED = 3 # The goal was achieved successfully by the action server (Terminal State)

uint8 ABORTED = 4 # The goal was aborted during execution by the action server due

# to some failure (Terminal State)

uint8 REJECTED = 5 # The goal was rejected by the action server without being processed,

# because the goal was unattainable or invalid (Terminal State)

uint8 PREEMPTING = 6 # The goal received a cancel request after it started executing

# and has not yet completed execution

uint8 RECALLING = 7 # The goal received a cancel request before it started executing,

# but the action server has not yet confirmed that the goal is canceled

uint8 RECALLED = 8 # The goal received a cancel request before it started executing

# and was successfully cancelled (Terminal State)

uint8 LOST = 9 # An action client can determine that a goal is LOST. This should not be

# sent over the wire by an action server

#Allow for the user to associate a string with GoalStatus for debugging

string text

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

MSG: actionlib\_msgs/GoalID

# The stamp should store the time at which this goal was requested.

# It is used by an action server when it tries to preempt all

# goals that were requested before a certain time

time stamp

# The id provides a way to associate feedback and

# result message with specific goal requests. The id

# specified must be unique.

string id

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

MSG: robotnik\_msgs\_new/SetElevatorFeedback

# ====== DO NOT MODIFY! AUTOGENERATED FROM AN ACTION DEFINITION ======

robotnik\_msgs\_new/ElevatorStatus status

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

MSG: robotnik\_msgs\_new/ElevatorStatus

# state

string RAISING=raising

string LOWERING=lowering

string IDLE=idle

string ERROR\_G\_IO=error\_getting\_io

string ERROR\_S\_IO=error\_setting\_io

string ERROR\_TIMEOUT=error\_timeout\_in\_action

# position

string UP=up

string DOWN=down

string UNKNOWN=unknown

# IDLE, RAISING, LOWERING

string state

# UP, DOWN, UNKNOWN

string position

float32 height

")

(provide :robotnik\_msgs\_new/SetElevatorActionFeedback "d977e6a7039669a4be1509f51814113f")