;; Auto-generated. Do not edit!

(when (boundp 'gazebo\_msgs\_new::SetLinkState)

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

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

(shadow 'SetLinkState (find-package "GAZEBO\_MSGS\_NEW")))

(unless (find-package "GAZEBO\_MSGS\_NEW::SETLINKSTATE")

(make-package "GAZEBO\_MSGS\_NEW::SETLINKSTATE"))

(unless (find-package "GAZEBO\_MSGS\_NEW::SETLINKSTATEREQUEST")

(make-package "GAZEBO\_MSGS\_NEW::SETLINKSTATEREQUEST"))

(unless (find-package "GAZEBO\_MSGS\_NEW::SETLINKSTATERESPONSE")

(make-package "GAZEBO\_MSGS\_NEW::SETLINKSTATERESPONSE"))

(in-package "ROS")

(defclass gazebo\_msgs\_new::SetLinkStateRequest

:super ros::object

:slots (\_link\_state ))

(defmethod gazebo\_msgs\_new::SetLinkStateRequest

(:init

(&key

((:link\_state \_\_link\_state) (instance gazebo\_msgs\_new::LinkState :init))

)

(send-super :init)

(setq \_link\_state \_\_link\_state)

self)

(:link\_state

(&rest \_\_link\_state)

(if (keywordp (car \_\_link\_state))

(send\* \_link\_state \_\_link\_state)

(progn

(if \_\_link\_state (setq \_link\_state (car \_\_link\_state)))

\_link\_state)))

(:serialization-length

()

(+

;; gazebo\_msgs\_new/LinkState \_link\_state

(send \_link\_state :serialization-length)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; gazebo\_msgs\_new/LinkState \_link\_state

(send \_link\_state :serialize s)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; gazebo\_msgs\_new/LinkState \_link\_state

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

;;

self)

)

(defclass gazebo\_msgs\_new::SetLinkStateResponse

:super ros::object

:slots (\_success \_status\_message ))

(defmethod gazebo\_msgs\_new::SetLinkStateResponse

(:init

(&key

((:success \_\_success) nil)

((:status\_message \_\_status\_message) "")

)

(send-super :init)

(setq \_success \_\_success)

(setq \_status\_message (string \_\_status\_message))

self)

(:success

(&optional \_\_success)

(if \_\_success (setq \_success \_\_success)) \_success)

(:status\_message

(&optional \_\_status\_message)

(if \_\_status\_message (setq \_status\_message \_\_status\_message)) \_status\_message)

(:serialization-length

()

(+

;; bool \_success

1

;; string \_status\_message

4 (length \_status\_message)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; bool \_success

(if \_success (write-byte -1 s) (write-byte 0 s))

;; string \_status\_message

(write-long (length \_status\_message) s) (princ \_status\_message s)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; bool \_success

(setq \_success (not (= 0 (sys::peek buf ptr- :char)))) (incf ptr- 1)

;; string \_status\_message

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_status\_message (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;;

self)

)

(defclass gazebo\_msgs\_new::SetLinkState

:super ros::object

:slots ())

(setf (get gazebo\_msgs\_new::SetLinkState :md5sum-) "8a5146eb66ae4d26b0860b08f3f271be")

(setf (get gazebo\_msgs\_new::SetLinkState :datatype-) "gazebo\_msgs\_new/SetLinkState")

(setf (get gazebo\_msgs\_new::SetLinkState :request) gazebo\_msgs\_new::SetLinkStateRequest)

(setf (get gazebo\_msgs\_new::SetLinkState :response) gazebo\_msgs\_new::SetLinkStateResponse)

(defmethod gazebo\_msgs\_new::SetLinkStateRequest

(:response () (instance gazebo\_msgs\_new::SetLinkStateResponse :init)))

(setf (get gazebo\_msgs\_new::SetLinkStateRequest :md5sum-) "8a5146eb66ae4d26b0860b08f3f271be")

(setf (get gazebo\_msgs\_new::SetLinkStateRequest :datatype-) "gazebo\_msgs\_new/SetLinkStateRequest")

(setf (get gazebo\_msgs\_new::SetLinkStateRequest :definition-)

"gazebo\_msgs\_new/LinkState link\_state

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

MSG: gazebo\_msgs\_new/LinkState

# @todo: FIXME: sets pose and twist of a link. All children link poses/twists of the URDF tree are not updated accordingly, but should be.

string link\_name # link name, link\_names are in gazebo scoped name notation, [model\_name::body\_name]

geometry\_msgs/Pose pose # desired pose in reference frame

geometry\_msgs/Twist twist # desired twist in reference frame

string reference\_frame # set pose/twist relative to the frame of this link/body

# leave empty or \"world\" or \"map\" defaults to world-frame

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

MSG: geometry\_msgs/Pose

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

Point position

Quaternion orientation

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

MSG: geometry\_msgs/Point

# This contains the position of a point in free space

float64 x

float64 y

float64 z

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

MSG: geometry\_msgs/Quaternion

# This represents an orientation in free space in quaternion form.

float64 x

float64 y

float64 z

float64 w

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

MSG: geometry\_msgs/Twist

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

Vector3 linear

Vector3 angular

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

MSG: geometry\_msgs/Vector3

# This represents a vector in free space.

# It is only meant to represent a direction. Therefore, it does not

# make sense to apply a translation to it (e.g., when applying a

# generic rigid transformation to a Vector3, tf2 will only apply the

# rotation). If you want your data to be translatable too, use the

# geometry\_msgs/Point message instead.

float64 x

float64 y

---

bool success # return true if set wrench successful

string status\_message # comments if available

")

(setf (get gazebo\_msgs\_new::SetLinkStateResponse :md5sum-) "8a5146eb66ae4d26b0860b08f3f271be")

(setf (get gazebo\_msgs\_new::SetLinkStateResponse :datatype-) "gazebo\_msgs\_new/SetLinkStateResponse")

(setf (get gazebo\_msgs\_new::SetLinkStateResponse :definition-)

"gazebo\_msgs\_new/LinkState link\_state

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

MSG: gazebo\_msgs\_new/LinkState

# @todo: FIXME: sets pose and twist of a link. All children link poses/twists of the URDF tree are not updated accordingly, but should be.

string link\_name # link name, link\_names are in gazebo scoped name notation, [model\_name::body\_name]

geometry\_msgs/Pose pose # desired pose in reference frame

geometry\_msgs/Twist twist # desired twist in reference frame

string reference\_frame # set pose/twist relative to the frame of this link/body

# leave empty or \"world\" or \"map\" defaults to world-frame

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

MSG: geometry\_msgs/Pose

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

Point position

Quaternion orientation

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

MSG: geometry\_msgs/Point

# This contains the position of a point in free space

float64 x

float64 y

float64 z

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

MSG: geometry\_msgs/Quaternion

# This represents an orientation in free space in quaternion form.

float64 x

float64 y

float64 z

float64 w

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

MSG: geometry\_msgs/Twist

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

Vector3 linear

Vector3 angular

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

MSG: geometry\_msgs/Vector3

# This represents a vector in free space.

# It is only meant to represent a direction. Therefore, it does not

# make sense to apply a translation to it (e.g., when applying a

# generic rigid transformation to a Vector3, tf2 will only apply the

# rotation). If you want your data to be translatable too, use the

# geometry\_msgs/Point message instead.

float64 x

float64 y

---

bool success # return true if set wrench successful

string status\_message # comments if available

")

(provide :gazebo\_msgs\_new/SetLinkState "8a5146eb66ae4d26b0860b08f3f271be")