;; Auto-generated. Do not edit!

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

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

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

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

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

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

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

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

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

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

(in-package "ROS")

(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 gazebo\_msgs\_new::GetModelStateRequest

:super ros::object

:slots (\_model\_name \_relative\_entity\_name ))

(defmethod gazebo\_msgs\_new::GetModelStateRequest

(:init

(&key

((:model\_name \_\_model\_name) "")

((:relative\_entity\_name \_\_relative\_entity\_name) "")

)

(send-super :init)

(setq \_model\_name (string \_\_model\_name))

(setq \_relative\_entity\_name (string \_\_relative\_entity\_name))

self)

(:model\_name

(&optional \_\_model\_name)

(if \_\_model\_name (setq \_model\_name \_\_model\_name)) \_model\_name)

(:relative\_entity\_name

(&optional \_\_relative\_entity\_name)

(if \_\_relative\_entity\_name (setq \_relative\_entity\_name \_\_relative\_entity\_name)) \_relative\_entity\_name)

(:serialization-length

()

(+

;; string \_model\_name

4 (length \_model\_name)

;; string \_relative\_entity\_name

4 (length \_relative\_entity\_name)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; string \_model\_name

(write-long (length \_model\_name) s) (princ \_model\_name s)

;; string \_relative\_entity\_name

(write-long (length \_relative\_entity\_name) s) (princ \_relative\_entity\_name s)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; string \_model\_name

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

;; string \_relative\_entity\_name

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_relative\_entity\_name (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;;

self)

)

(defclass gazebo\_msgs\_new::GetModelStateResponse

:super ros::object

:slots (\_header \_pose \_twist \_success \_status\_message ))

(defmethod gazebo\_msgs\_new::GetModelStateResponse

(:init

(&key

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

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

((:twist \_\_twist) (instance geometry\_msgs::Twist :init))

((:success \_\_success) nil)

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

)

(send-super :init)

(setq \_header \_\_header)

(setq \_pose \_\_pose)

(setq \_twist \_\_twist)

(setq \_success \_\_success)

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

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)))

(:twist

(&rest \_\_twist)

(if (keywordp (car \_\_twist))

(send\* \_twist \_\_twist)

(progn

(if \_\_twist (setq \_twist (car \_\_twist)))

\_twist)))

(: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

()

(+

;; std\_msgs/Header \_header

(send \_header :serialization-length)

;; geometry\_msgs/Pose \_pose

(send \_pose :serialization-length)

;; geometry\_msgs/Twist \_twist

(send \_twist :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)))))

;; std\_msgs/Header \_header

(send \_header :serialize s)

;; geometry\_msgs/Pose \_pose

(send \_pose :serialize s)

;; geometry\_msgs/Twist \_twist

(send \_twist :serialize s)

;; 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))

;; std\_msgs/Header \_header

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

;; geometry\_msgs/Pose \_pose

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

;; geometry\_msgs/Twist \_twist

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

;; 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::GetModelState

:super ros::object

:slots ())

(setf (get gazebo\_msgs\_new::GetModelState :md5sum-) "4c515e936d3319c9610c559c60bfc3d4")

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

(setf (get gazebo\_msgs\_new::GetModelState :request) gazebo\_msgs\_new::GetModelStateRequest)

(setf (get gazebo\_msgs\_new::GetModelState :response) gazebo\_msgs\_new::GetModelStateResponse)

(defmethod gazebo\_msgs\_new::GetModelStateRequest

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

(setf (get gazebo\_msgs\_new::GetModelStateRequest :md5sum-) "4c515e936d3319c9610c559c60bfc3d4")

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

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

"string model\_name # name of Gazebo Model

string relative\_entity\_name # return pose and twist relative to this entity

# an entity can be a model, body, or geom

# be sure to use gazebo scoped naming notation (e.g. [model\_name::body\_name])

# leave empty or \"world\" will use inertial world frame

---

Header header # Standard metadata for higher-level stamped data types.

# \* header.seq holds the number of requests since the plugin started

# \* header.stamp timestamp related to the pose

# \* header.frame\_id not used but currently filled with the relative\_entity\_name

geometry\_msgs/Pose pose # pose of model in relative entity frame

geometry\_msgs/Twist twist # twist of model in relative entity frame

bool success # return true if get successful

string status\_message # comments if available

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

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/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

")

(setf (get gazebo\_msgs\_new::GetModelStateResponse :md5sum-) "4c515e936d3319c9610c559c60bfc3d4")

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

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

"string model\_name # name of Gazebo Model

string relative\_entity\_name # return pose and twist relative to this entity

# an entity can be a model, body, or geom

# be sure to use gazebo scoped naming notation (e.g. [model\_name::body\_name])

# leave empty or \"world\" will use inertial world frame

---

Header header # Standard metadata for higher-level stamped data types.

# \* header.seq holds the number of requests since the plugin started

# \* header.stamp timestamp related to the pose

# \* header.frame\_id not used but currently filled with the relative\_entity\_name

geometry\_msgs/Pose pose # pose of model in relative entity frame

geometry\_msgs/Twist twist # twist of model in relative entity frame

bool success # return true if get successful

string status\_message # comments if available

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

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/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

")

(provide :gazebo\_msgs\_new/GetModelState "4c515e936d3319c9610c559c60bfc3d4")