;; Auto-generated. Do not edit!

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

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

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

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

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

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

(in-package "ROS")

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

(defclass gazebo\_msgs\_new::ODEJointProperties

:super ros::object

:slots (\_damping \_hiStop \_loStop \_erp \_cfm \_stop\_erp \_stop\_cfm \_fudge\_factor \_fmax \_vel ))

(defmethod gazebo\_msgs\_new::ODEJointProperties

(:init

(&key

((:damping \_\_damping) (make-array 0 :initial-element 0.0 :element-type :float))

((:hiStop \_\_hiStop) (make-array 0 :initial-element 0.0 :element-type :float))

((:loStop \_\_loStop) (make-array 0 :initial-element 0.0 :element-type :float))

((:erp \_\_erp) (make-array 0 :initial-element 0.0 :element-type :float))

((:cfm \_\_cfm) (make-array 0 :initial-element 0.0 :element-type :float))

((:stop\_erp \_\_stop\_erp) (make-array 0 :initial-element 0.0 :element-type :float))

((:stop\_cfm \_\_stop\_cfm) (make-array 0 :initial-element 0.0 :element-type :float))

((:fudge\_factor \_\_fudge\_factor) (make-array 0 :initial-element 0.0 :element-type :float))

((:fmax \_\_fmax) (make-array 0 :initial-element 0.0 :element-type :float))

((:vel \_\_vel) (make-array 0 :initial-element 0.0 :element-type :float))

)

(send-super :init)

(setq \_damping \_\_damping)

(setq \_hiStop \_\_hiStop)

(setq \_loStop \_\_loStop)

(setq \_erp \_\_erp)

(setq \_cfm \_\_cfm)

(setq \_stop\_erp \_\_stop\_erp)

(setq \_stop\_cfm \_\_stop\_cfm)

(setq \_fudge\_factor \_\_fudge\_factor)

(setq \_fmax \_\_fmax)

(setq \_vel \_\_vel)

self)

(:damping

(&optional \_\_damping)

(if \_\_damping (setq \_damping \_\_damping)) \_damping)

(:hiStop

(&optional \_\_hiStop)

(if \_\_hiStop (setq \_hiStop \_\_hiStop)) \_hiStop)

(:loStop

(&optional \_\_loStop)

(if \_\_loStop (setq \_loStop \_\_loStop)) \_loStop)

(:erp

(&optional \_\_erp)

(if \_\_erp (setq \_erp \_\_erp)) \_erp)

(:cfm

(&optional \_\_cfm)

(if \_\_cfm (setq \_cfm \_\_cfm)) \_cfm)

(:stop\_erp

(&optional \_\_stop\_erp)

(if \_\_stop\_erp (setq \_stop\_erp \_\_stop\_erp)) \_stop\_erp)

(:stop\_cfm

(&optional \_\_stop\_cfm)

(if \_\_stop\_cfm (setq \_stop\_cfm \_\_stop\_cfm)) \_stop\_cfm)

(:fudge\_factor

(&optional \_\_fudge\_factor)

(if \_\_fudge\_factor (setq \_fudge\_factor \_\_fudge\_factor)) \_fudge\_factor)

(:fmax

(&optional \_\_fmax)

(if \_\_fmax (setq \_fmax \_\_fmax)) \_fmax)

(:vel

(&optional \_\_vel)

(if \_\_vel (setq \_vel \_\_vel)) \_vel)

(:serialization-length

()

(+

;; float64[] \_damping

(\* 8 (length \_damping)) 4

;; float64[] \_hiStop

(\* 8 (length \_hiStop)) 4

;; float64[] \_loStop

(\* 8 (length \_loStop)) 4

;; float64[] \_erp

(\* 8 (length \_erp)) 4

;; float64[] \_cfm

(\* 8 (length \_cfm)) 4

;; float64[] \_stop\_erp

(\* 8 (length \_stop\_erp)) 4

;; float64[] \_stop\_cfm

(\* 8 (length \_stop\_cfm)) 4

;; float64[] \_fudge\_factor

(\* 8 (length \_fudge\_factor)) 4

;; float64[] \_fmax

(\* 8 (length \_fmax)) 4

;; float64[] \_vel

(\* 8 (length \_vel)) 4

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; float64[] \_damping

(write-long (length \_damping) s)

(dotimes (i (length \_damping))

(sys::poke (elt \_damping i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_hiStop

(write-long (length \_hiStop) s)

(dotimes (i (length \_hiStop))

(sys::poke (elt \_hiStop i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_loStop

(write-long (length \_loStop) s)

(dotimes (i (length \_loStop))

(sys::poke (elt \_loStop i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_erp

(write-long (length \_erp) s)

(dotimes (i (length \_erp))

(sys::poke (elt \_erp i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_cfm

(write-long (length \_cfm) s)

(dotimes (i (length \_cfm))

(sys::poke (elt \_cfm i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_stop\_erp

(write-long (length \_stop\_erp) s)

(dotimes (i (length \_stop\_erp))

(sys::poke (elt \_stop\_erp i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_stop\_cfm

(write-long (length \_stop\_cfm) s)

(dotimes (i (length \_stop\_cfm))

(sys::poke (elt \_stop\_cfm i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_fudge\_factor

(write-long (length \_fudge\_factor) s)

(dotimes (i (length \_fudge\_factor))

(sys::poke (elt \_fudge\_factor i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_fmax

(write-long (length \_fmax) s)

(dotimes (i (length \_fmax))

(sys::poke (elt \_fmax i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_vel

(write-long (length \_vel) s)

(dotimes (i (length \_vel))

(sys::poke (elt \_vel i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; float64[] \_damping

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_damping (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_damping i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_hiStop

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_hiStop (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_hiStop i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_loStop

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_loStop (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_loStop i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_erp

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_erp (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_erp i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_cfm

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_cfm (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_cfm i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_stop\_erp

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_stop\_erp (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_stop\_erp i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_stop\_cfm

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_stop\_cfm (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_stop\_cfm i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_fudge\_factor

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_fudge\_factor (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_fudge\_factor i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_fmax

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_fmax (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_fmax i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_vel

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_vel (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_vel i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;;

self)

)

(setf (get gazebo\_msgs\_new::ODEJointProperties :md5sum-) "1b744c32a920af979f53afe2f9c3511f")

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

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

"# access to low level joint properties, change these at your own risk

float64[] damping # joint damping

float64[] hiStop # joint limit

float64[] loStop # joint limit

float64[] erp # set joint erp

float64[] cfm # set joint cfm

float64[] stop\_erp # set joint erp for joint limit \"contact\" joint

float64[] stop\_cfm # set joint cfm for joint limit \"contact\" joint

float64[] fudge\_factor # joint fudge\_factor applied at limits, see ODE manual for info.

float64[] fmax # ode joint param fmax

float64[] vel # ode joint param vel

")

(provide :gazebo\_msgs\_new/ODEJointProperties "1b744c32a920af979f53afe2f9c3511f")