## **Conversion Namespace Reference**

The namespace **Conversion** provides some utilities to convert from one representation into another. Representations include tf, Eigen::Affine3d, std::vector, geometry\_msgs::Pose, geometry\_msgs::Point, and CRCL. More...

## **Functions**

template <typename ,="" from="" to="" typename=""></typename>	
То	Convert (From f)
	Empty conversion of type from into type to. If called, asserts. More
template<>	
tf::Pose	Convert < Eigen::Affine3d, tf::Pose > (Eigen::Affine3d pose) Convert Eigen::Affine3d into tf::Pose. More
template<>	
tf::Pose	Convert< geometry_msgs::Pose, tf::Pose >
	(geometry_msgs::Pose m)
	Convert geometry_msgs::Pose into tf::Pose. More
template<>	
tf::Quaternion	Convert < Eigen::Quaterniond, tf::Quaternion >
	(Eigen::Quaterniond e)
	Convert Eigen::Quaterniond into tf::Quaternion. More
template<>	
tf::Pose	Convert < tf::Quaternion, tf::Pose > (tf::Quaternion q)
	Convert tf::Quaternion into tf::Pose. More
template<>	
tf::Pose	Convert < tf::Vector3, tf::Pose > (tf::Vector3 t)
	Convert tf::Vector3 into tf::Pose. More
tf::Pose	CreateRPYPose (std::vector< double > ds)
	CreateRPYPose taks array of double and create a tf::Pose. More
tf::Pose	CreatePose (tf::Vector3 axis, double angle)
	Create Pose from a axis and angle rotation representation. More
tf::Quaternion	RPYRadians (double roll, double pitch, double yaw)

	Create Quaternion from a rpy rotation representation designated in radians. More
tf::Quaternion	RPYDegrees (double roll, double pitch, double yaw) Create Quaternion from a rpy rotation representation designated in degrees. More
template<>	
tf::Vector3	Convert < Eigen::Vector3d, tf::Vector3 > (Eigen::Vector3d e) Convert geometry_msgs::Pose into tf::Vector3. More
template<>	
tf::Pose	Convert < Eigen::Vector3d, tf::Pose > (Eigen::Vector3d e) Convert Eigen::Vector3d into tf::Pose. More
template <typename t=""></typename>	
tf::Vector3	<pre>matrixEigenToTfVector (T e) Convert Eigen matrix into tf::Vector3. Example: tf::Vector3 v = matrixEigenToTfVector<eigen::matrix3d>(m);. More</eigen::matrix3d></pre>
template<>	
tf::Matrix3x3	Convert < Eigen::Matrix3d, tf::Matrix3x3 > (Eigen::Matrix3d e) Convert Eigen Matrix3d into tf::Matrix3x3. More
tf::Pose	Identity () Create Identity Pose. More
template<>	
Eigen::Affine3d	Convert < tf::Pose, Eigen::Affine3d > (tf::Pose pose) Convert < tf::Pose, Eigen::Affine3d > converts tf pose into an Eigen affine 4x4 matrix o represent the pose. More
template<>	
Eigen::Affine3d	Convert < geometry_msgs::Pose, Eigen::Affine3d > (geometry_msgs::Pose m)
	Convert geometry_msgs::Pose into an Eigen affine3d 4x4 matrix o represent the pose. Uses tf conversion utilities. More
template<>	
Eigen::Translation3d	(geometry_msgs::Pose pose)
	Convert geometry_msgs::Pose into an Eigen::Translation3d. More
template<>	
Eigen::Vector3d	Convert < tf::Vector3, Eigen::Vector3d > (tf::Vector3 t) Convert tf::Vector3 into an Eigen::Vector3d. More

template<>

Eigen::Affine3d Convert < Eigen::Vector3d, Eigen::Affine3d > (Eigen::Vector3d

translation)

Convert Eigen::Vector3d translation into an Eigen::Affine3d pose.

More...

Eigen::Affine3d **CreateEigenPose** (double zangle)

Create Eigen::Affine3d as an axis angle definition around z axis.

More...

template<>

Eigen::Affine3d Convert < Eigen::Translation3d, Eigen::Affine3d >

(Eigen::Translation3d trans)

Convert Eigen::Translation3d translation into an Eigen::Affine3d

pose. More...

template<>

Eigen::Vector3d Convert< geometry msgs::Point, Eigen::Vector3d >

(geometry msgs::Point point)

Convert geometry\_msgs::Point translation into an Eigen::Vector3d

vector. More...

template<>

Eigen::Affine3d Convert < geometry\_msgs::Point, Eigen::Affine3d >

(geometry\_msgs::Point point)

Convert geometry\_msgs::Point translation into an Eigen::Affine3d

pose. More...

template<>

geometry msgs::Pose Convert < tf::Pose, geometry msgs::Pose > (tf::Pose m)

Convert tf::Pose pose into an geometry\_msgs::Pose pose. More...

template<>

geometry\_msgs::Pose Convert< geometry\_msgs::Point, geometry\_msgs::Pose >

(geometry\_msgs::Point point)

Convert geometry\_msgs::Point point into an geometry\_msgs::Pose

pose. More...

template<>

geometry msgs::Point Convert< Eigen::Vector3d, geometry msgs::Point >

(Eigen::Vector3d point)

Convert Eigen::Vector3d point into an geometry msgs::Point

position vector. More...

template<>

geometry\_msgs::Pose Convert< Eigen::Affine3d, geometry\_msgs::Pose >

(Eigen::Affine3d e)

Convert Eigen::Affine3d pose into an geometry\_msgs::Pose pose.

More...

template<>

geometry msgs::Point Convert < Eigen::Affine3d, geometry msgs::Point >

(Eigen::Affine3d pose)

Convert Eigen::Affine3d pose into an geometry\_msgs::Point

translation element. More...

template<>

JointState Convert< std::vector< double >, JointState > (std::vector<

double > src)

Convert array of std::vector<double> doubles into an JointState

position, but blanking velcity, and effort. More...

## **Detailed Description**

The namespace **Conversion** provides some utilities to convert from one representation into another. Representations include tf, Eigen::Affine3d, std::vector, geometry\_msgs::Pose, geometry\_msgs::Point, and CRCL.

Clearly there may be faster const references, but they require special line to convert, cannot be done in line since you cannot pass a const reference to a constructor on the stack in g++ unless you override a warning.

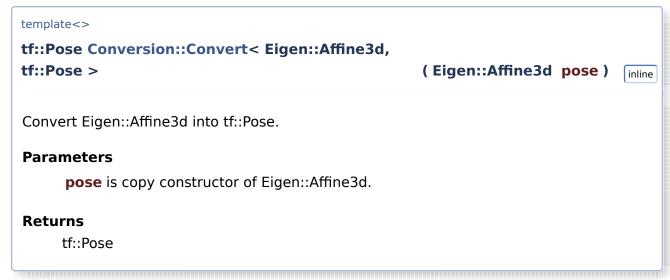
For g++, compilation would be faster if these conversion routines were placed in source file (cpp) OR you used precompiled header in g++. here is a "silent" error when exceeding precompiled header limits in g++. (Or was at one time).

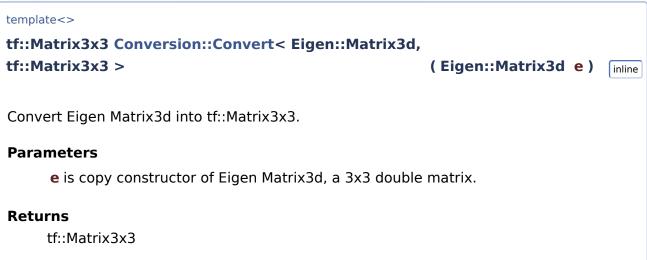
## **Function Documentation**

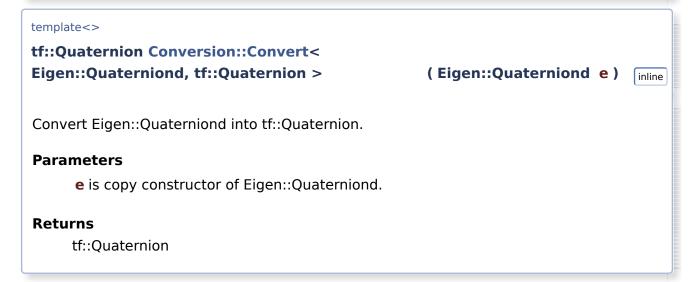
template<typename From , typename To > To Conversion::Convert (From f) inline Empty conversion of type from into type to. If called, asserts. **Parameters from** is defined in the template corresponding typename. **Returns** to is defined in the template corresponding typename template<> geometry msgs::Point Conversion::Convert< Eigen::Affine3d, geometry msgs::Point > (Eigen::Affine3d pose) Convert Eigen::Affine3d pose into an geometry msgs::Point translation element. **Parameters** e is Eigen::Affine3d defining pose. **Returns** geometry msgs::Point translation element. template<> geometry\_msgs::Pose Conversion::Convert< Eigen::Affine3d, geometry\_msgs::Pose > (Eigen::Affine3d e) [inline] Convert Eigen::Affine3d pose into an geometry msgs::Pose pose. **Parameters** e is Eigen::Affine3d defining equivalent pose. **Returns** 

5 of 15 11/21/2016 03:50 PM

geometry\_msgs::Pose pose.







Eigen::Affine3d Conversion::Convert <
Eigen::Translation3d, Eigen::Affine3d > (Eigen::Translation3d trans) inline

Convert Eigen::Translation3d translation into an Eigen::Affine3d pose.

Parameters
 t is translation defined as a Eigen::Translation3d.

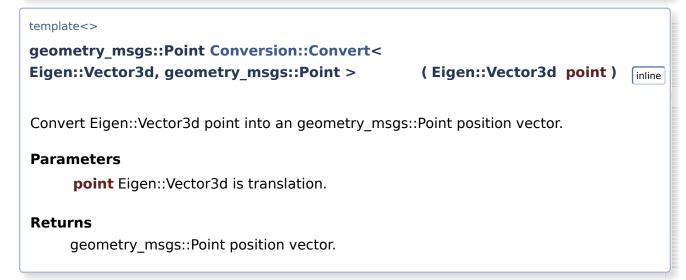
Returns
 Eigen::Affine3d pose

Eigen::Affine3d Conversion::Convert <
Eigen::Vector3d, Eigen::Affine3d > (Eigen::Vector3d translation) inline

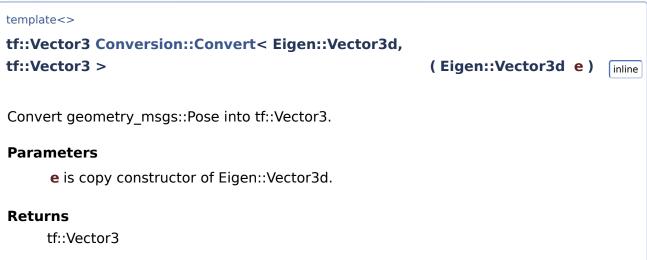
Convert Eigen::Vector3d translation into an Eigen::Affine3d pose.

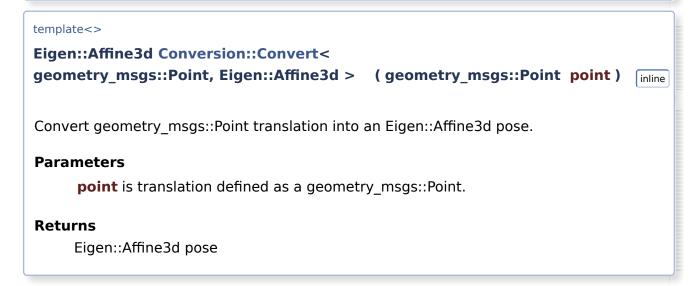
Parameters
 translation is defined as a Eigen::Vector3d.

Returns
 Eigen::Affine3d pose









```
Eigen::Vector3d Conversion::Convert<
geometry_msgs::Point, Eigen::Vector3d > (geometry_msgs::Point point) inline

Convert geometry_msgs::Point translation into an Eigen::Vector3d vector.

Parameters
    point is translation defined as a geometry_msgs::Point.

Returns
    Eigen::Vector3d vector
```

```
template<>
Eigen::Affine3d Conversion::Convert<
geometry_msgs::Pose, Eigen::Affine3d > (geometry_msgs::Pose m) inline

Convert geometry_msgs::Pose into an Eigen affine3d 4x4 matrix o represent the pose.
Uses tf conversion utilities.

Parameters
m is defined as a geometry_msgs::Pose..

Returns
Eigen Affine3d pose
```

```
tf::Pose Conversion::Convert<
geometry_msgs::Pose, tf::Pose > (geometry_msgs::Pose m) inline

Convert geometry_msgs::Pose into tf::Pose.

Parameters
pose is copy constructor of geometry_msgs::Pose.

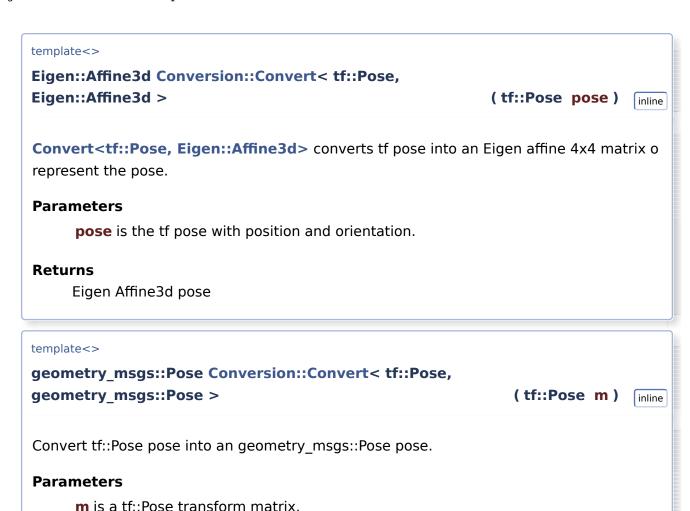
Returns
tf::Pose
```

```
JointState Conversion::Convert< std::vector<
double >, JointState > (std::vector< double > src) inline

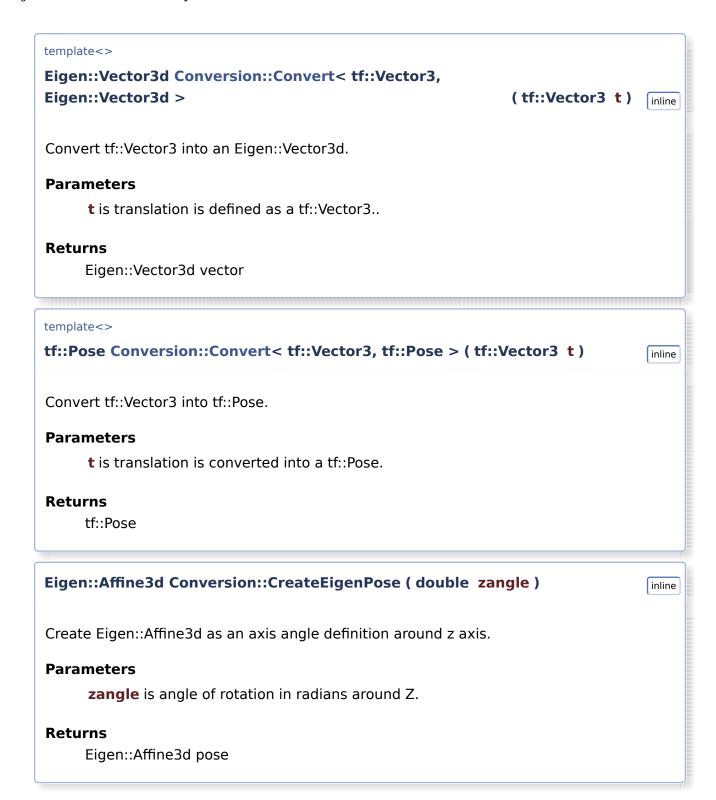
Convert array of std::vector<double> doubles into an JointState position, but blanking velcity, and effort.

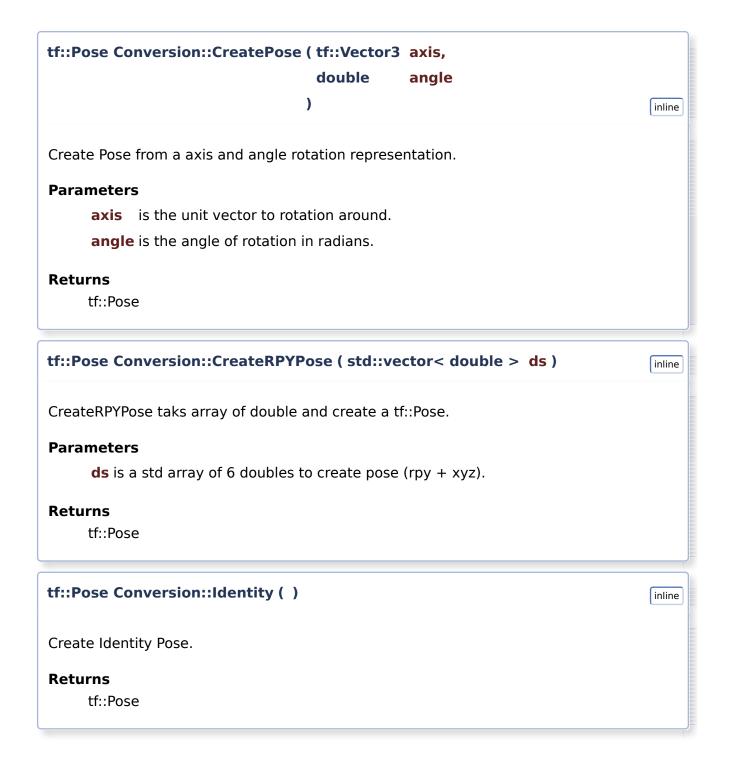
Parameters
src is a std::vector of doubles defining the value for each joint.

Returns
sensor_msgs::JointState_<std::allocator<void>> definition.
```



**Returns**geometry\_msgs::Pose pose





```
template<typename T >
tf::Vector3 Conversion::matrixEigenToTfVector ( T e )
                                                                                   inline
Convert Eigen matrix into tf::Vector3. Example: tf::Vector3 v =
matrixEigenToTfVector<Eigen::Matrix3d>(m);.
Parameters
      e is copy constructor of Eigen Matrix, either 3x3, 4x4, double or float.
Returns
     tf::Vector3
tf::Quaternion Conversion::RPYDegrees (double roll,
                                             double pitch,
                                             double yaw
                                            )
                                                                                   inline
Create Quaternion from a rpy rotation representation designated in degrees.
Parameters
      roll rotation around x axis in degrees.
      pitch rotation around y axis in degrees.
      yaw rotation around z axis in degrees.
Returns
     tf::Quaternion
```

Generated on Mon Nov 21 2016 15:49:46 for My Project by

