AEROSPACE STANDARD	AS6057	REV. A
	Issued 2011-03 Revised 2014-06 Superseding AS6057	

(R) JAUS Manipulator Service Set

RATIONALE

This document, the JAUS Manipulator Service Set (AS6057A), defines a message-passing interface for services commonly found in serial manipulator systems. These services represent the platform-independent capabilities common across all serial manipulator types. Additional capabilities are specified in the JAUS Core Service Set (AS5710) and are frequently referenced herein. The Revision A publication addresses several technical inconsistencies in the original document, including proper reference to Core 1.1 services and separation of overlapping input vocabularies, while addressing backward compatibility with existing implementations.

INTRODUCTION

The primary goal of the JAUS Manipulator Service Set is logical interoperability between communicating elements in an unmanned system. To this end, each service defines the messages (vocabulary) and protocol (rules) for data exchange. This logical interoperability is independent of the physical transport, and it is expected that a Transport Standard, such as the JAUS/SDP Transport Specification [AS5669], is used in conjunction with this specification.

Each service in the JAUS Manipulator Service Set can be described using the JAUS Service Interface Definition Language [JSIDL]. JSIDL creates a formal schema based on Relax NG Compact [rng] that allows for validation of each service definition described herein. Although knowledge of JSIDL is not required to understand or implement this Specification, it is highly recommended for supporting context. For convenience, the JAUS Manipulator Service Set contains both a text based and XML based representation for each service.

The AS6057 Standard does not replace the latest JAUS Reference Architecture Version 3.3 [RA33P1, RA33P2, RA33P3], but is a direct evolution of that work. This Standard has been carefully designed to allow for the simplest possible migration of RA 3.3 implementations to a services-based framework. Even though the notion of services has come to define formal interfaces between components, the message sets in those services trace back to identical messages in the Reference Architecture.

This document uses a number of conventions to simplify the text. All names are given in Camel Case. Names start with upper case, while reference names start with a lower case.

The tables and diagrams in this document are hand-transcribed from the JSIDL XML specification in the Appendixes. In case of transcription errors, the XML specification should be considered correct.

TO PLACE A DOCUMENT ORDER: **Tel: 877-606-7323 (inside USA and Canada)**
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: **http://www.sae.org**

SAE values your input. To provide feedback on this Technical Report, please visit
http://www.sae.org/technical/standards/AS6057A

TABLE OF CONTENTS

1. SCOPE

This document defines a set of standard application layer interfaces called JAUS Manipulator Services. JAUS Services provide the means for software entities in an unmanned system or system of unmanned systems to communicate and coordinate their activities. The Manipulator Services represent platform-independent capabilities commonly found across domains and types of unmanned systems. At present, twenty-five (25) services are defined in this document. These services are categorized as:

Low Level Manipulator Control Services – The one service in this category allows for low-level command of the manipulator joint actuation efforts. This is an open-loop command that could be used in a simple tele-operation scenario. The service in this category is listed as follows:

- Primitive Manipulator Service

Manipulator Sensor Services – These services, when queried, return instantaneous sensor data. Three services are defined that return respectively joint positions, joint velocities, and joint torques or forces. Two additional services return the end effector pose and velocity with respect to the manipulator base coordinate frame. The services in this category are listed as follows:

- Manipulator Joint Position Sensor Service
- Manipulator Joint Velocity Sensor Service
- Manipulator Joint Force/Torque Sensor Service
- Manipulator End Effector Pose Sensor Service
- Manipulator End Effector Velocity State Sensor Service

Mid Level Position and Velocity Driver Services – These services take as inputs the desired joint positions, the desired joint velocities, the desired end effector pose, or the desired end effector velocity state. Closed-loop control is implied. The services in this category are listed as follows:

- Manipulator Joint Position Driver Service
- Manipulator Joint Position List Driver Service
- Manipulator End Effector Pose Driver Service
- Manipulator End Effector Pose List Driver Service
- Manipulator Joint Velocity Driver Service
- Manipulator End Effector Velocity State Driver Service
- Manipulator Actuator Force/Torque Driver Service

Pan Tilt Driver Services –These services provide an interface to a specific pan tilt mechanism. The pan tilt mechanism is a two degree of freedom serial manipulator. It is comprised of two revolute joints whose axes intersect and are perpendicular. The services in this category are listed as follows:

- Primitive Pan Tilt Service
- Pan Tilt Joint Position Sensor Service
- Pan Tilt Joint Position Driver Service

- Pan Tilt Joint Velocity Sensor Service
- Pan Tilt Joint Velocity Driver Service

End Effector Driver Services – This service provides an interface to a specific end effector. The end effector is a one-degree of freedom serial manipulator, usually mounted on the end of a *n*-degree of freedom serial manipulator. Examples include grippers, screw drivers, or other tools. The services in this category are listed as follows:

- Primitive End Effector Service

Each service is described by a JAUS Service Definition (JSD) which specifies the message set and protocol required for compliance. Each JSD is fully compliant with the JAUS Service Interface Definition Language [JSIDL].

1.1 Purpose

The purpose of this document is to facilitate interoperation of unmanned vehicle systems, subsystems, and payloads by standardization of the message set and associated protocol.

1.2 JAUS Core Service Set

The JAUS Service Definitions defined herein make use of the inheritance functionality provided by JSIDL to incorporate capabilities as specified by the JAUS Core Service Set (AS5710). These documents must be used together to define a complete service. It is important to note that details related to 'Message Serialization' and 'Understanding Protocol Descriptions' can be found in the JAUS Core Service Set document, and are not repeated here.

1.3 Compliance

The JAUS Manipulator Service Set must support compliance assessment. To do so, this specification must be sufficiently precise to enable the "compliant"/"not compliant" distinction to be made independently of the underlying transport mechanism. It is important to note that implementations are considered compliant to individual Service Definitions within this Specification; it is not necessary that a single entity realize each service to be considered compliant.

1.4 Document Organization

The layout of this document is as follows. Section 2 lists external references used throughout the specification. Section 3 specifies common conventions and definitions. Section 4 specifies the JAUS Service Definition for each of the manipulator services, with particular emphasis on the description, assumptions, message set, and protocol behavior. Section 5 describes message encoding for each message set. Finally, Appendix A and Appendix B contain the complete JSIDL representation for each service and their associated message set.

2. REFERENCES

2.1 Applicable Documents

The following publications form a part of this document to the extent specified herein. The applicable issue of all publications shall be the issue in effect on the date of the publication for this specification, unless otherwise noted. In the event of conflict between the text of this document and references cited herein, the text of this document takes precedence.

Nothing in this document, however, supersedes applicable laws and regulations.

2.1.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), Web address: www.sae.org.

AIR5665 Architecture Framework for Unmanned Systems

AS5669 JAUS / SDP Transport Specification, Revision B

AS5684 JAUS Service Interface Definition Language

AS5710 JAUS Core Service Set, Revision A

AS6009 JAUS Mobility Service Set

2.1.2 JAUS Technical References

RA33P1 JAUS Reference Architecture Specification, Volume II, Part 1, Architecture Framework, Version 3.3, June 22, 2007

RA33P2 JAUS Reference Architecture Specification, Volume II, Part 2, Message Definition, Version 3.3, June 22, 2007

RA33P3 JAUS Reference Architecture Specification, Volume II, Part 3, Message Set, Version 3.3, June 22, 2007

2.1.3 Other Publications

[att] Decimal Expansion of PI [http://www.research.att.com/~njas/sequences/A000796]

[crane] Carl D. Crane. Joseph Duffy. Kinematics Analysis of Robot Manipulators. Cambridge University Press. 2008.

[rng] Relax NG: [http://www.oasis-open.org/committees/relax-ng/spec-20011203.html] Standard lightweight XML schema language.

2.2 List of Acronyms

ID Identifier

API Application Programming Interface

ASCII American Standard Code for Information Interchange

IEEE Institute of Electrical and Electronics Engineers

JAUS Joint Architecture for Unmanned Systems

JSD JAUS Service Definition

JSIDL JAUS Service (Interface) Definition Language

RA (JAUS) Reference Architecture

UML Unified Modeling Language

URL Uniform Resource Locator

URN Uniform Resource Name

URI Uniform Resource Identifier

UUID Universally Unique Identifier

XML Extensible Markup Language

3. COMMON CONVENTIONS

3.1 Constant values

In the context of this Standard, PI (π) is defined as 3.14159265358979323846 [att].

3.2 Definition of Coordinate Systems

3.2.1 Vehicle Coordinate System

This coordinate system is defined as attached to the vehicle frame. Typically, the X-axis points in the forward direction and the Z-axis points downward. The Y-axis is defined so as to have a right-handed coordinate system, i.e., **i** × **j** = **k** where **i**, **j**, and **k** are unit vectors along the X, Y, and Z coordinate axes.

3.2.2 Manipulator Base Coordinate System

This coordinate system is attached to the vehicle base just as the vehicle coordinate system is. The Z axis is along S_1 and the direction of the X-axis is user defined, but fixed with respect to the vehicle frame (see Figure 1). Since the manipulator base coordinate system and the vehicle coordinate system are both attached to the vehicle base, the transformation matrix that describes the relative position and orientation of these two coordinate systems will be constant. Cases in which a manipulator may be attached to the end of another manipulator are beyond the scope of the current document.

3.2.3 End Effector Coordinate System

This coordinate system is attached to the last link of the serial manipulator. The origin is located at the point that is a distance S_n (S_6 for a six axis manipulator) along the last joint axis vector (S_6 for a six axis manipulator) from the intersection of the lines along the last joint axis and the preceding link axis (S_6 and a_{56} for a six axis manipulator). The Z-axis is along the S_n vector as shown in Figure 1. The direction of the X-axis is user defined, but of course must be perpendicular to the Z-axis. The Y-axis is defined by the right-hand rule.

3.3 End Effector Pose

The End Effector Pose represents the position of the tool tip and the orientation of the end effector coordinate system.

3.4 Quaternion

This specification uses quaternion-based notation to reflect three dimensional rotation of a coordinate system as measured with respect to a base coordinate system. This rotation is defined by a unit quaternion (d ; a, b, c) where the quaternion q is expressed as q = d + ai +bj + ck where i, j, and k represent the unit quaternions (0; 1,0,0), (0; 0,1,0), and (0;0,0,1) respectively.

3.5 Manipulator Linkage Notation

Manipulators are often used on Unmanned Systems to alter the environment. To standardize the method of referencing manipulator links (or joints), the following figures illustrate the notation that will be used later in the document. Figure 2 illustrates the parameters used for link ij. Figure 3 and Figure 4 illustrate additional parameters used for rotational joints and prismatic joints respectively [crane].

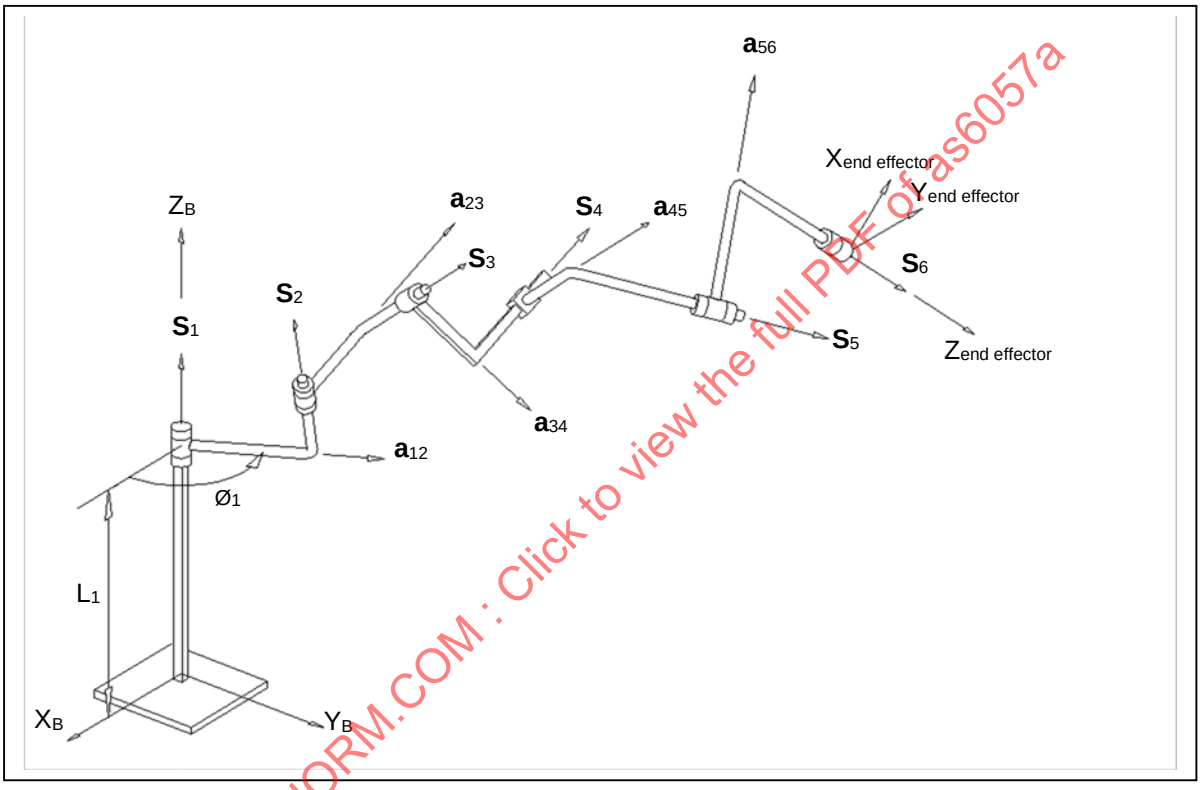
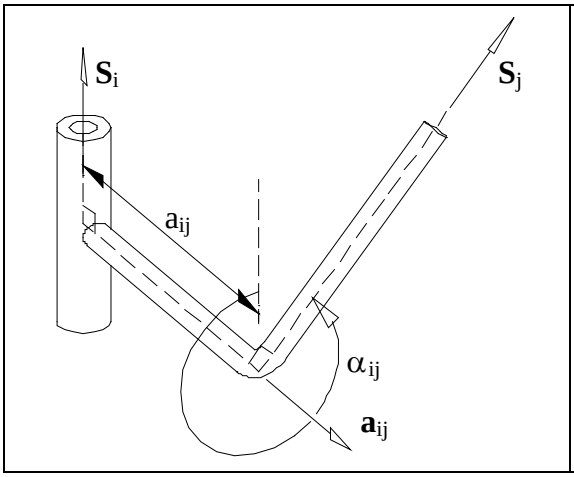


FIGURE 1 - MANIPULATOR BASE COORDINATE SYSTEM AND END EFFECTOR COORDINATE SYSTEM



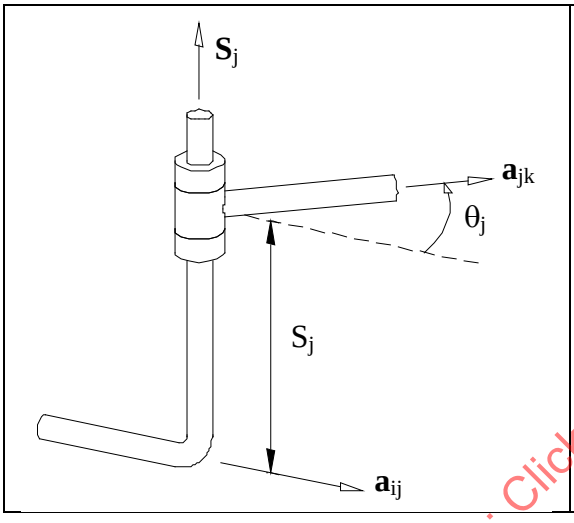
a_{ij} – link length
α_{ij} – twist angle

$\mathbf{a}_{ij}$ – unit vector along link ij
$\mathbf{S}_i$ – unit vector along joint axis i
$\mathbf{S}_j$ – unit vector along joint axis j

Link length a_{ij} is measured as the perpendicular distance between joint axis i and joint axis j along the vector $\mathbf{a}_{ij}$. Note that it can have a negative value.

Twist angle α_{ij} is the angle between $\mathbf{S}_i$ and $\mathbf{S}_j$ measured in a right hand sense about $\mathbf{a}_{ij}$.

FIGURE 2 - MANIPULATOR LINKAGE PARAMETERS FOR LINK IJ



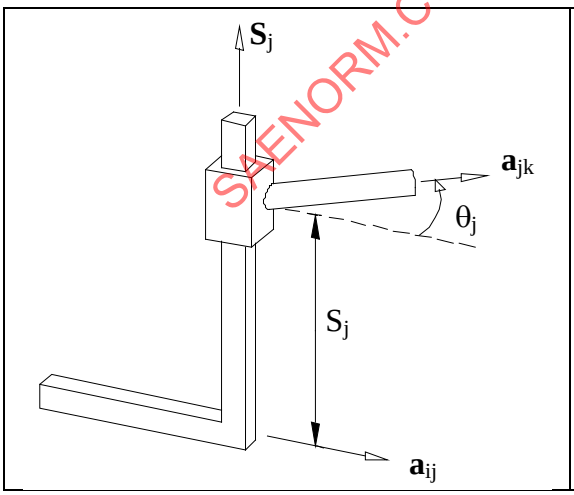
S_j – constant joint offset distance
θ_j – variable joint angle

$\mathbf{a}_{ij}$ – unit vector along link ij
$\mathbf{a}_{jk}$ – unit vector along link jk
$\mathbf{S}_j$ – unit vector along joint axis j

Joint offset S_j is measured as the perpendicular distance between link a_{ij} and link a_{jk} along the vector $\mathbf{S}_j$. Note that it can have a negative value.

Joint angle θ_j is the angle between $\mathbf{a}_{ij}$ and $\mathbf{a}_{jk}$ measured in a right hand sense about $\mathbf{S}_j$.

FIGURE 3 - MANIPULATOR LINKAGE PARAMETERS FOR REVOLUTE JOINT J



S_j – variable joint offset distance
θ_j – constant joint angle

$\mathbf{a}_{ij}$ – unit vector along link ij
$\mathbf{a}_{jk}$ – unit vector along link jk
$\mathbf{S}_j$ – unit vector along joint axis j

Joint offset S_j is measured as the perpendicular distance between link a_{ij} and link a_{jk} along the vector $\mathbf{S}_j$. Note that it can have a negative value.

Joint angle θ_j is the angle between $\mathbf{a}_{ij}$ and $\mathbf{a}_{jk}$ measured in a right hand sense about $\mathbf{S}_j$.

FIGURE 4 - MANIPULATOR LINKAGE PARAMETERS FOR PRISMATIC JOINT J

4. SERVICE DEFINITIONS

The following subsections provide a textual definition for each Service Definition in the JAUS Manipulator Service Set. Additional information on interpreting the service definition elements may be found in [JSIDL].

4.1 Manipulator Specification Service

name=ManipulatorSpecificationService
version=2.0
id= urn:jaus:jss: manipulator:ManipulatorSpecificationService

Inherits-from Events
  name=Events
  id= urn:jaus:jss:core: Events
  version=1.1

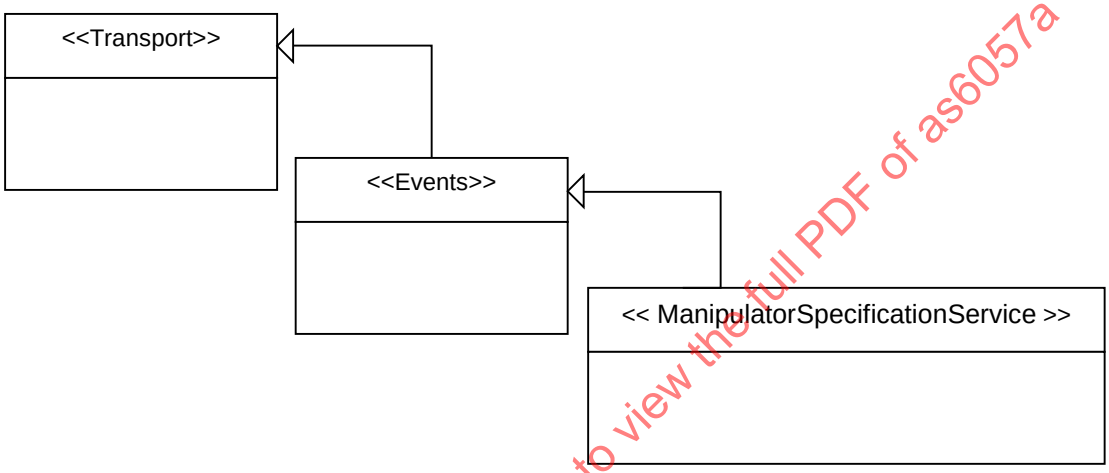


FIGURE 5 - MANIPULATOR SPECIFICATION SERVICE

Description

This service is used to describe a manipulator arm. When queried, the service will reply with a description of the manipulator's specification parameters, axes range of motion, and axes velocity limits. The notations used to describe these data are documented in many popular text books on robotics and were previously presented in Section 3.

The mechanism specification parameters as reported by the Report Manipulator Specifications Message consist of the number of joints, the type of each joint (either revolute or prismatic), the link description parameters for each link (link length and twist angle as shown in Figure 2), the constant joint parameter value (offset for a revolute joint (see Figure 3), and joint angle for a prismatic joint (see Figure 4)). The minimum and maximum allowable value for each joint and the maximum velocity for each joint follow this information.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 1 lists the vocabulary of the Manipulator Specification Service.

TABLE 1 - MANIPULATOR SPECIFICATION SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2600h	QueryManipulatorSpecifications	False
Output Set		
4600h	ReportManipulatorSpecifications	False

Protocol Behavior

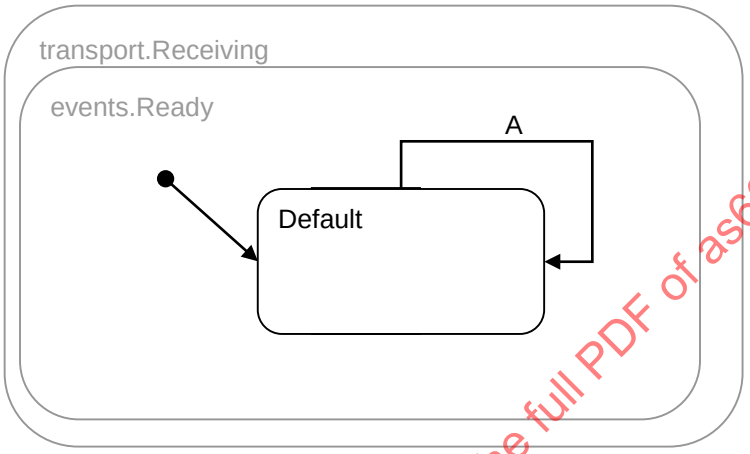


FIGURE 6 - MANIPULATOR SPECIFICATION SERVICE PROTOCOL BEHAVIOR

TABLE 2 - MANIPULATOR SPECIFICATION SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryManipulatorSpecifications		sendReportManipulatorSpecifications

TABLE 3 - MANIPULATOR JOINT POSITION SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportManipulatorSpecifications	Send a Report Manipulator Specs message.

4.2 Primitive Manipulator Service

name= PrimitiveManipulator
version=2.0
id= urn:jaus:jss:manipulator: PrimitiveManipulator

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

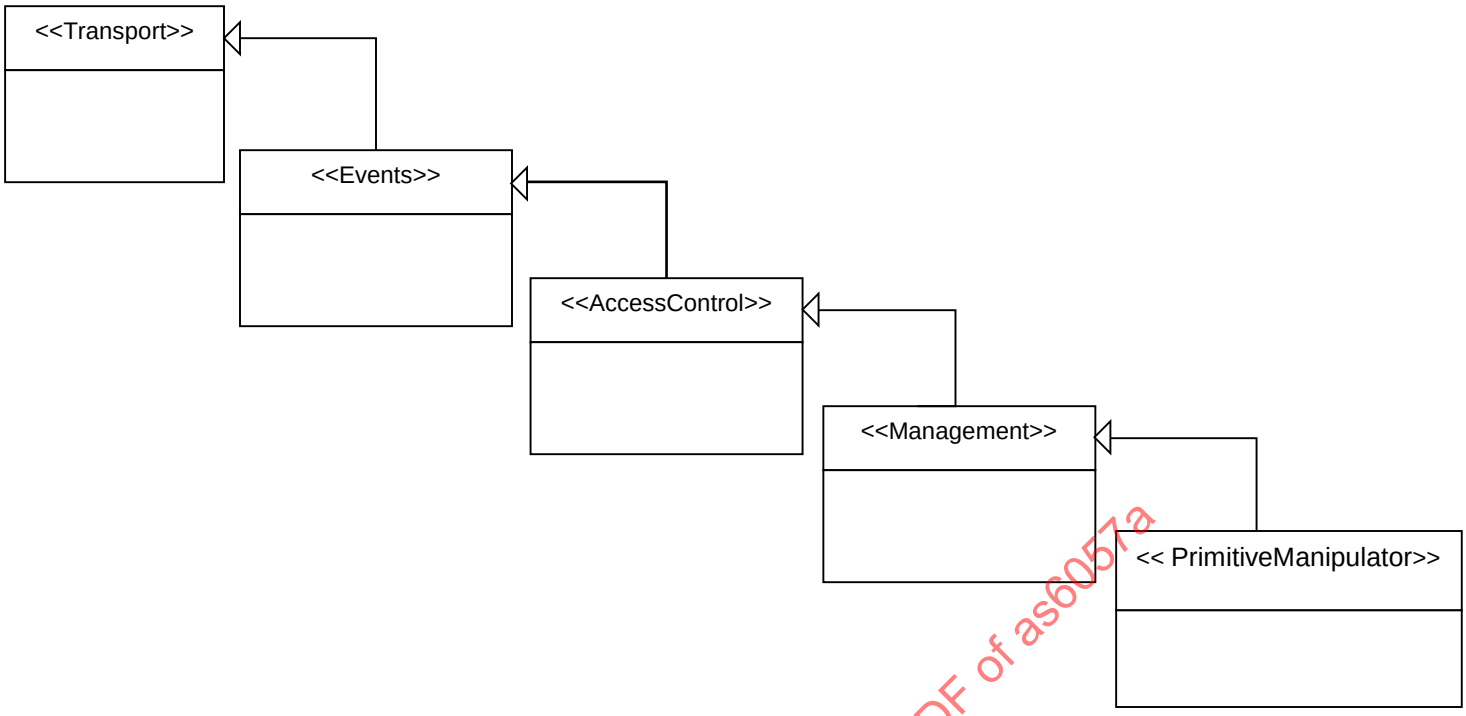


FIGURE 7 - PRIMITIVE MANIPULATOR SERVICE

Description

This service is the low level interface to a manipulator arm. Motion of the arm is accomplished via the Set Joint Effort message. In this message, each actuator is commanded to move with a percentage of maximum effort.

If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service.

Assumptions

Messages may be delayed, lost, or reordered.

Vocabulary

Table 4 lists the vocabulary of the Primitive Manipulator Service.

TABLE 4 - PRIMITIVE MANIPULATOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0601h	*SetJointEffort*	True
2601h	QueryJointEffort	False
Output Set		
4601h	ReportJointEffort	False

Protocol Behavior

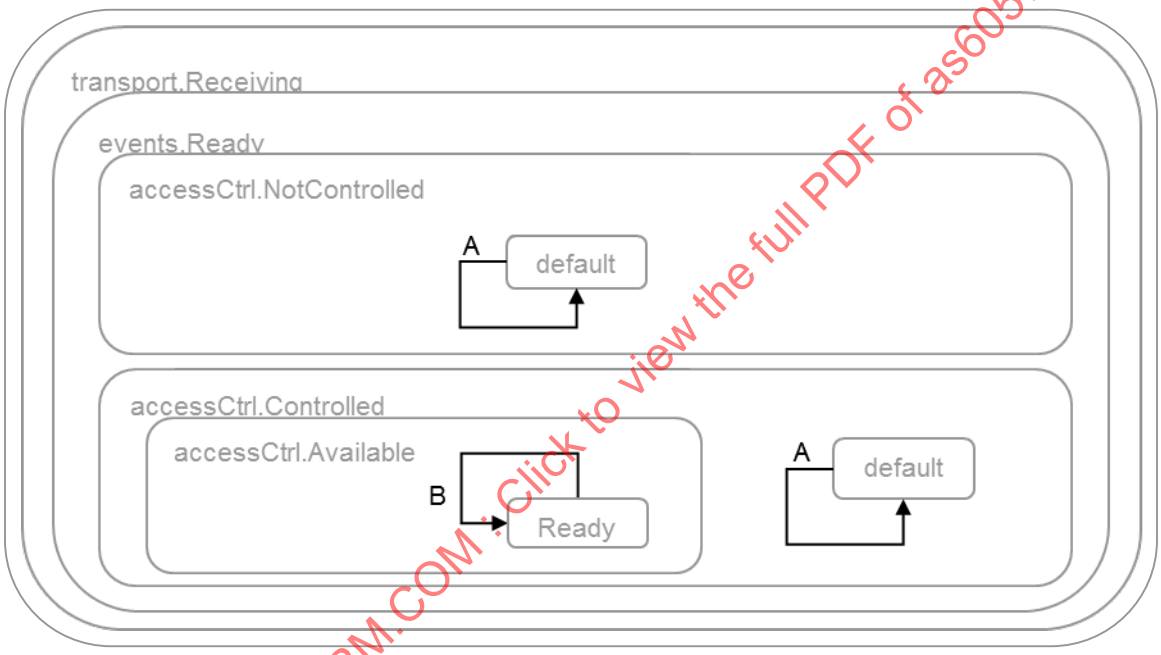


FIGURE 8 - PRIMITIVE MANIPULATOR SERVICE PROTOCOL BEHAVIOR

TABLE 5 - PRIMITIVE MANIPULATOR SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryJointEffort		sendReportJointEffort
B	SetJointEffort	*management.accessCtrl .isControllingClient*	*setJointEffort*

TABLE 6 - PRIMITIVE MANIPULATOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportJointEffort	Send a Report Joint Effort message.
setJointEffort	Set the joint motion efforts for the manipulator. The manipulator joints move accordingly.

4.3 Manipulator Joint Position Sensor Service

name= ManipulatorJointPositionSensor
version=2.0
id= urn:jaus:jss: manipulator: ManipulatorJointPositionSensor

Inherits-from Events
  name=Events
  id= urn:jaus:jss:core: Events
  version=1.1

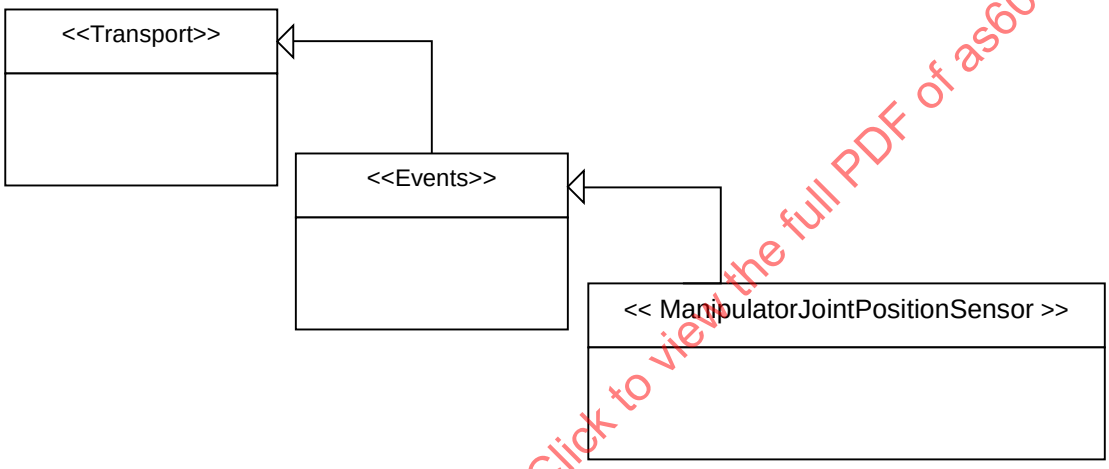


FIGURE 9 - MANIPULATOR JOINT POSITION SENSOR SERVICE

Description

The function of the Joint Position Sensor Service is to report the values of manipulator joint positions when queried. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 7 lists the vocabulary of the Manipulator Joint Position Sensor Service.

TABLE 7 - MANIPULATOR JOINT POSITION SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2602h	QueryJointPosition	False
Output Set		
4602h	ReportJointPosition	False

Protocol Behavior

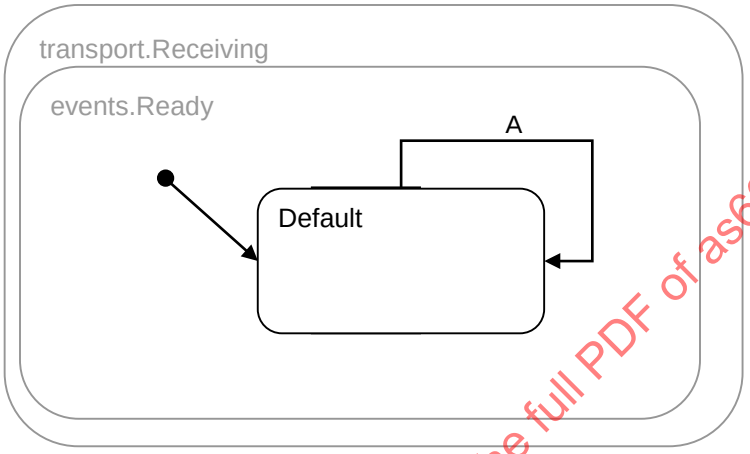


FIGURE 10 - MANIPULATOR JOINT POSITION SENSOR PROTOCOL BEHAVIOR

TABLE 8 - MANIPULATOR JOINT POSITION SENSOR SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryJointPosition		sendJointPosition

TABLE 9 - MANIPULATOR JOINT POSITION SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendJointPosition	Send Report Joint Position message to the service that sent the query.

4.4 Manipulator Joint Velocity Sensor Service

name= ManipulatorJointVelocitySensor
version=2.0
id= urn:jaus:jss: manipulator: ManipulatorJointVelocitySensor

Inherits-from Events
  name=events
  id= urn:jaus:jss:core: Events
  version=1.1

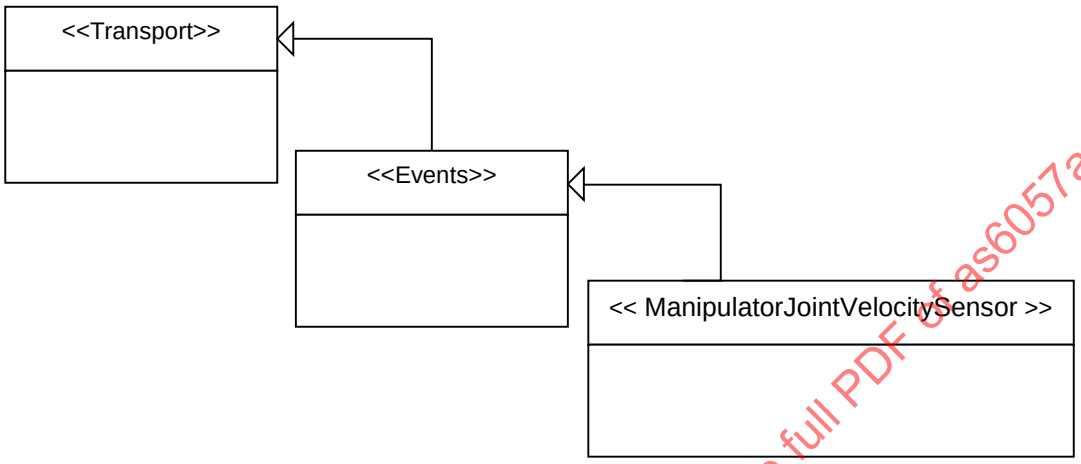


FIGURE 11 - MANIPULATOR JOINT VELOCITY SENSOR SERVICE

Description

The function of the Joint Velocity Sensor Service is to report the values of manipulator joint velocities when queried. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 10 lists the vocabulary of the Manipulator Joint Velocity Sensor Service.

TABLE 10 - MANIPULATOR JOINT VELOCITY SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2603h	QueryJointVelocity	False
Output Set		
4603h	ReportJointVelocity	False

Protocol Behavior

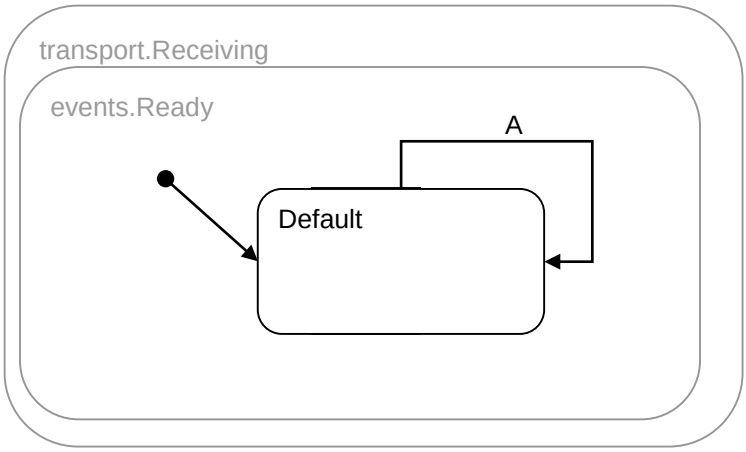


FIGURE 12 - MANIPULATOR JOINT VELOCITY SENSOR PROTOCOL BEHAVIOR

TABLE 11 - MANIPULATOR JOINT VELOCITY SENSOR SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryJointVelocity		sendJointVelocity

TABLE 12 - MANIPULATOR JOINT VELOCITY SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendJointVelocity	Send ReportJointVelocity message to the service that sent the query.

4.5 Manipulator Joint Force/Torque Sensor Service

name= ManipulatorJointForceTorqueSensor
version=2.0
id= urn:jaus:jss: manipulator: ManipulatorJointForceTorqueSensor

Inherits-from Events
  name=events
  id= urn:jaus:jss:core: Events
  version=1.1

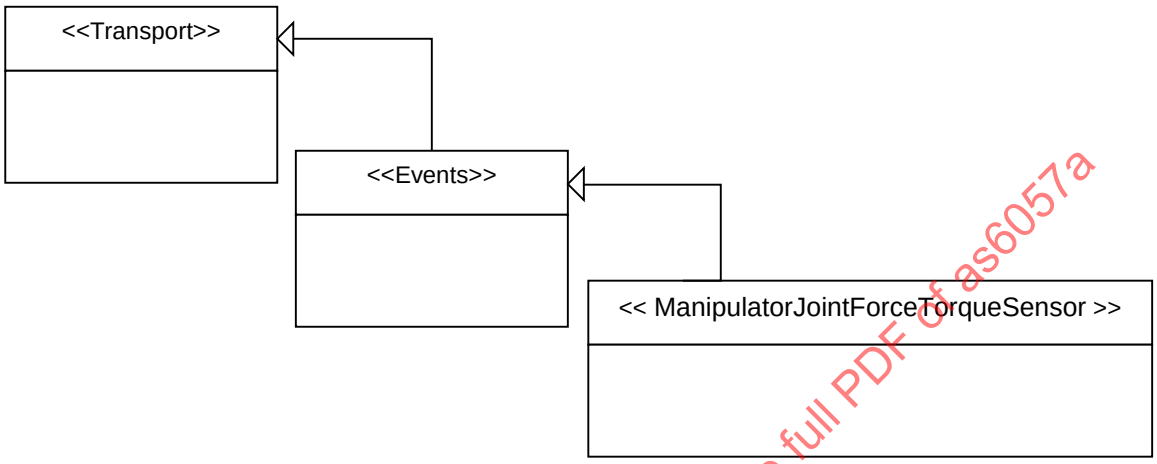


FIGURE 13 - MANIPULATOR JOINT FORCE/TORQUE SENSOR SERVICE

Description

The function of the Joint Force/Torque Sensor is to report the values of instantaneous torques (for revolute joints) and forces (for prismatic joints) that are applied at the individual joints of the manipulator kinematic model when queried. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 13 lists the vocabulary of the Manipulator Force Torque Sensor Service.

TABLE 13 - MANIPULATOR JOINT FORCE TORQUE SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2605h	QueryJointForceTorque	False
Output Set		
4605h	ReportJointForceTorque	False

Protocol Behavior

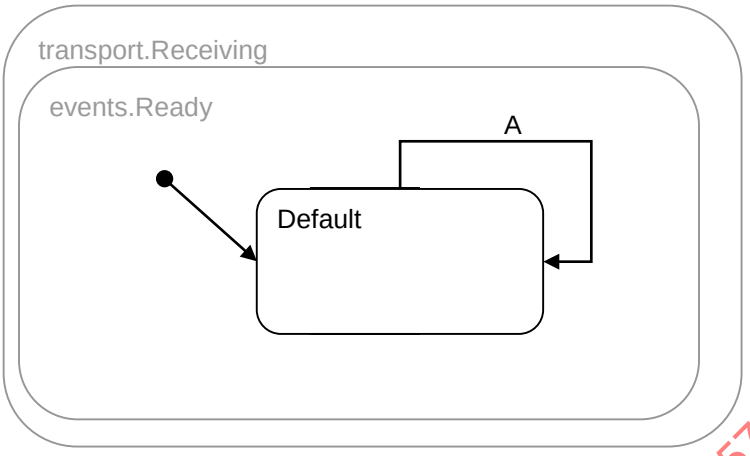


FIGURE 14 - MANIPULATOR JOINT FORCE TORQUE SENSOR PROTOCOL BEHAVIOR

TABLE 14 - MANIPULATOR JOINT FORCE TORQUE SENSOR SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryJointForceTorque		sendJointForceTorque

TABLE 15 - MANIPULATOR JOINT FORCE TORQUE SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendJointForceTorque	Send ReportJointForceTorques message to the service that sent the query.

4.6 Manipulator Tool Offset Service

name= ManipulatorToolOffsetService
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorToolOffsetService

Inherits-from AccessControl
  name=accessControl
  id= urn:jaus:jss:core:AccessControl
  version=1.1

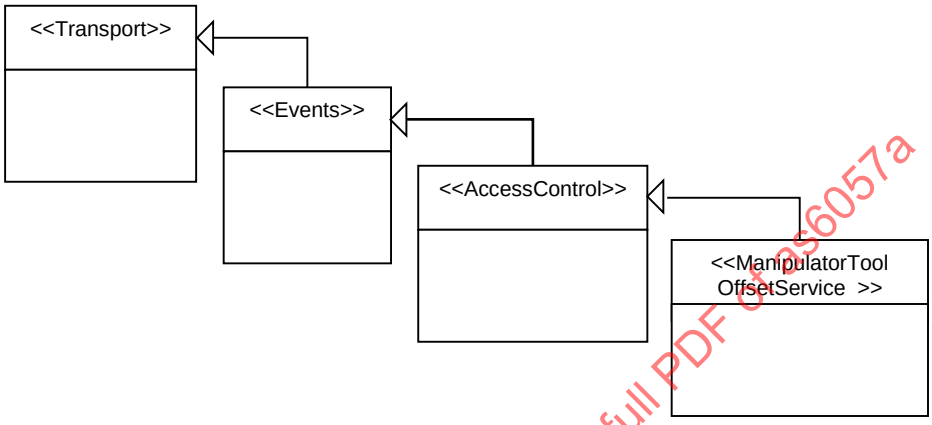


FIGURE 15 - MANIPULATOR TOOL OFFSET SERVICE

Description

The function of the Manipulator Tool Offset Service is to configure the position offset of any tool attached to the manipulator flange.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 16 lists the vocabulary of the Manipulator Tool Offset Service.

TABLE 16 - MANIPULATOR TOOL OFFSET SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0604h	*SetToolOffset*	True
2604h	QueryToolOffset	False
Output Set		
4604h	ReportToolOffset	False

Protocol Behavior

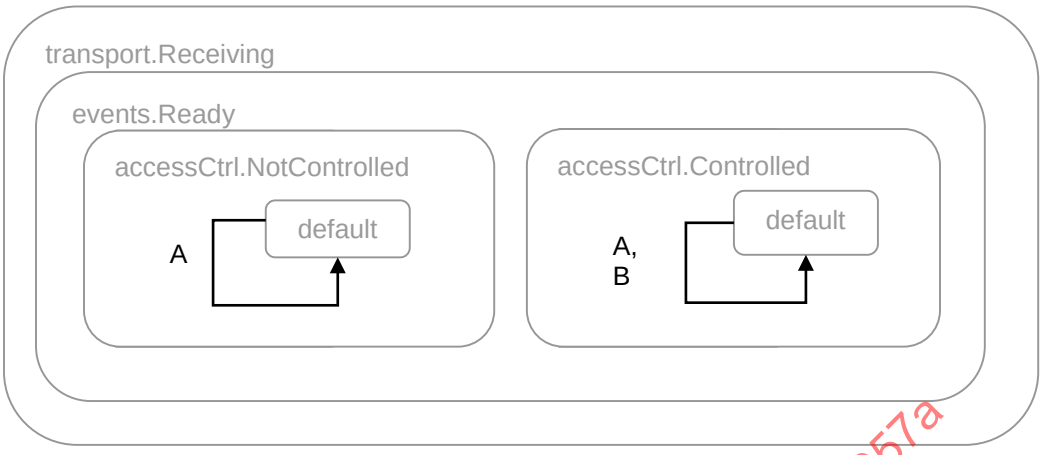


FIGURE 16 - MANIPULATOR TOOL OFFSET SERVICE PROTOCOL BEHAVIOR

TABLE 17 - MANIPULATOR TOOL OFFSET SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryToolOffset		sendReportToolOffset
B	*SetToolOffset*	*accessCtrl.isControllingClient*	*setToolOffset*

TABLE 18 - MANIPULATOR TOOL OFFSET SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportToolOffset	Send a Report Tool Offset message.
setToolOffset	Set the location of the tool tip as measured in the end effector coordinate system.

4.7 Manipulator End Effector Pose Sensor Service

name= ManipulatorEndEffectorPoseSensor
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorEndEffectorPoseSensor

Inherits-from AccessControl
  name=accessControl
  id= urn:jaus:jss:core:AccessControl
  version=1.1

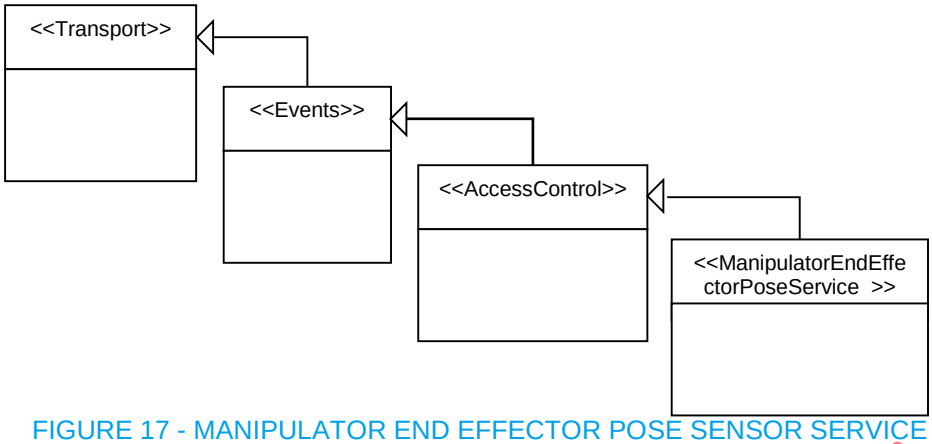


FIGURE 17 - MANIPULATOR END EFFECTOR POSE SENSOR SERVICE

Description

The function of the End Effector Pose Sensor Service is to report the position and orientation of the tool tip with respect to the manipulator base coordinate system. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service and a Manipulator Tool Offset Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 19 lists the vocabulary of the Manipulator End Effector Pose Sensor Service.

TABLE 19 - MANIPULATOR END EFFECTOR POSE SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2615h	QueryEndEffectorPose	False
Output Set		
4615h	ReportEndEffectorPose	False

Protocol Behavior

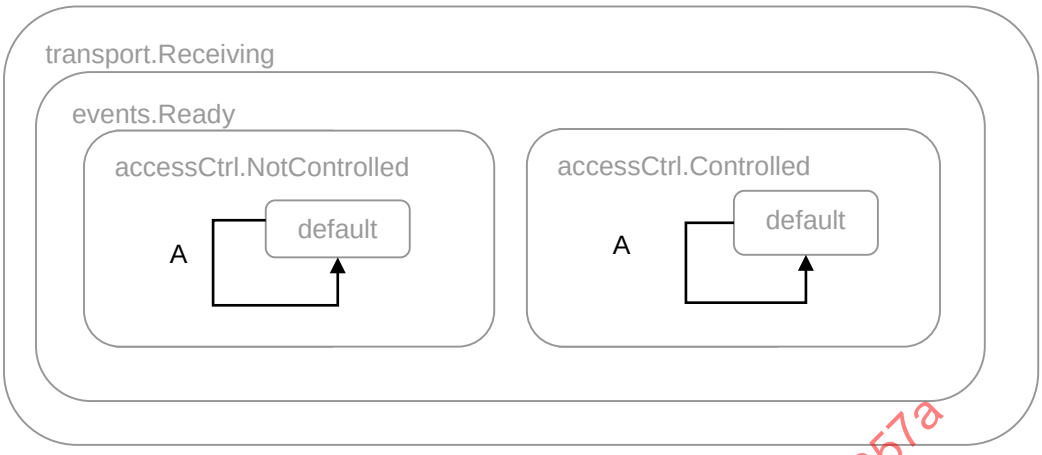


FIGURE 18 - MANIPULATOR END EFFECTOR POSE SENSOR SERVICE PROTOCOL BEHAVIOR

TABLE 20 - MANIPULATOR END EFFECTOR POSE SENSOR SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryEndEffectorPose		sendReportEndEffectorPose

TABLE 21 - MANIPULATOR END EFFECTOR POSE SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportEndEffectorPose	Send a Report End Effector Pose message.

4.8 Manipulator End Effector Velocity State Sensor Service

name= ManipulatorEndEffectorVelocityStateSensor
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorEndEffectorVelocityStateSensor

Inherits-from Access Control
  name=accessControl
  id= urn:jaus:jss:core:AccessControl
  version=1.1

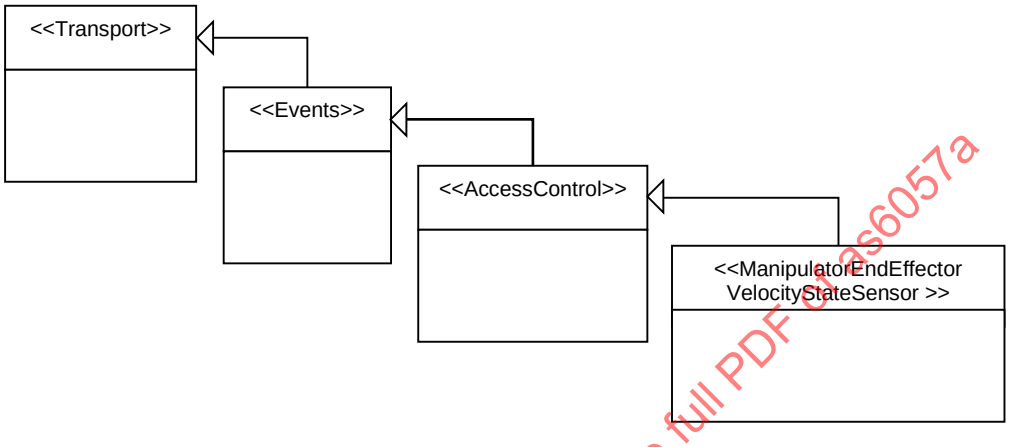


FIGURE 19 - MANIPULATOR END EFFECTOR VELOCITY STATE SENSOR SERVICE

Description

The function of the End Effector Velocity State Sensor is to report the velocity state of the tool tip as defined by two length-three vectors, i.e., ω_e and $v_{tool,e}$. These vectors respectively represent the angular velocity of the end effector coordinate system and the linear velocity of the tool tip as measured with respect to the manipulator base coordinate system. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service and a Manipulator Tool Offset Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 22 lists the vocabulary of the Manipulator End Effector Velocity State Sensor Service.

TABLE 22 - MANIPULATOR END EFFECTOR VELOCITY STATE SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2616h	QueryEndEffectorVelocityState	False
Output Set		
4616h	ReportEndEffectorVelocityState	False

Protocol Behavior

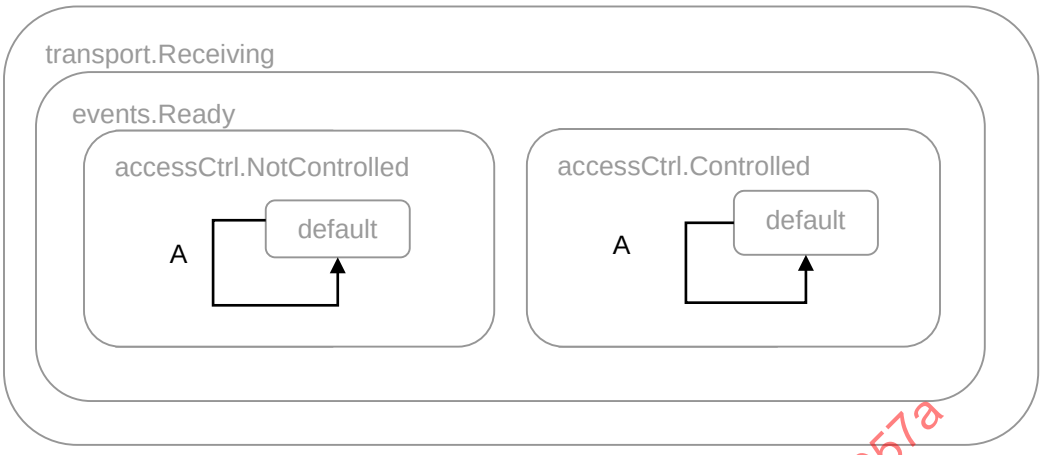


FIGURE 20 - MANIPULATOR END EFFECTOR VELOCITY STATE SENSOR SERVICE PROTOCOL BEHAVIOR

TABLE 23 - MANIPULATOR END EFFECTOR VELOCITY STATE SENSOR SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryEndEffector VelocityState		sendReportEnd EffectorVelocityState

TABLE 24 - MANIPULATOR END EFFECTOR VELOCITY STATE SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportEnd EffectorVelocityState	Send a Report End Effector Velocity State message.

4.9 Manipulator Joint Motion Profile Service

name= ManipulatorJointMotionProfile
version=2.0
id= urn:jaus:jss:manipulator: ManipulatorJointMotionProfile

Inherits-from AccessControl
  name=accessControl
  id= urn:jaus:jss:core:accessControl
  version=1.1

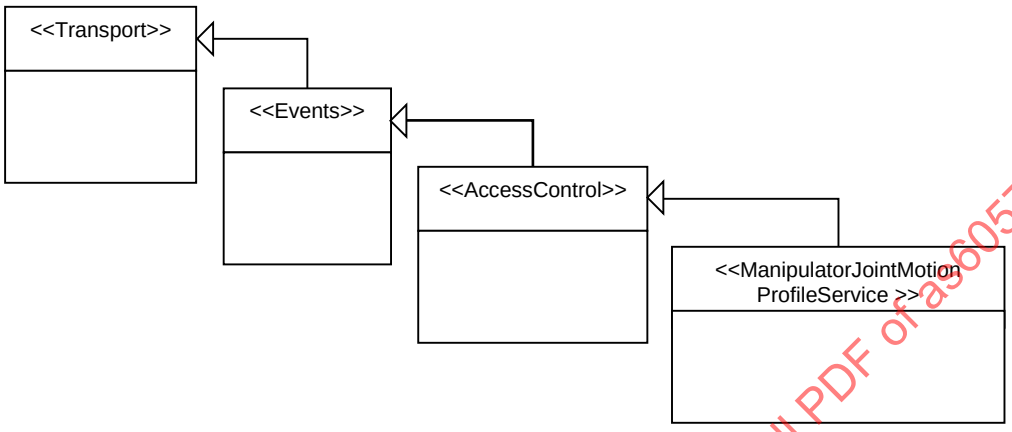


FIGURE 21 - MANIPULATOR JOINT MOTION PROFILE SERVICE

Description

The function of the Joint Motion Profile Service is to allow for configuration of the motion profile for all services co-located on this component. The Set Motion Profile message is used to set maximum velocity and acceleration rates for each of the joint parameters. All motions utilize the motion profile data that was most recently sent.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 25 lists the vocabulary of the Manipulator Joint Motion Profile Service.

TABLE 25 - MANIPULATOR JOINT MOTION PROFILE SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0607h	*SetJointMotionProfile*	True
2607h	QueryJointMotionProfile	False
Output Set		
4607h	ReportJointMotionProfile	False

Protocol Behavior

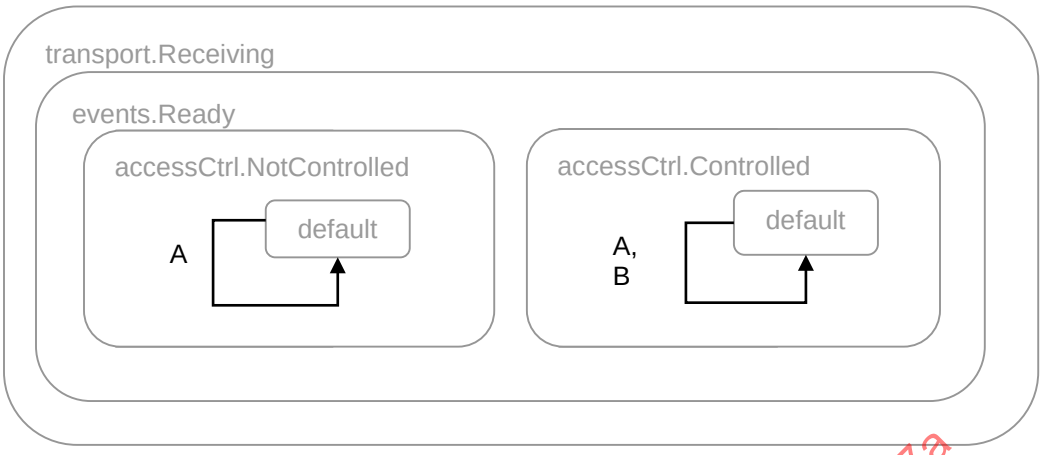


FIGURE 22 - MANIPULATOR JOINT MOTION PROFILE SERVICE PROTOCOL BEHAVIOR

TABLE 26 - MANIPULATOR JOINT MOTION PROFILE SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryJointMotionProfile	*motionProfileExists*	sendReportJointMotionProfile
B	*SetJointMotionProfile*	*management.accessCtrl.isControllingClient*	*setJointMotionProfile*

TABLE 27 - MANIPULATOR JOINT MOTION PROFILE SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
motionProfileExists	True if a motion profile is available.

TABLE 28 - MANIPULATOR JOINT MOTION PROFILE SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportJointMotionProfile	Send a ReportJointMotionProfile message.
setJointMotionProfile	Set the joint motion profile parameters for the manipulator.

4.10 Manipulator Joint Position Driver Service

name= ManipulatorJointPositionDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorJointPositionDriver

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

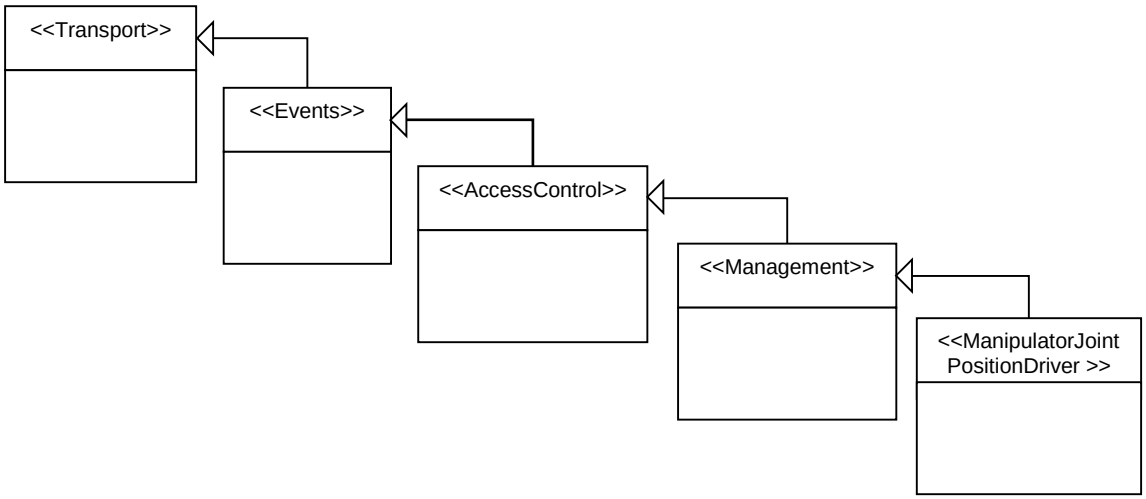


FIGURE 23 - MANIPULATOR JOINT POSITION DRIVER SERVICE

Description

The function of the Joint Position Driver is to perform closed-loop joint position control. A single target is provided via the Set Joint Position message. The target remains unchanged until a new Set Joint Position message is received. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service and a Manipulator Joint Motion Profile Service. Note that while the Joint Position Driver mandates upper and lower bounds on the joint range, this is not intended to limit motion on continuous-turn axes when commanded by effort, torque or velocity using one of the other Driver Services.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 29 lists the vocabulary of the Manipulator Joint Position Driver Service.

TABLE 29 - MANIPULATOR JOINT POSITION DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0602h	*SetJointPosition*	True
2608h	QueryCommandedJointPosition	False
Output Set		
4608h	ReportCommandedJointPosition	False

Protocol Behavior

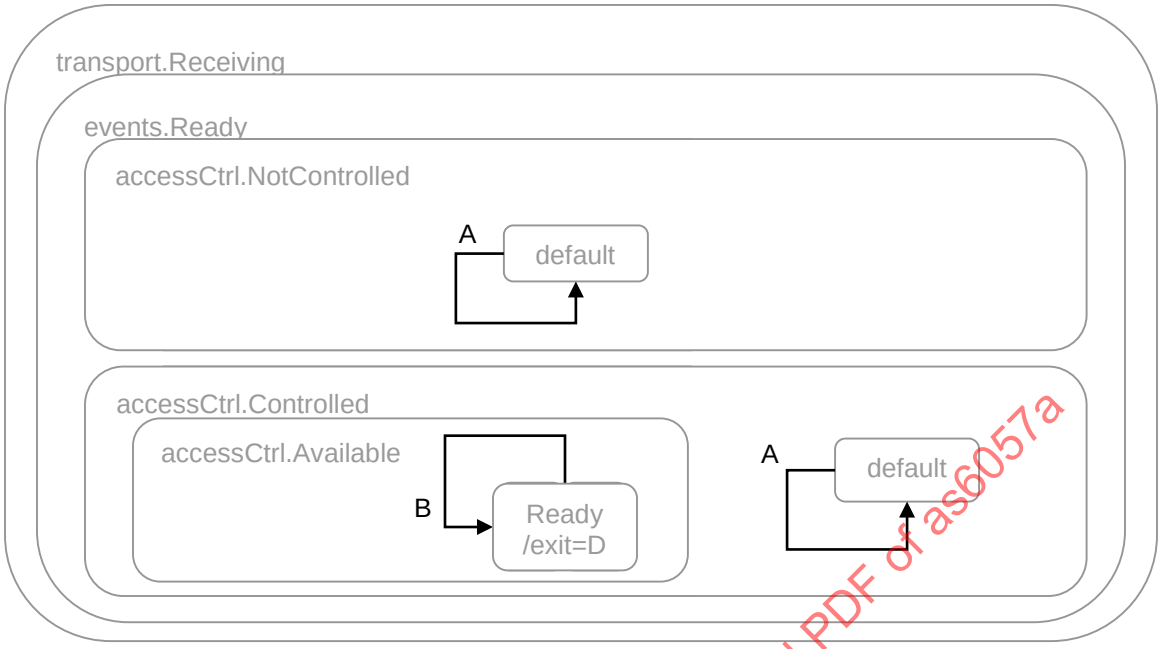


FIGURE 24 - MANIPULATOR JOINT POSITION DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 30 - MANIPULATOR JOINT POSITION DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 31 - MANIPULATOR JOINT POSITION DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommanded JointPosition		sendReportCommanded JointPosition
B	*SetJointPosition*	*management.accessCtrl. isControllingClient AND motionProfileExists*	*setJointPosition*

TABLE 32 - MANIPULATOR JOINT POSITION DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
motionProfileExists	True if a motion profile is available.

TABLE 33 - MANIPULATOR JOINT POSITION DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommanded JointPosition	Send a ReportCommandedJointPosition message.
setJointPosition	Set the desired joint values for the manipulator.
stopMotion	Stop motion of the manipulator.

4.11 Manipulator Joint Position List Driver Service

name= ManipulatorJointPositionListDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorJointPositionListDriver

Inherits-from ManipulatorListDriver
name= manipulatorListDriver
id= urn:jaus:jss:manipulator:ManipulatorListDriver
version=1.1

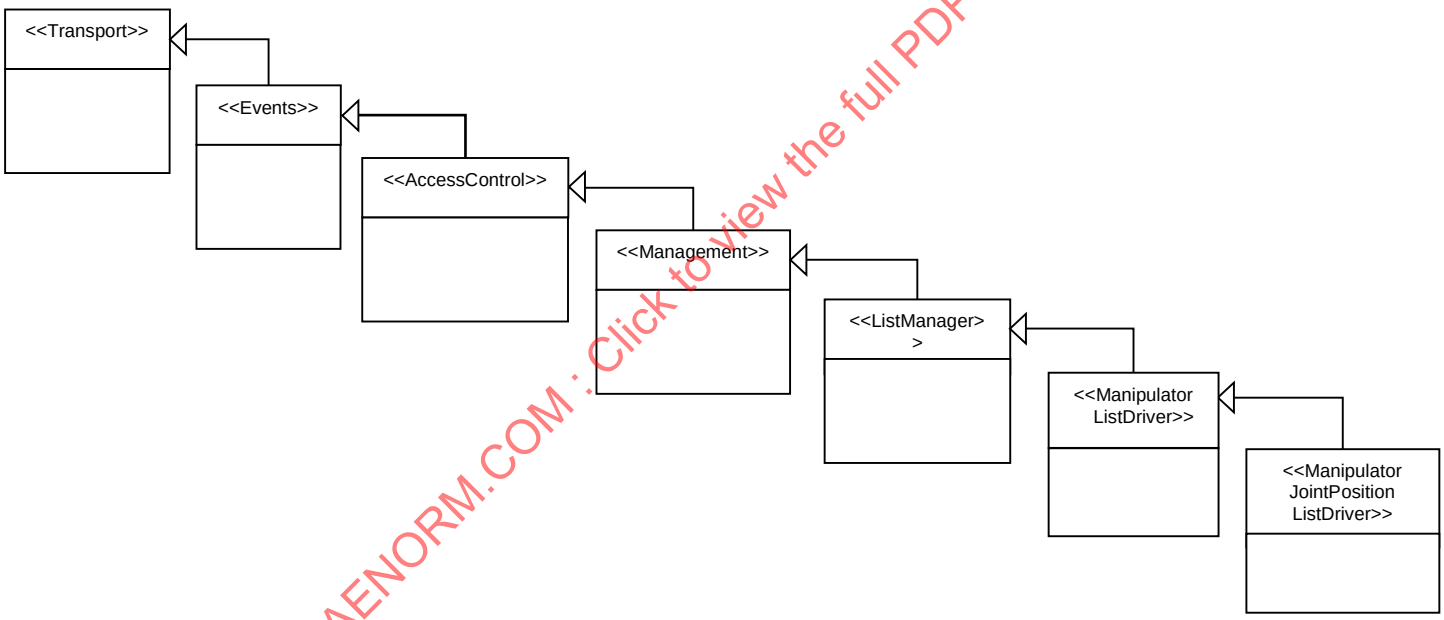


FIGURE 25 – MANIPULATOR JOINT POSITION LIST DRIVER SERVICE

Description

The function of the Joint Position List Driver is to perform closed-loop joint position control through a sequence of targets. The sequence of targets is specified by one or more SetElement messages, as defined by the List Manager Service. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service and a Manipulator Joint Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 34 lists the vocabulary of the Manipulator Joint Position List Driver Service.

TABLE 34 - MANIPULATOR JOINT POSITION LIST DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2608h	QueryCommandedJointPosition	False
Output Set		
4608h	ReportCommandedJointPosition	False

Protocol Behavior

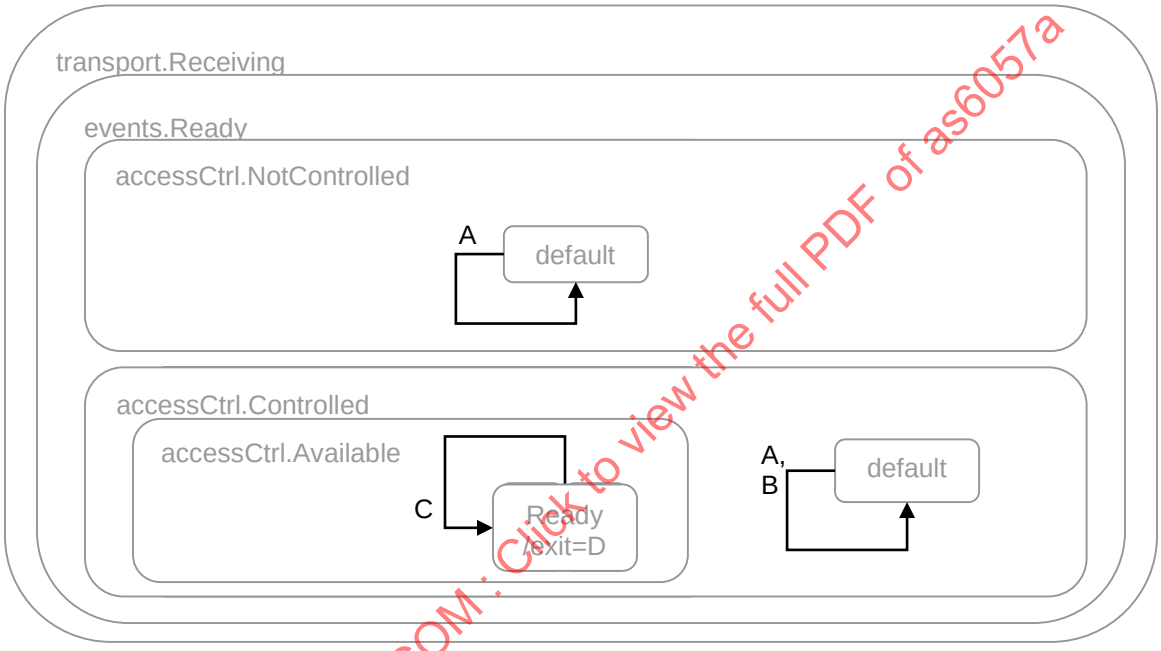


FIGURE 26 - MANIPULATOR JOINT POSITION LIST DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 35 - MANIPULATOR JOINT POSITION LIST DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 36 - MANIPULATOR JOINT POSITION LIST DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommanded JointPosition	*targetExists*	sendReportCommanded JointPosition
B	*SetElement*	*management.accessCtrl. isControllingClient AND isValidElementRequest AND isElementSupported*	*setElement* sendConfirmElementRequest
C	*ExecuteList*	*management.accessCtrl. isControllingClient* AND *elementExists AND isListValid*	*executeTargetList*

TABLE 37 - MANIPULATOR JOINT POSITION LIST DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
elementExists	True if the Element UID specified in the message that triggered the transition exists in the list.
isValidElementRequest	True if the resulting list will not be invalid as defined by the List Manager Service description and the receiving entity has sufficient memory to store the element(s).
isElementSupported	True if the message that triggered the transition contains payload(s) of valid serialized Set Joint Position message(s).
targetExists	True if a valid target position has been received.
isListValid	True if the list contains Manipulator Joint Position messages.

TABLE 38 - MANIPULATOR JOINT POSITION LIST DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
setElement	Store the given targets(s) in the target list with sequence specified by the previous and next element UIDs. If this action represents an insert or append into an existing list, the service should modify the NextUID of the previous element and/or the PreviousUID of the next element to reflect the updated sequence
executeTargetList	Begin sequential execution of the target list starting at the specified element.
sendReportCommanded JointPosition	Send a ReportCommandedJointPosition message.
sendConfirmElementRequest	Send a ConfirmElementRequest message.
stopMotion	Stop motion of the manipulator.

4.12 Manipulator End Effector Pose Driver Service

name= ManipulatorEndEffectorPoseDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorEndEffectorPoseDriver

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

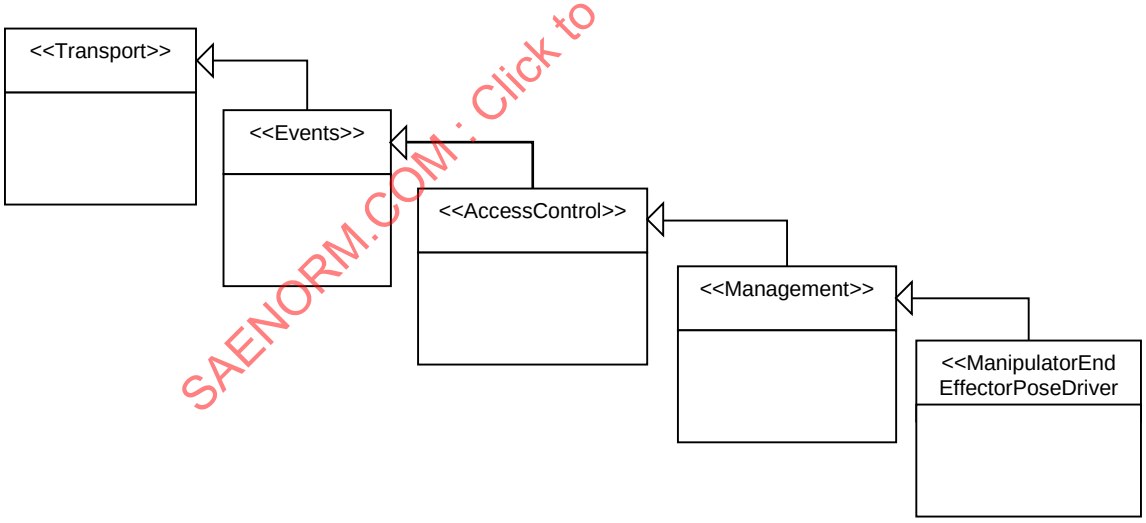


FIGURE 27 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE

Description

The function of the End Effector Pose Driver is to perform closed-loop position and orientation control of the tool tip. The input is the desired position and orientation of the end effector pose specified in the manipulator base coordinate system. It is assumed that the manipulator begins motion immediately after receiving the “Set End Effector Pose” message. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service, a Manipulator Tool Offset Service, and a Manipulator Joint Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 39 lists the vocabulary of the Manipulator End Effector Pose Driver Service.

TABLE 39 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0610h	*SetEndEffectorPose*	True
2610h	QueryCommandedEndEffectorPose	False
Output Set		
4610h	ReportCommandedEndEffectorPose	False

Protocol Behavior

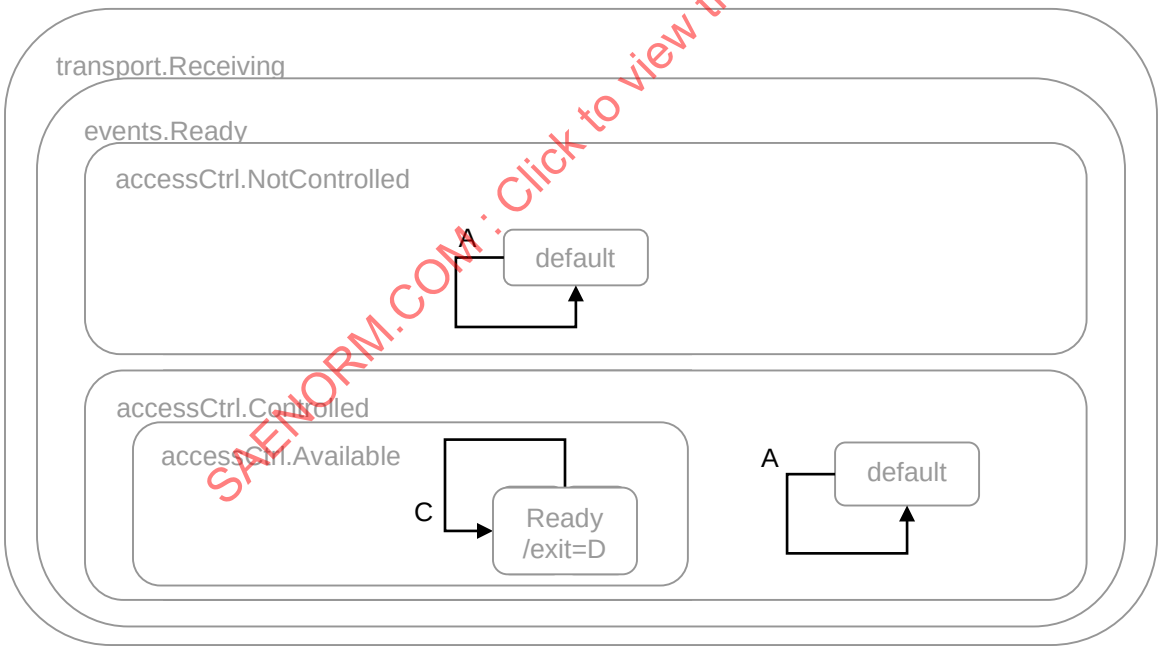


FIGURE 28 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 40 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 41 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommandedEnd EffectorPose		sendReportCommanded EndEffectorPose
C	*SetEndEffectorPose*	*management.accessCtrl. isControllingClient AND motionProfileExists*	*setEndEffectorPose*

TABLE 42 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
motionProfileExists	True if a motion profile is available.

TABLE 43 - MANIPULATOR END EFFECTOR POSE DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommanded EndEffectorPose	Send a ReportCommandedEndEffectorPose message.
setEndEffectorPose	Set the desired position and orientation for the manipulator end effector.
stopMotion	Stop motion of the manipulator.

4.13 Manipulator List Driver Service

name=ManipulatorListDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorListDriver

Inherits-from ListManager
  name= listManager
  id= urn:jaus:jss:core:ListManager
  version=1.1

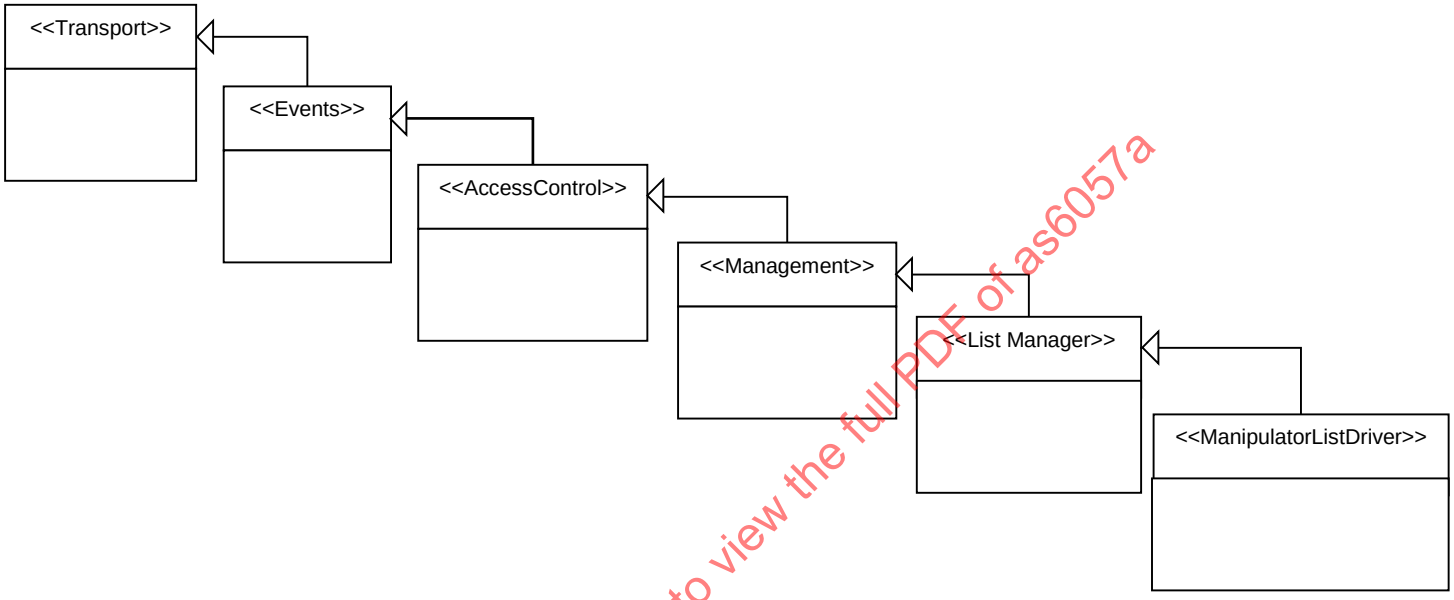


FIGURE 29 - MANIPULATOR LIST DRIVER SERVICE

Description

The function of the Manipulator List Driver is to add support for executing a list of manipulator positions. It is expected that child services will inherit this service to provide functionality by overriding the *isListValid()* guard in the protocol.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 44 lists the vocabulary of the Manipulator List Driver Service.

TABLE 44 - MANIPULATOR LIST DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
061Eh	*ExecuteList*	True
261Eh	QueryActiveElement	False
Output Set		
461Eh	ReportActiveElement	False

Protocol Behavior

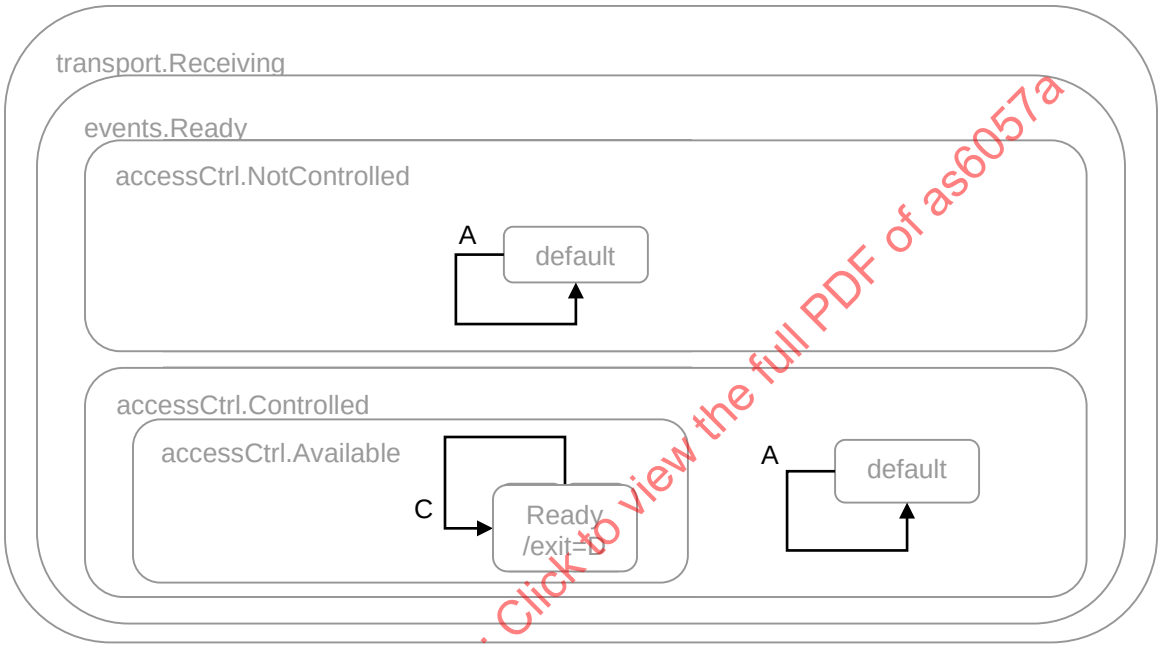


FIGURE 30 - MANIPULATOR LIST DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 45 - MANIPULATOR LIST DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	**State**	**Type**	**Guard**	**Actions**
D	management. Ready	exit		*stopMotion*

TABLE 46 - MANIPULATOR LIST DRIVER SERVICE TRANSITION TABLE

Label	**Trigger**	**Guard**	**Actions**
A	QueryActiveElement		sendReportActiveElement
C	*ExecuteList*	*management.accessCtrl. isControllingClient* AND *elementExists AND isListValid*	*executeTargetList*

TABLE 47 - MANIPULATOR LIST DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
elementExists	True if the Element UID specified in the message that triggered the transition exists in the list.
isListValid	This guard is always FALSE. It must be overridden by inheriting services to provide functionality.

TABLE 48 - MANIPULATOR LIST DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
executeTargetList	Begin sequential execution of the target list starting at the specified element.
sendReportActiveElement	Send a ReportActiveElement message.
stopMotion	Stop motion of the manipulator.

4.14 Manipulator End Effector Pose List Driver Service

name= ManipulatorEndEffectorPoseListDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorEndEffectorPoseListDriver

Inherits-from ManipulatorListDriver
name= manipulatorListDriver
id= urn:jaus:jss:manipulator: ManipulatorListDriver
version=1.1

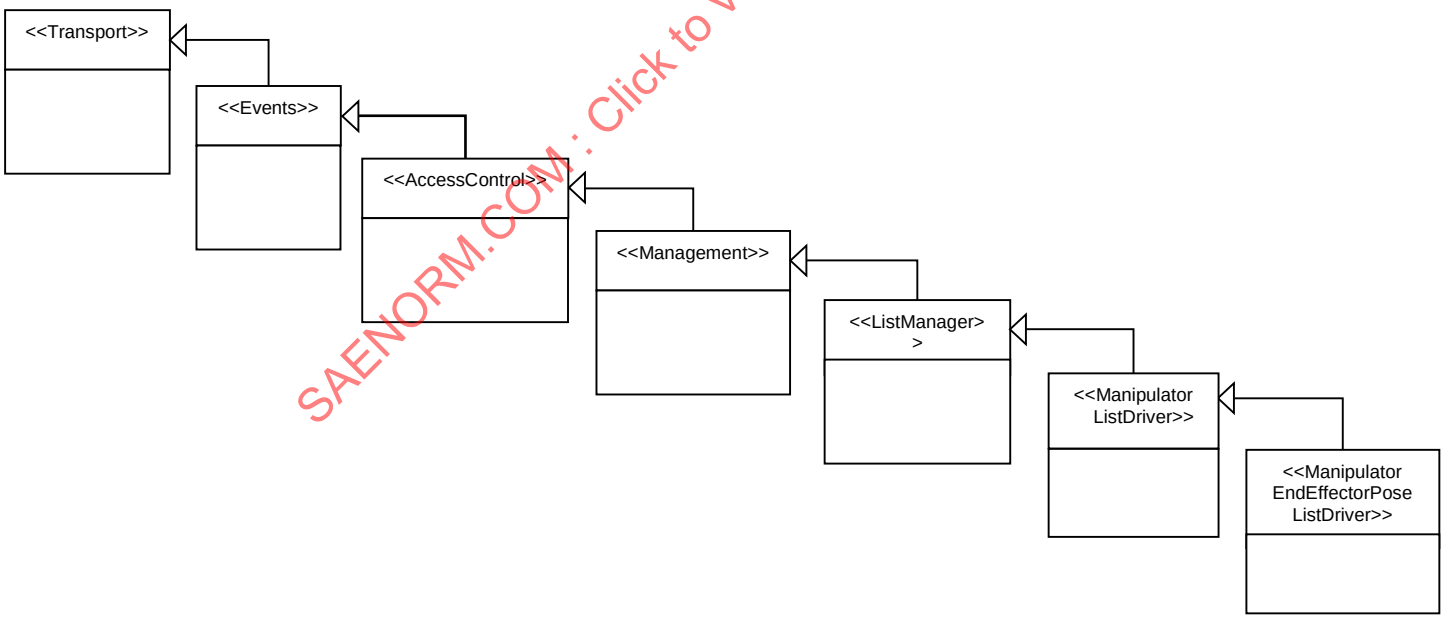


FIGURE 31 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE

Description

The function of the End Effector Pose List Driver is to perform closed-loop control of a sequence of positions and orientations of the tool tip specified in the manipulator base coordinate system. The sequence of targets is specified by one or more SetElement messages, as defined by the List Manager Service [AS6009]. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service, a Manipulator Tool Offset Service, and a Manipulator Joint Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 49 lists the vocabulary of the Manipulator End Effector Pose List Driver Service.

TABLE 49 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2610h	QueryCommandedEndEffectorPose	False
Output Set		
4610h	ReportCommandedEndEffectorPose	False

Protocol Behavior

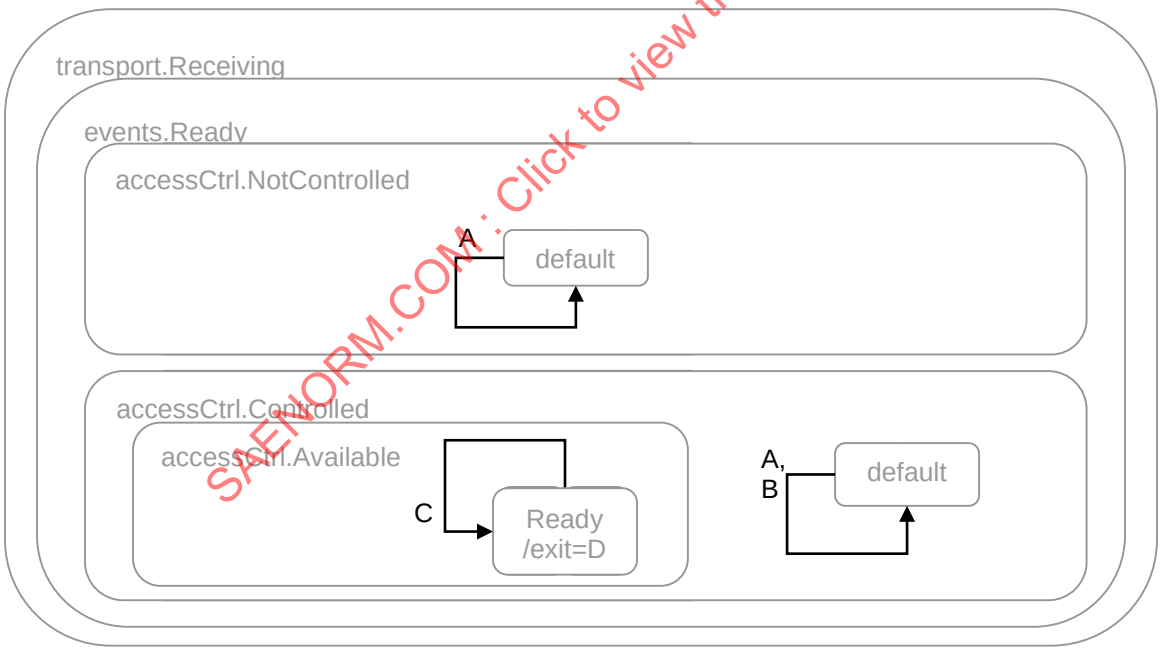


FIGURE 32 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 50 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 51 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommandedEnd EffectorPose		sendReportCommanded EndEffectorPose
B	*SetElement*	*management.accessCtrl. isControllingClient AND isValidElementRequest AND isElementSupported*	*setElement* sendConfirmElementRequest
C	*ExecuteList*	*management.accessCtrl. isControllingClient* AND *elementExists AND isListValid*	*executeTargetList*

TABLE 52 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
elementExists	True if the Element UID specified in the message that triggered the transition exists in the list.
isValidElementRequest	True if the resulting list will not be invalid as defined by the List Manager Service description and the receiving entity has sufficient memory to store the element(s).
isElementSupported	True if the message that triggered the transition contains payload(s) of valid serialized Set End Effector Pose message(s).
isListValid	True if the list contains Set End Effector Pose messages.

TABLE 53 - MANIPULATOR END EFFECTOR POSE LIST DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
setElement	Store the given targets(s) in the target list with sequence specified by the previous and next element UIDs. If this action represents an insert or append into an existing list, the service should modify the NextUID of the previous element and/or the PreviousUID of the next element to reflect the updated sequence
executeTargetList	Begin sequential execution of the target list starting at the specified element.
sendReportCommanded EndEffectorPose	Send a ReportCommandedEndEffectorPose message.
sendConfirmElementRequest	Send a ConfirmElementRequest message.
stopMotion	Stop motion of the manipulator.

4.15 Manipulator Joint Velocity Driver Service

name= ManipulatorJointVelocityDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorJointVelocityDriver

Inherits-from Management
name=management
id= urn:jaus:jss:core:Management
version=1.1

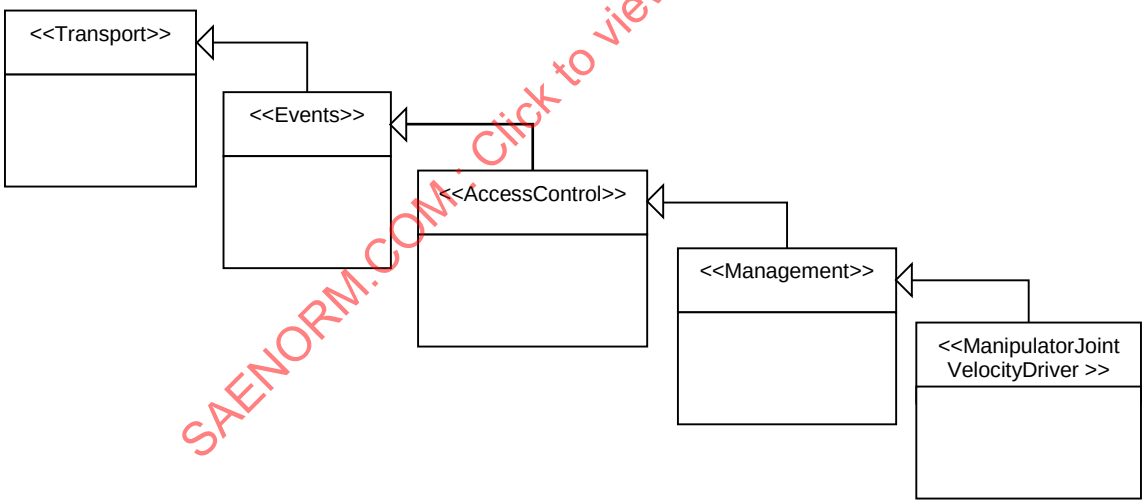


FIGURE 33 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE

Description

The function of the Joint Velocity Driver is to perform closed-loop joint velocity control. The input is the desired instantaneous joint velocities. It is assumed that the manipulator begins motion immediately after receiving the “SET JOINT VELOCITY” message. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service and a Manipulator Joint Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 54 lists the vocabulary of the Manipulator Joint Velocity Driver Service.

TABLE 54 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0603h	*SetJointVelocity*	True
2611h	QueryCommandedJointVelocity	False
Output Set		
4611h	ReportCommandedJointVelocity	False

Protocol Behavior

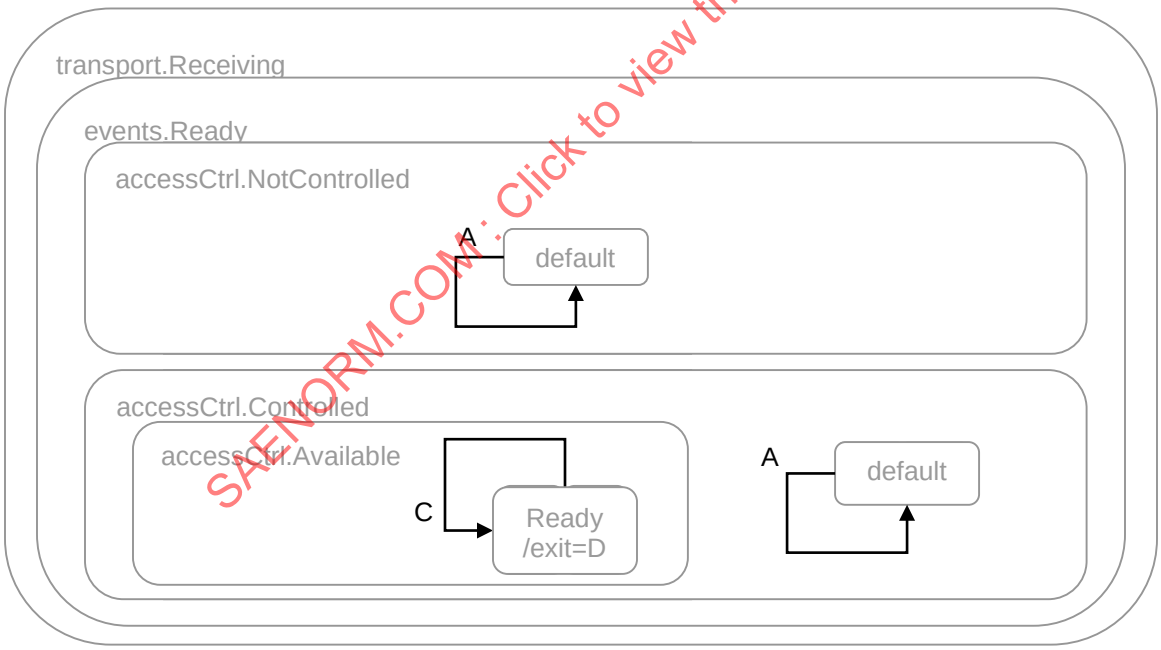


FIGURE 34 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 55 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 56 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommanded JointVelocity		sendReportCommandedJointVelocity
C	*SetJointVelocity*	*management.accessCtrl. isControllingClient AND motionProfileExists*	*setJointVelocity*

TABLE 57 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE STATE TRANSITION TABLE

Condition	Interpretation
motionProfileExists	True if a motion profile is available.

TABLE 58 - MANIPULATOR JOINT VELOCITY DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommandedJoint Velocity	Send a ReportCommandedJointVelocity message
setJointVelocity	Set the desired velocities for the individual joints of the manipulator.
stopMotion	Stop motion of the manipulator.

4.16 Manipulator End Effector Velocity State Driver Service

name= ManipulatorEndEffectorVelocityStateDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorEndEffectorVelocityStateDriver

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

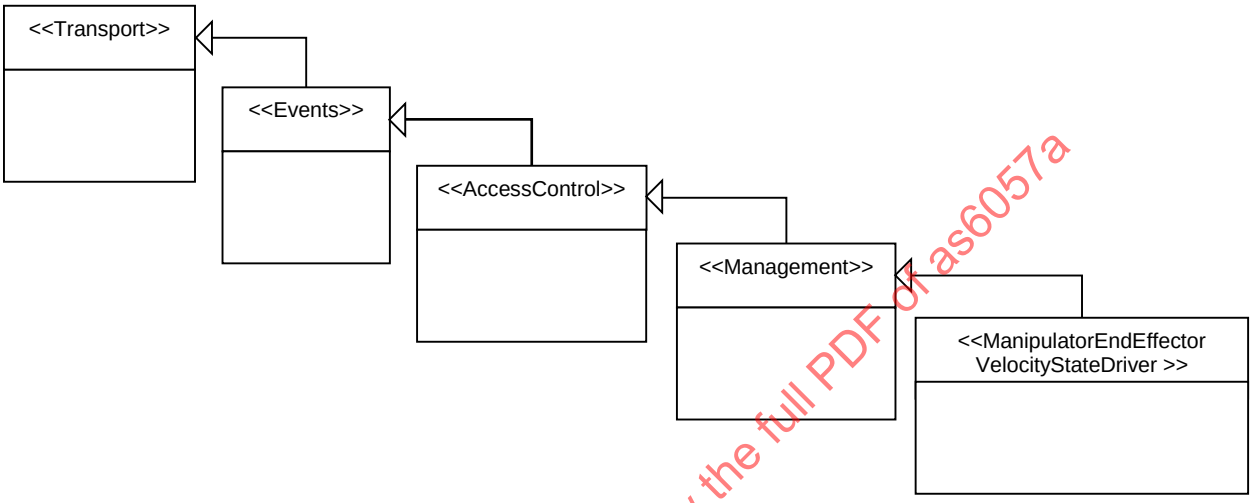


FIGURE 35 - MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER SERVICE

Description

The function of the End Effector Velocity State Driver is to perform closed-loop velocity control of the tool tip. The velocity state of the tool tip is defined by two length-three vectors, i.e., $\boldsymbol{\omega}_e$ and $\mathbf{v}_{tool,e}$. These vectors respectively represent the angular velocity of the end effector coordinate system and the linear velocity of the tool tip as measured with respect to the manipulator base coordinate system. It is assumed that the manipulator begins motion immediately after receiving the "Set End Effector Velocity State" message. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service, a Manipulator Tool Offset Service, and a Manipulator Joint Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 59 lists the vocabulary of the Manipulator End Effector Velocity State Driver Service.

TABLE 59 - MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0612h	*SetEndEffectorVelocityState*	True
2612h	QueryCommandedEndEffectorVelocityState	False
Output Set		
4612h	ReportCommandedEndEffectorVelocityState	False

Protocol Behavior

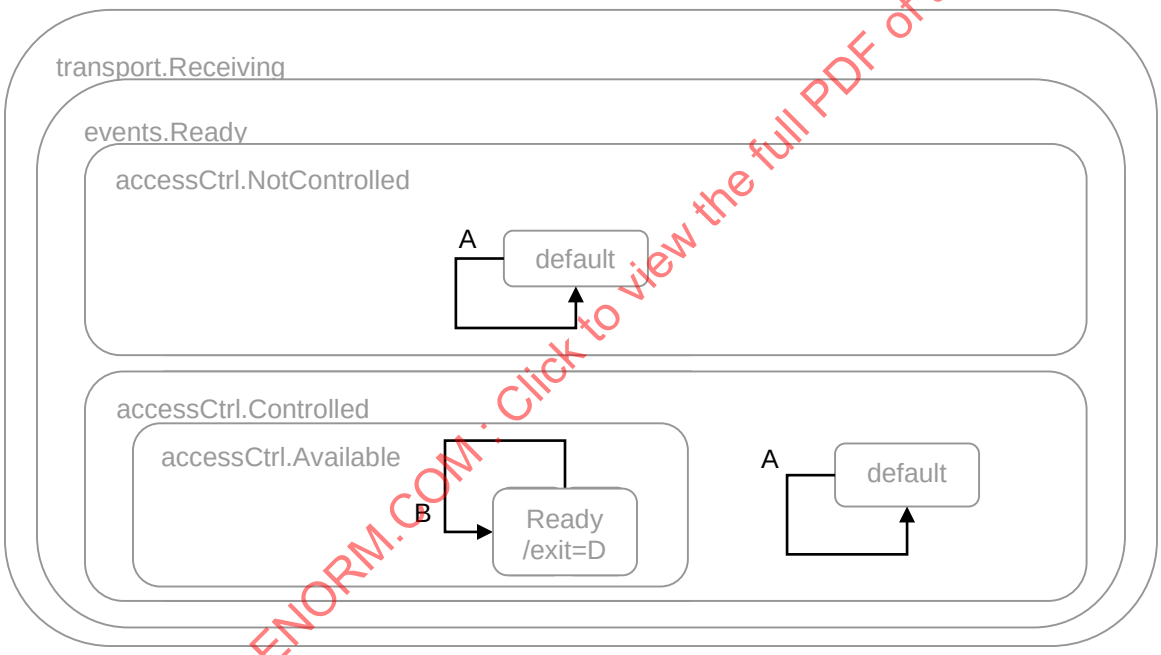


FIGURE 36 - MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 60 - MANIPULATOR END EFFECTOR VELOCITY DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 61 - MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommandedEnd EffectorVelocityState		sendReportCommandedEnd EffectorVelocityState
B	*SetEndEffectorVelocity State*	*management.accessCtrl. isControllingClient AND motionProfileExists*	*setEndEffectorVelocityState*

TABLE 62 - MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
motionProfileExists	True if a motion profile is available.

TABLE 63 - MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommandedEnd EffectorVelocityState	Send a ReportCommandedEndEffector Velocity State message.
setEndEffectorVelocityState	Set the desired velocity state for the end effector.
stopMotion	Stop motion of the manipulator.

4.17 Manipulator Actuator Force/Torque Driver Service

name= ManipulatorActuatorForceTorqueDriver
version=2.0
id= urn:jaus:jss:manipulator:ManipulatorActuatorForceTorqueDriver

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

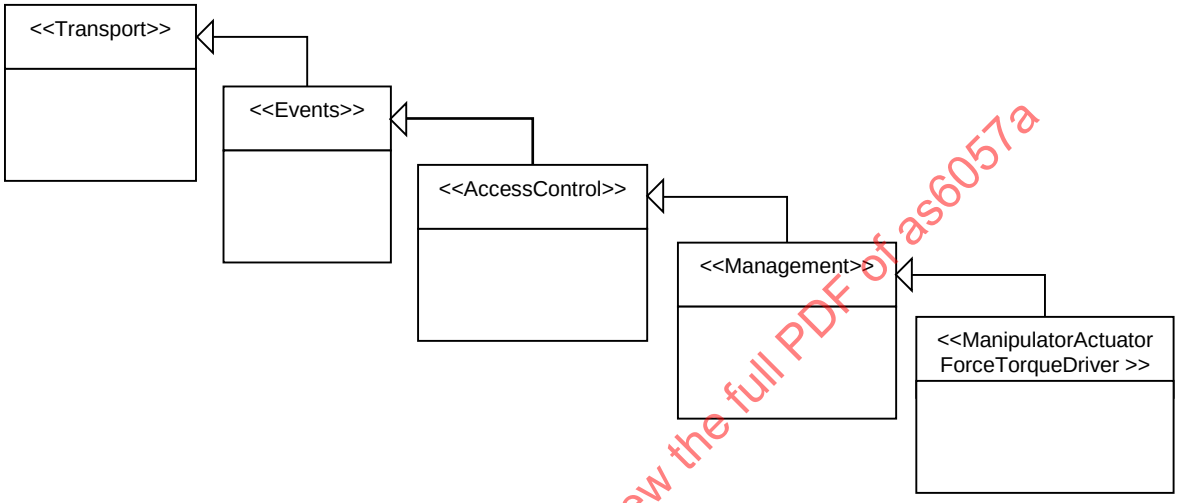


FIGURE 37 - MANIPULATOR ACTUATOR FORCE/TORQUE DRIVER SERVICE

Description

The function of the Actuator Force/Torque Driver is to perform closed-loop force control (for a prismatic actuator) and closed-loop torque control (for a revolute actuator). If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Manipulator Specification Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 64 lists the vocabulary of the Manipulator End Effector Velocity State Driver Service.

TABLE 64 - MANIPULATOR ACTUATOR FORCE TORQUE DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0613h	*SetActuatorForceTorques*	True
2613h	QueryCommandedActuatorForceTorque	False
Output Set		
4613h	ReportCommandedActuatorForceTorque	False

Protocol Behavior

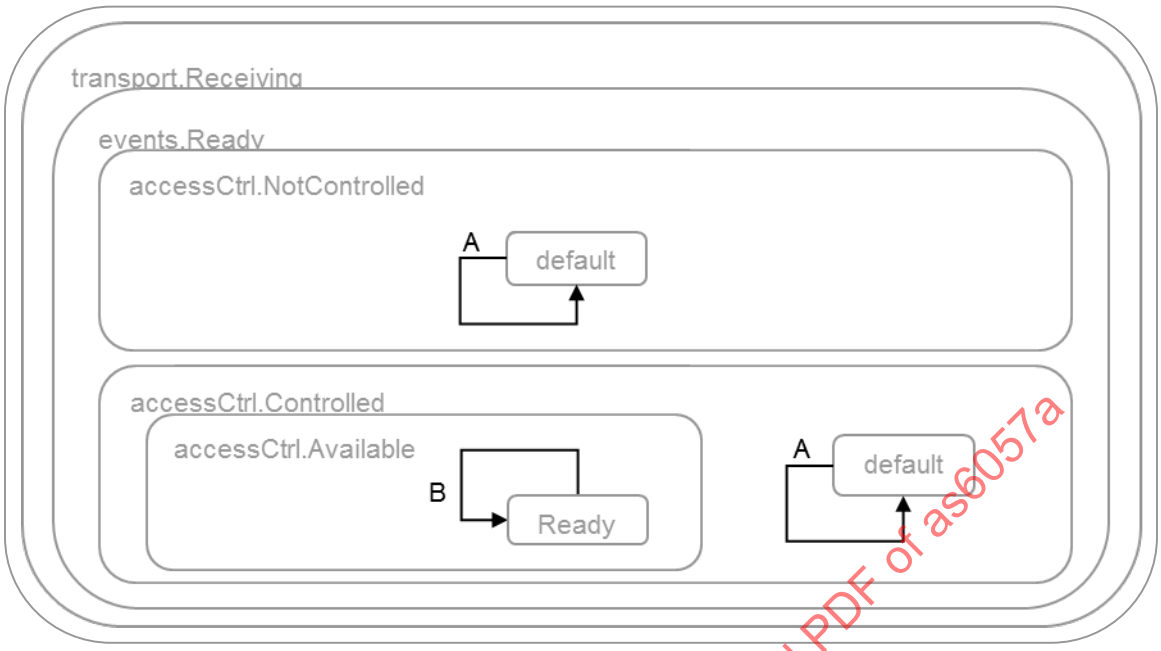


FIGURE 38 - MANIPULATOR ACTUATOR FORCE/TORQUE DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 65 - MANIPULATOR ACTUATOR FORCE TORQUE DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommanded CommandedActuator ForceTorque		sendReportCommanded ActuatorForceTorque
B	*SetActuatorForce Torque*	*management.accessCtrl. isControllingClient*	*setActuatorForceTorques*

TABLE 66 - MANIPULATOR ACTUATOR FORCE TORQUE DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommanded ActuatorForceTorque	Send a Report Commanded Actuator Force Torque message.
setActuatorForceTorques	Set the desired actuator forces and torques.

4.18 Primitive Pan Tilt Service

Name= PrimitivePanTilt
version=2.0
id= urn:jaus:jss:manipulator: PrimitivePanTilt

Inherits-from Management
name=management
id= urn:jaus:jss:core:Management
version=1.1

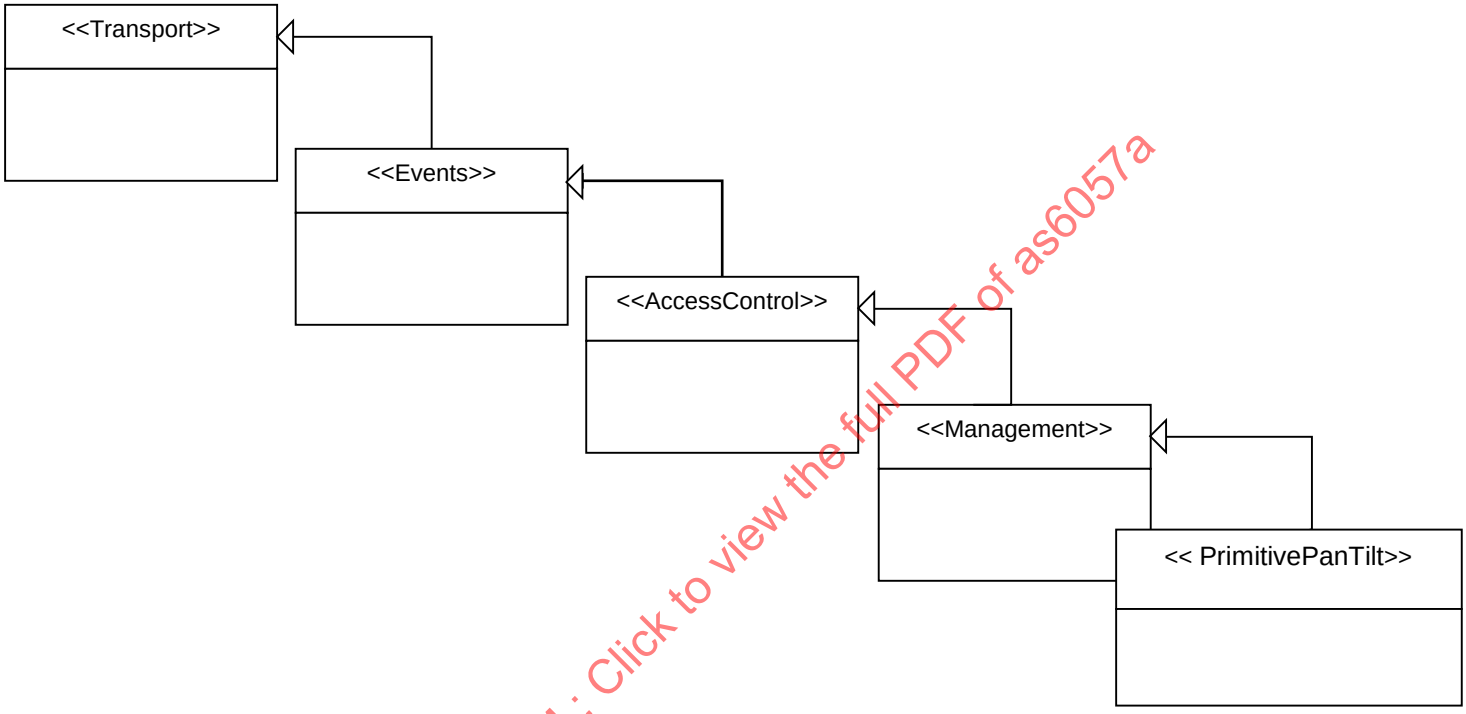


FIGURE 39 - PRIMITIVE PAN TILT SERVICE

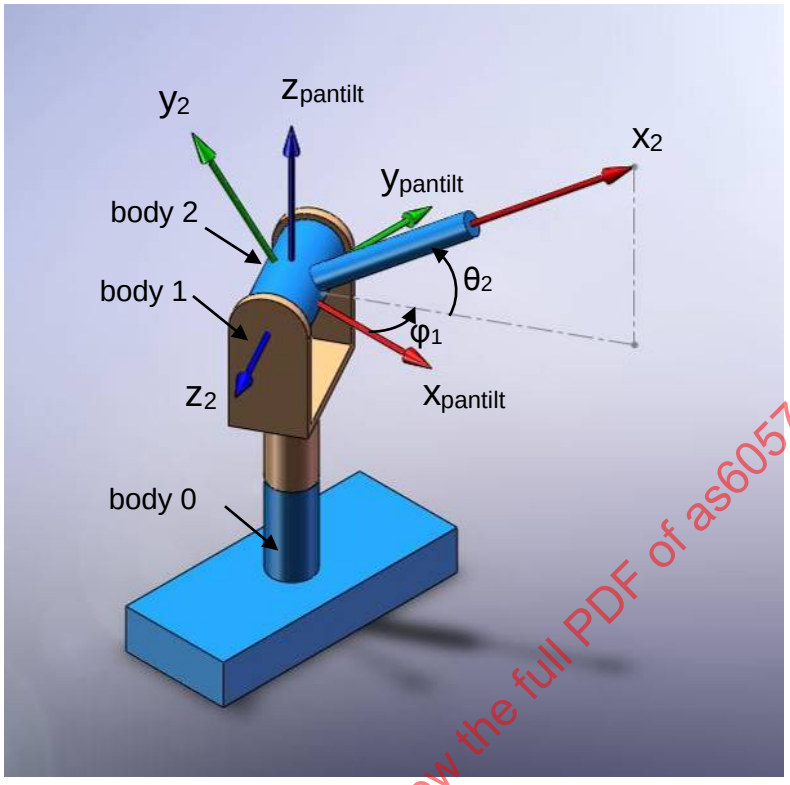


FIGURE 40 - PAN TILT MECHANISM

Description

A pan tilt mechanism is a two degree of freedom manipulator that has special geometry. The device could be controlled using the standard manipulator services. However because of how often this mechansim is used, specific services have been developed.

Figure 32 shows the device together with two coordinate systems. The 'pantilt' coordinate system is fixed to the base, i.e., body 0. The origin of this coordinate system is located at the intersection point of the two axes of rotation. The zpantilt axis is along the first axis of rotation (joint 1). The x2y2z2 coordinate system is fixed to body 2. Its origin is coincident with the origin of the 'pantilt' coordinate system and its z2 axis is along the second axis of rotation (joint 2). The constant mechanism parameters for this device are a12 = 0, α12 = 90°, and S2 = 0. The figure also shows the angles φ1 and θ2 which are the two rotational freedoms for the device. The 4×4 transformation matrix that relates coordinate system 2 to the pantilt coordinate system can be written in terms of these two angles as

$$ {}^{pantilt}_{2}\mathbf{T} = \begin{bmatrix} \cos(\phi_1)\cos(\theta_2) & -\cos(\phi_1)\sin(\theta_2) & \sin(\phi_1) & 0 \\ \sin(\phi_1)\cos(\theta_2) & -\sin(\phi_1)\sin(\theta_2) & -\cos(\phi_1) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

The Primitive Pan Tilt Service is the low level interface to a pan tilt mechanism. Motion of the pan tilt mechanism is accomplished via the Set Pan Tilt Joint Effort message. In this message, each actuator is commanded to move with a percentage of maximum effort. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Pan Tilt Specification Service.

Assumptions

Messages may be delayed, lost, or reordered.

Vocabulary

Table 67 lists the vocabulary of the Primitive Pan Tilt Service.

TABLE 67 - PRIMITIVE PAN TILT SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0621h	*SetPanTiltJointEffort*	True
2621h	QueryPanTiltJointEffort	False
Output Set		
4621h	ReportPanTiltJointEffort	False

Protocol Behavior

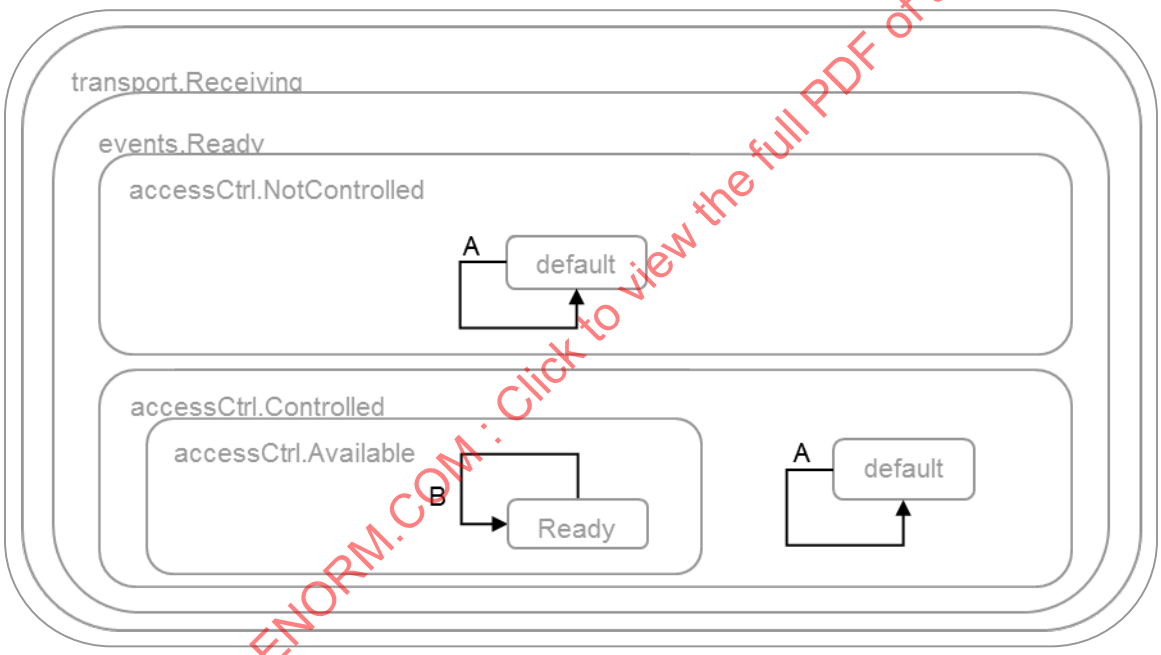


FIGURE 41 - PRIMITIVE PAN TILT SERVICE PROTOCOL BEHAVIOR

TABLE 68 - PRIMITIVE PAN TILT SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryPanTiltJoint Effort		sendReportPanTiltJoint Effort
B	SetPanTiltJointEffort	*management.accessCtrl. isControllingClient*	*setPanTiltJointEffort*

TABLE 69 - PRIMITIVE PAN TILT SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportPanTiltJoint Effort	Send a Report Pan Tilt Joint Effort message.
setPanTiltJointEffort	Set the joint motion efforts for the two joints of the pan tilt mechanism.

4.19 Pan Tilt Joint Position Sensor Service

name= PanTiltJointPositionSensor
version=2.0
id= urn:jaus:jss: manipulator: PanTiltJointPositionSensor

Inherits-from Events
  name=events
  id= urn:jaus:jss:core: Events
  version=1.1

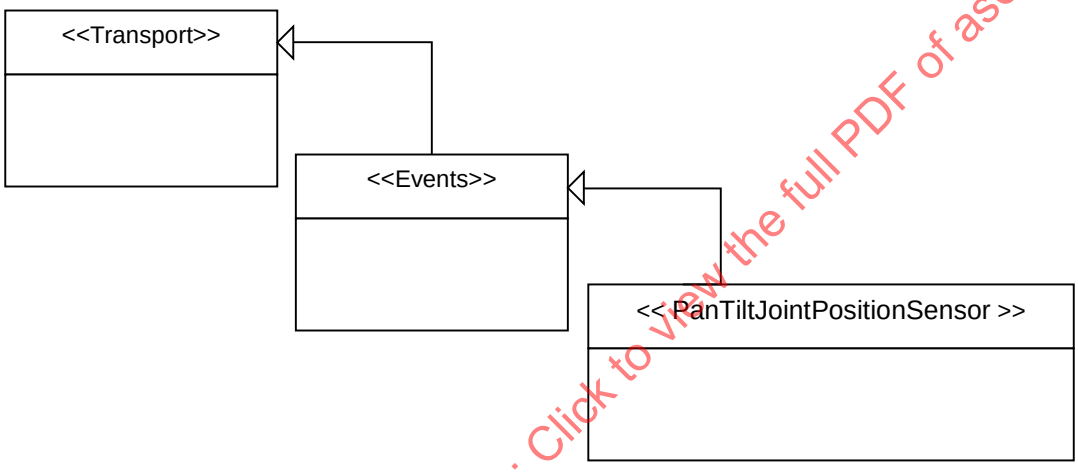


FIGURE 42 - PAN TILT JOINT POSITION SENSOR SERVICE

Description

The function of the Pan Tilt Joint Position Sensor Service is to report the values of the two joint angles of the pan tilt mechanism when queried. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Pan Tilt Specification Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 70 lists the vocabulary of the Pan Tilt Joint Position Sensor Service.

TABLE 70 - PAN TILT JOINT POSITION SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2622h	QueryPanTiltJointPositions	False
Output Set		
4622h	ReportPanTiltJointPosition	False

Protocol Behavior

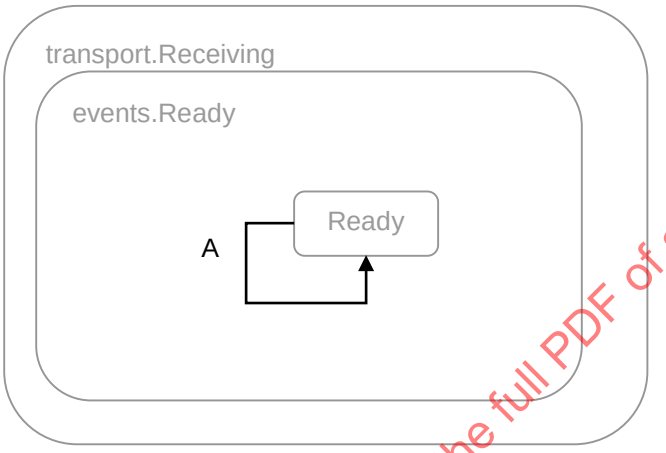


FIGURE 43 - PAN TILT JOINT POSITION SENSOR PROTOCOL BEHAVIOR

TABLE 71 - PAN TILT JOINT POSITION SENSOR SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryPanTiltJoint Position		sendReportPanTiltJointPosition

TABLE 72 - PAN TILT JOINT POSITION SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportPanTiltJointPosition	Send Report Pan Tilt Joint Position message to the service that sent the query

4.20 Pan Tilt Joint Velocity Sensor Service

name= PanTiltJointVelocitySensor
version=2.0
id= urn:jaus:jss: manipulator: PanTiltJointVelocitySensor

Inherits-from Events
  name=events
  id= urn:jaus:jss:core: Events
  version=1.1

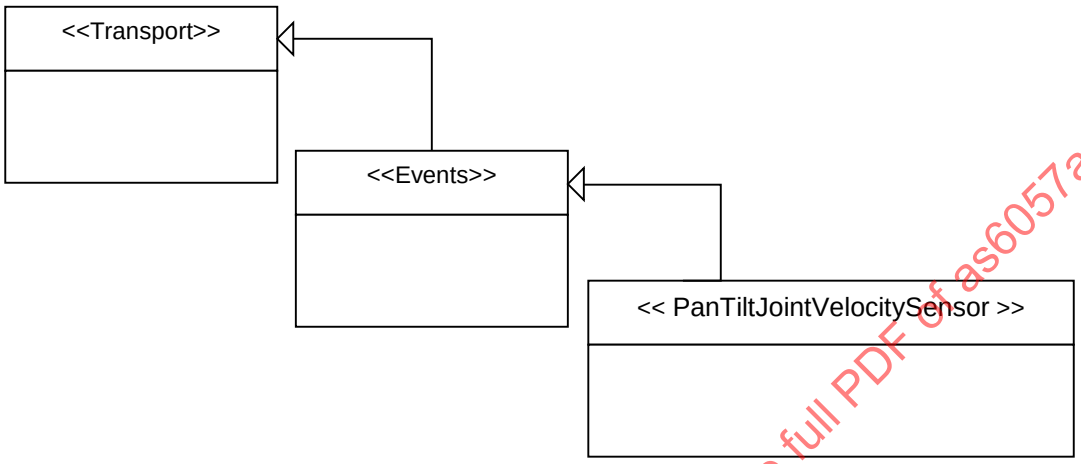


FIGURE 44 - PAN TILT JOINT VELOCITY SENSOR SERVICE

Description

The function of the Pan Tilt Joint Velocity Sensor Service is to report the values of the two joint velocities of the pan tilt mechanism when queried. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Pan Tilt Specification Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 73 lists the vocabulary of the Manipulator Joint Velocity Sensor Service.

TABLE 73 - PAN TILT JOINT VELOCITY SENSOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2623h	QueryPanTiltJointVelocity	False
Output Set		
4623h	ReportPanTiltJointVelocity	False

Protocol Behavior

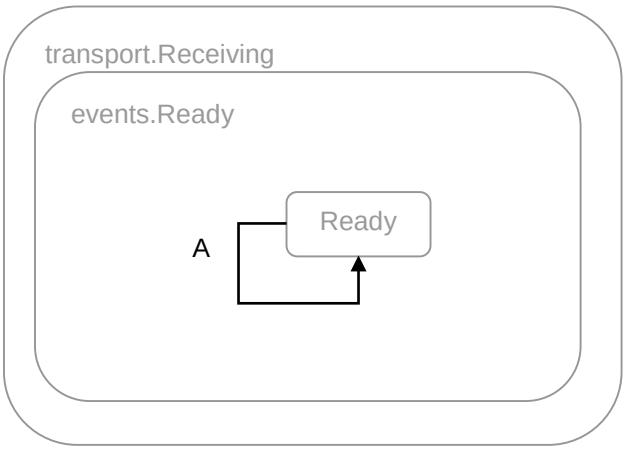


FIGURE 45 - PAN TILT JOINT VELOCITY SENSOR PROTOCOL BEHAVIOR

TABLE 74 - PAN TILT JOINT VELOCITY SENSOR SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryPanTiltJoint Velocity		sendReportPanTiltJointVelocity

TABLE 75 - PAN TILT JOINT VELOCITY SENSOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportPanTiltJointVelocity	Send Report Pan Tilt Joint Velocity message to the service that sent the query.

4.21 Pan Tilt Joint Position Driver Service

name= PanTiltJointPositionDriver
version=2.0
id= urn:jaus:jss:manipulator:PanTiltJointPositionDriver

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

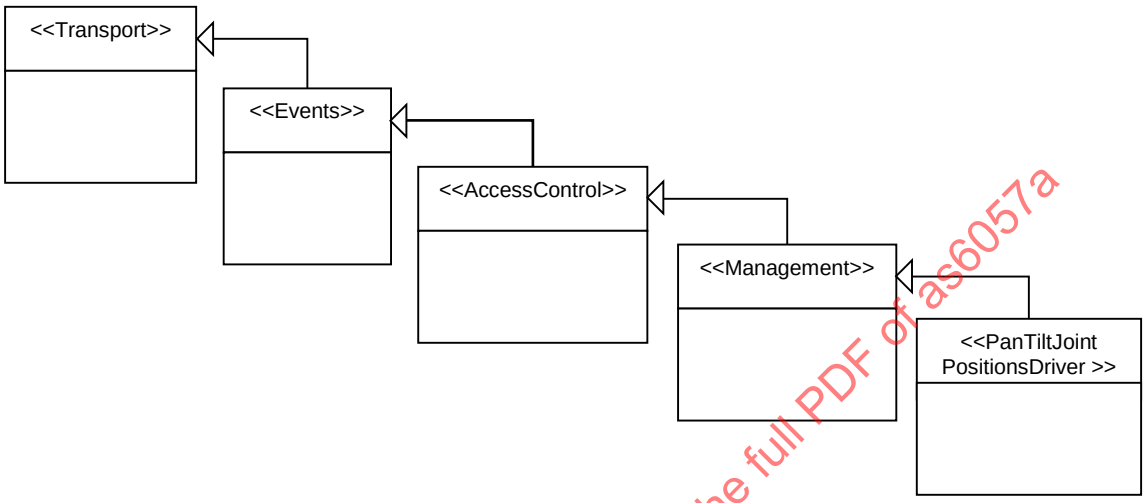


FIGURE 46 - PAN TILT JOINT POSITION DRIVER SERVICE

Description

The function of the Pan Tilt Joint Position Driver is to perform closed-loop joint position control. A single target is provided via the Set Pan Tilt Joint Position message. The target remains unchanged until a new Set Pan Tilt Joint Position message is received. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Pan Tilt Specification Service and a Pan Tilt Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 76 lists the vocabulary of the Pan Tilt Joint Position Driver Service.

TABLE 76 - PAN TILT JOINT POSITION DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0622h	*SetPanTiltJointPosition*	True
2628h	QueryCommandedPanTiltJointPositions	False
Output Set		
4628h	ReportCommandedPanTiltJointPositions	False

Protocol Behavior

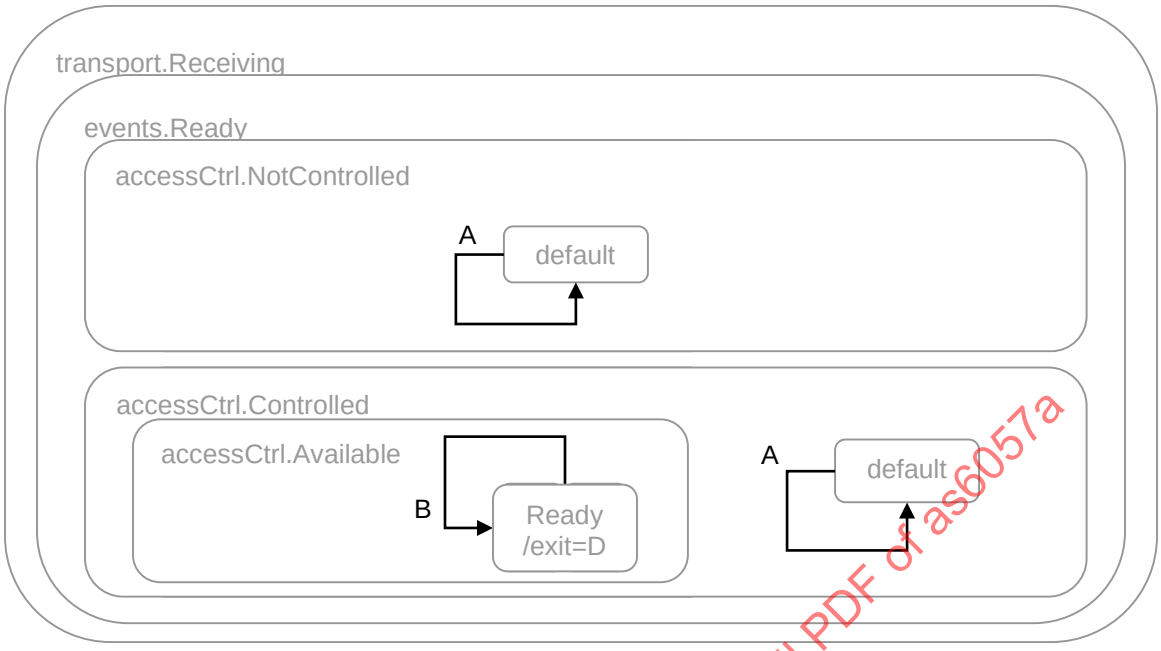


FIGURE 47 - PAN TILT JOINT POSITION DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 77 - PAN TILT JOINT POSITION DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 78 - PAN TILT JOINT POSITION DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommandedPan TiltJointPosition		sendReportCommanded PanTiltJointPosition
B	*SetPanTiltJointPosition*	*management.accessCtrl. isControllingClient AND panTiltMotionProfileExists*	*setPanTiltJointPosition*

TABLE 79 - PAN TILT JOINT POSITION DRIVER SERVICE STATE TRANSITION TABLE

Condition	Interpretation
panTiltMotionProfileExists	True if a motion profile is available.

TABLE 80 - PAN TILT JOINT POSITION DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommandedPan TiltJointPosition	Send a ReportCommandedPanTiltJoint Position message
setPanTiltJointPosition	Set the desired joint values for the pan tilt mechanism.
stopMotion	Stop motion of the pan tilt unit.

4.22 Pan Tilt Joint Velocity Driver Service

name= PanTiltJointVelocityDriver
version=2.0
id= urn:jaus:jss:manipulator:PanTiltJointVelocityDriver

Inherits-from Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

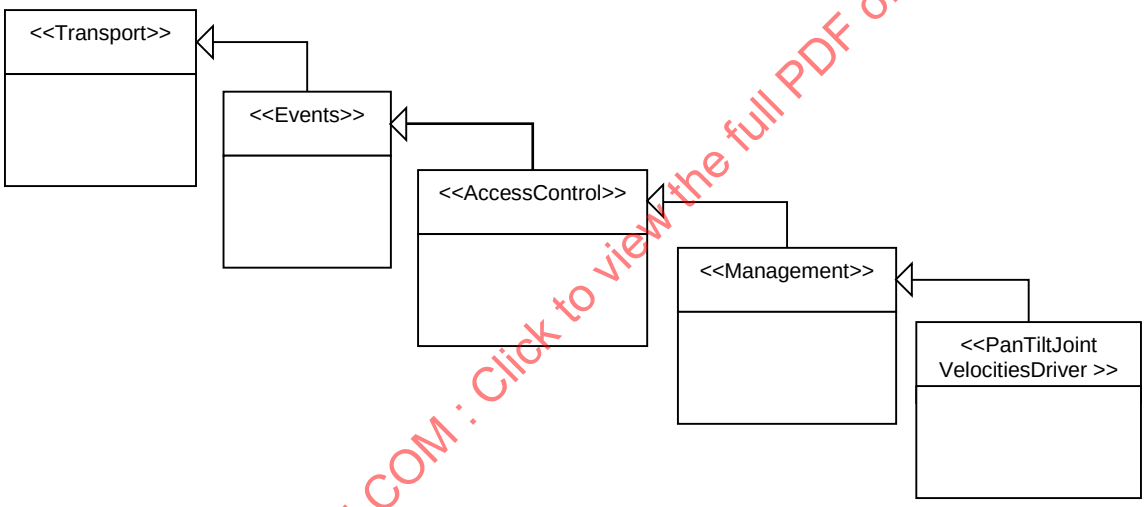


FIGURE 48 - PAN TILT JOINT VELOCITY DRIVER SERVICE

Description

The function of The Pan Tilt Joint Velocity Driver is to perform closed-loop joint velocity control. The input is the desired instantaneous desired joint velocities for the pan tilt mechanism. It is assumed that the pan tilt mechanism begins motion immediately after receiving the Set Pan Tilt Joint Velocity message. If backward compatibility with 1.0 implementations is required, then this service must be co-located on the same component as a Pan Tilt Specification Service and a Pan Tilt Motion Profile Service.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 81 lists the vocabulary of the PAN TILT JOINT VELOCITYJoint Velocity Driver Service.

TABLE 81 - PAN TILT JOINT VELOCITY DRIVER SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0623h	*SetPanTiltJointVelocity*	True
2631h	QueryCommandedPanTiltJointVelocity	False
Output Set		
4631h	ReportCommandedPanTiltJointVelocity	False

Protocol Behavior

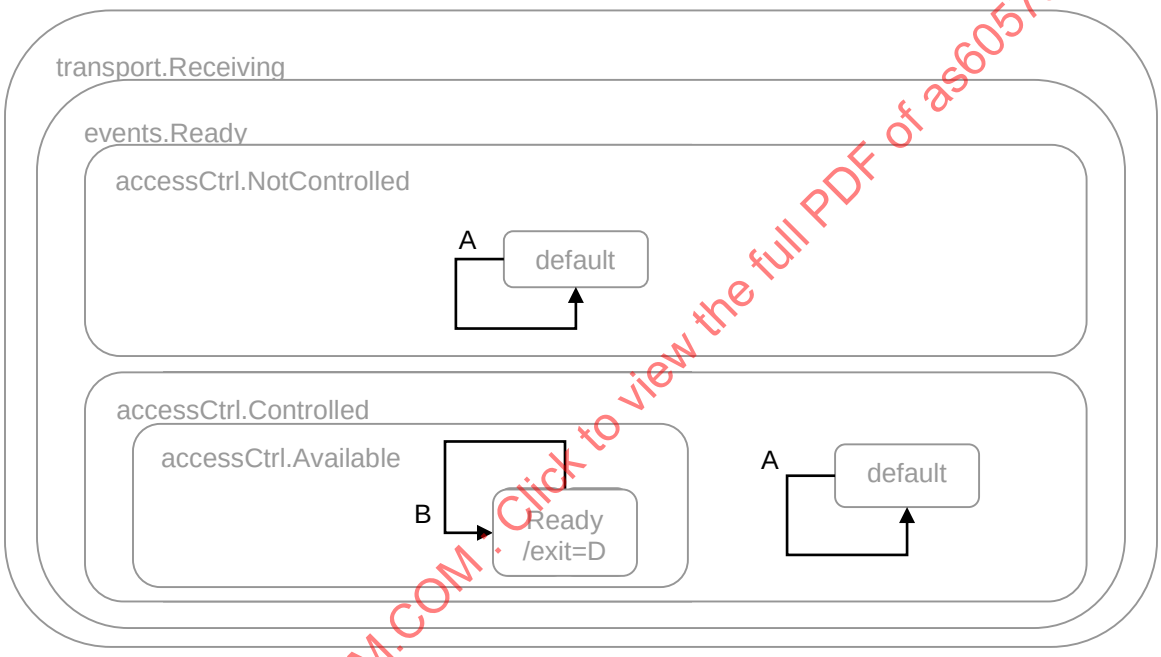


FIGURE 49 - PAN TILT JOINT VELOCITY DRIVER SERVICE PROTOCOL BEHAVIOR

TABLE 82 - PAN TILT JOINT VELOCITY DRIVER SERVICE ENTRY/EXIT ACTION TABLE

Label	State	Type	Guard	Actions
D	management. Ready	exit		*stopMotion*

TABLE 83 - PAN TILT JOINT VELOCITY DRIVER SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryCommandedPan TiltJointVelocity		sendReportCommanded PanTiltJointVelocity
B	*SetPanTiltJointVelocity*	*management.accessCtrl. isControllingClient AND panTiltMotionProfileExists*	*SetPanTiltJointVelocity*

TABLE 84 - PAN TILT JOINT VELOCITY DRIVER SERVICE STATE CONDITIONS TABLE

Condition	Interpretation
panTiltMotionProfileExists	True if a pan tilt motion profile is available.

TABLE 85 - PAN TILT JOINT VELOCITY DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportCommandedPan Tilt Joint Velocity	Send a ReportCommandedPanTiltJoint Velocity message.
SetPanTiltJointVelocity	Set the desired velocities for the individual joints of the pan tilt mechanism.
stopMotion	Stop motion of the pan tilt unit.

4.23 Pan Tilt Specification Service

name=PanTiltSpecificationService
version=2.0
id= urn:jaus:jss:manipulator:PanTiltSpecificationService

Inherits-from Events
  name=events
  id= urn:jaus:jss:core:Events
  version=1.1

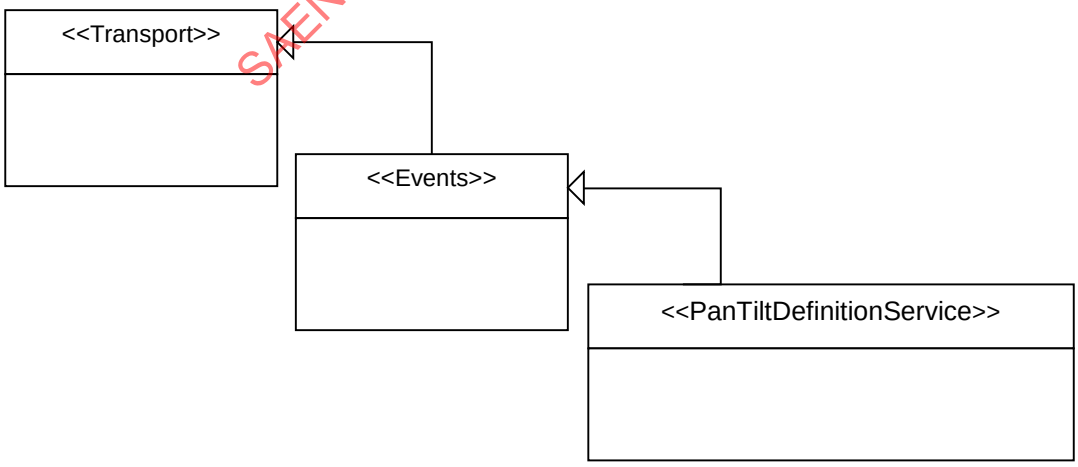


FIGURE 50 - PAN TILT SPECIFICATION SERVICE

Description

The function of the Pan Tilt Specification Service is to report the physical characteristics of a pan-tilt unit. The Report Pan Tilt Specification Message returns the minimum and maximum allowable value and the maximum velocity for each of the two joints as well as the position and orientation of the pan tilt base coordinate system relative to the vehicle coordinate system.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 86 lists the vocabulary of the Pan Tilt Description Service.

TABLE 86 - PAN TILT SPECIFICATION SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
2620h	QueryPanTiltSpecifications	False
Output Set		
4620h	ReportPanTiltSpecifications	False

Protocol Behavior

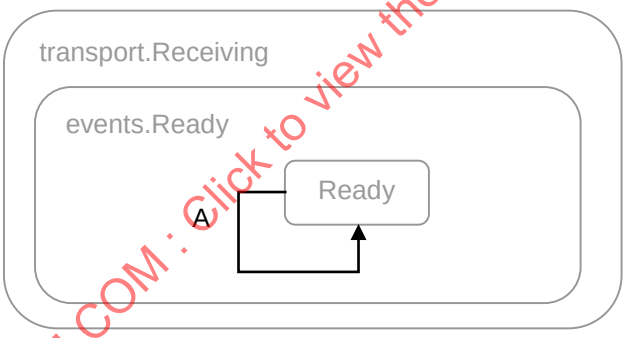


FIGURE 51 - PAN TILT DESCRIPTION SERVICE PROTOCOL BEHAVIOR

TABLE 87 - PAN TILT SPECIFICATION SERVICE STATE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryPanTiltSpecifications		sendReportPanTiltSpecifications

TABLE 88 - PAN TILT SPECIFICATION SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportPanTiltSpecifications	Send a Report Pan Tilt Specifications message.

4.24 Pan Tilt Motion Profile Service

name= PanTiltMotionProfileService
version=2.0
id= urn:jaus:jss:manipulator:PanTiltMotionProfileService

Inherits-from AccessControl
  name=accessControl
  id= urn:jaus:jss:core:AccessControl
  version=1.1

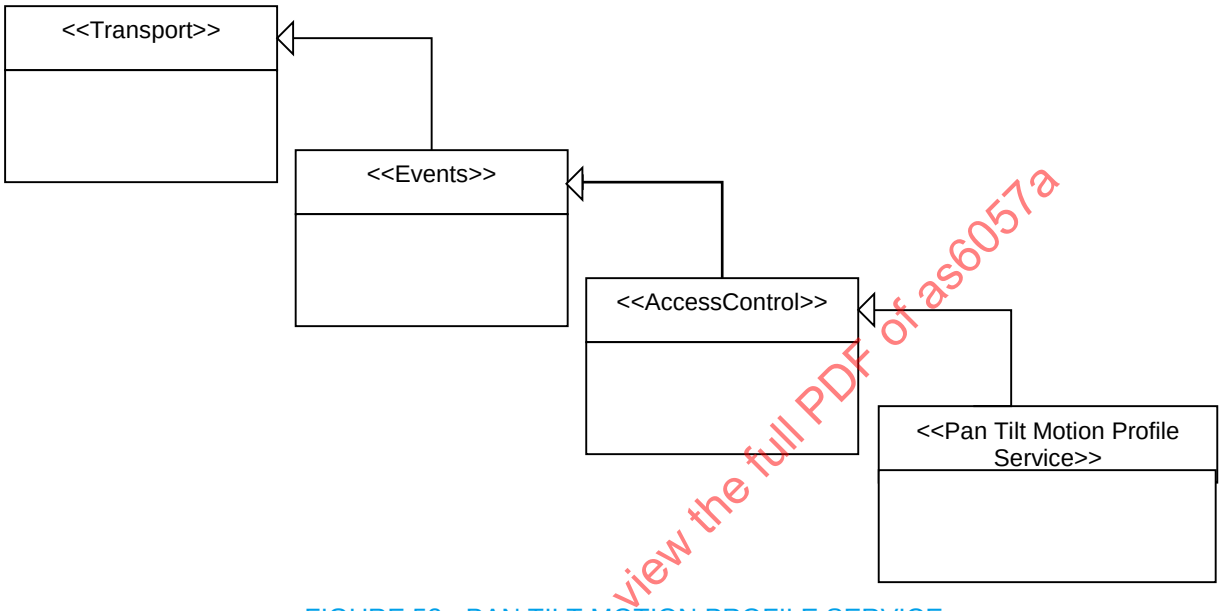


FIGURE 52 - PAN TILT MOTION PROFILE SERVICE

Description

The function of the Pan Tilt Motion Profile Service is to allow for configuration of the motion profile for all services co-located on this component. The Set Pan Tilt Motion Profile message is used to set maximum velocity and acceleration rates for each of the two variable joint parameters. All motions utilize the motion profile data that was most recently sent.

Assumptions

Messages may be delayed, lost or reordered.

Vocabulary

Table 89 lists the vocabulary of the Pan Tilt Motion Profile Service.

TABLE 89 - PAN TILT MOTION PROFILE SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0627h	*SetPanTiltMotionProfile*	True
2627h	QueryPanTiltMotionProfile	False
Output Set		
4627h	ReportPanTiltMotionProfile	False

Protocol Behavior

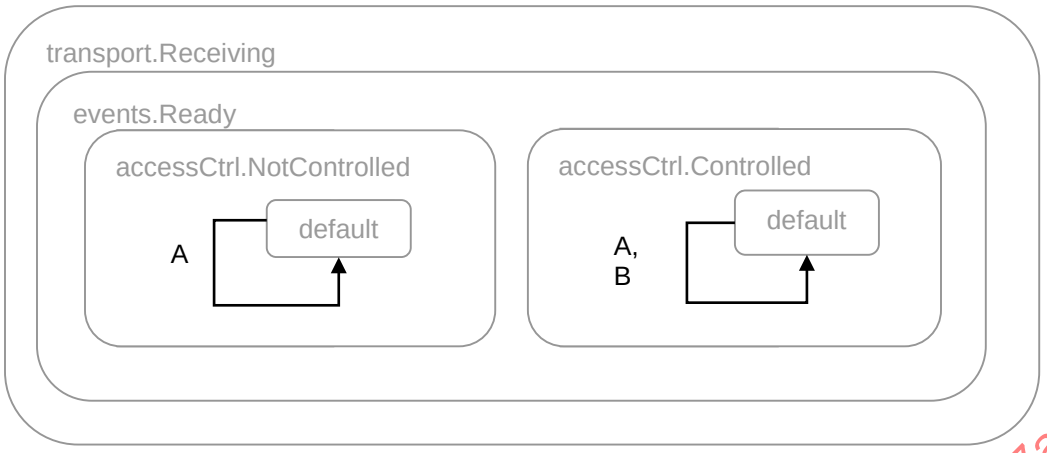


FIGURE 53 - PAN TILT MOTION PROFILE SERVICE PROTOCOL BEHAVIOR

TABLE 90 - PAN TILT MOTION PROFILE SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryPanTiltMotionProfile	*panTiltMotionProfileExists*	sendReportPanTiltMotionProfile
B	*SetPanTiltMotionProfile*	*management.accessCtrl.isControllingClient*	*setPanTiltMotionProfile*

TABLE 91 - PAN TILT MOTION PROFILE SERVICE STATE TRANSITION TABLE

Condition	Interpretation
panTiltMotionProfileExists	True if a pan tilt motion profile has been defined.

TABLE 92 - PAN TILT JOINT POSITION DRIVER SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportPanTiltMotionProfile	Send a ReportPanTiltMotionProfile message.
setPanTiltMotionProfile	Set the motion profile parameters for the pan tilt mechanism.

4.25 Primitive End Effector Service

Name= PrimitiveEndEffector
version=2.0
id= urn:jaus:jss:manipulator:PrimitiveEndEffector

Inherits-from Management
name=management
id= urn:jaus:jss:core:Management
version=1.1

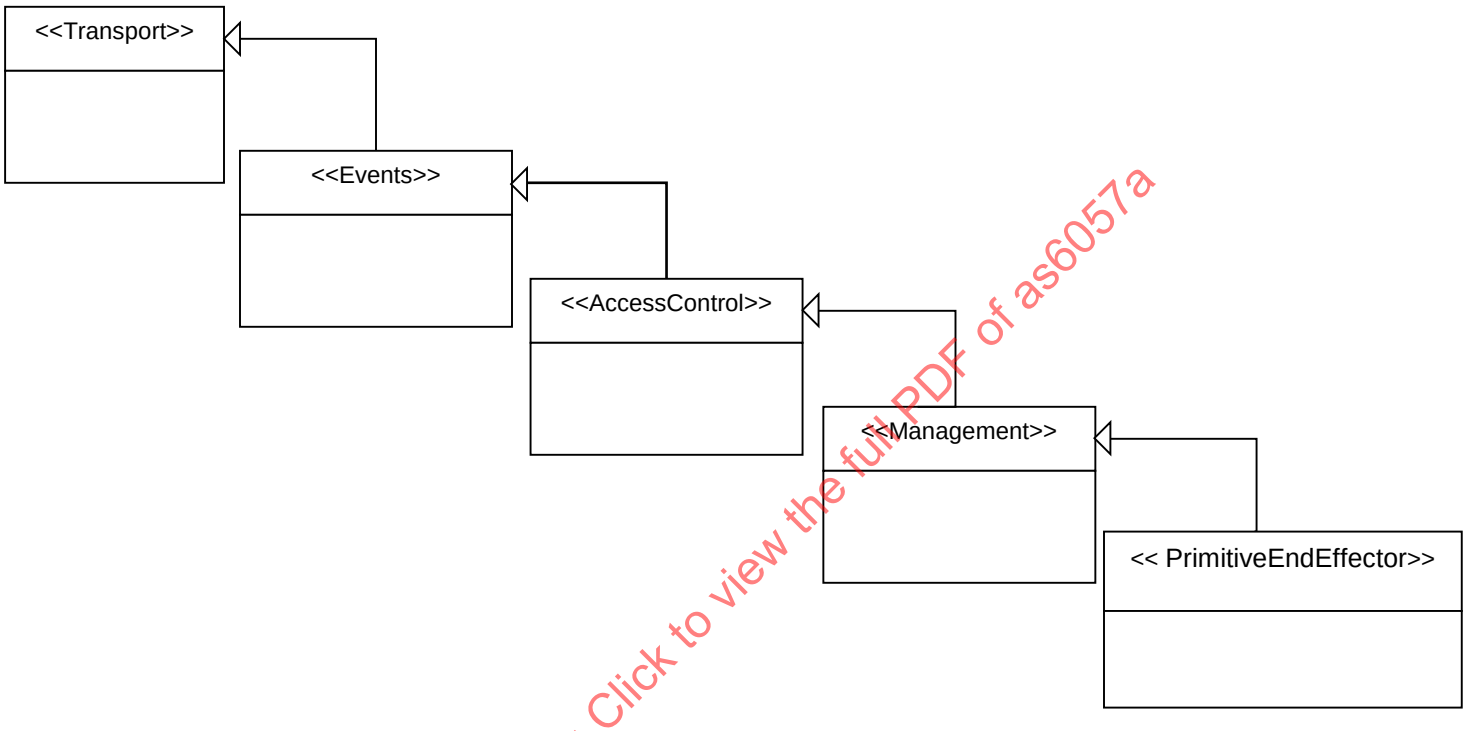


FIGURE 54 - PRIMITIVE END EFFECTOR SERVICE

Description

This service is the low level interface to an end effector. The End Effector is a one degree of freedom manipulator, usually mounted on the end of an *n*-degree of freedom manipulator.

Assumptions

Messages may be delayed, lost, or reordered.

Vocabulary

Table 93 lists the vocabulary of the Primitive End Effector Service.

TABLE 93 - PRIMITIVE END EFFECTOR SERVICE VOCABULARY

Message Id (hex)	Name	isCommand?
Input Set		
0633h	*SetEndEffectorEffort*	True
2632h	QueryEndEffectorSpecification	False
2633h	QueryEndEffectorEffort	False
Output Set		
4632h	ReportEndEffectorSpecification	False
4633h	ReportEndEffectorEffort	False

Protocol Behavior

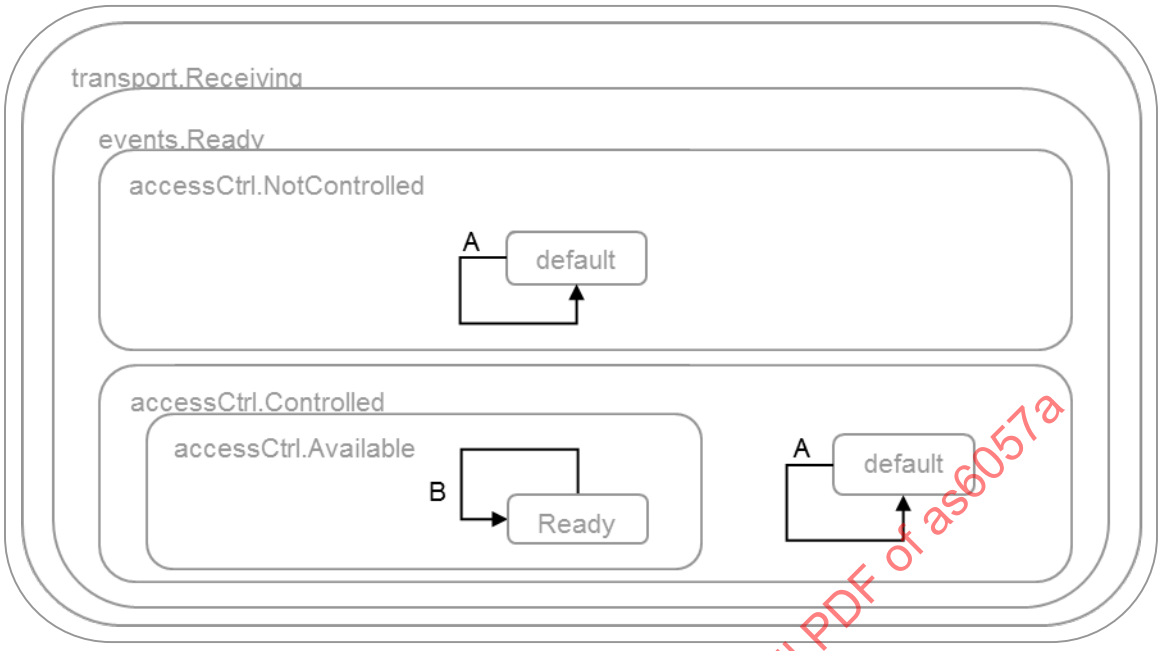


FIGURE 55 - PRIMITIVE END EFFECTOR SERVICE PROTOCOL BEHAVIOR

TABLE 94 - PRIMITIVE END EFFECTOR SERVICE TRANSITION TABLE

Label	Trigger	Guard	Actions
A	QueryEndEffector Specification		sendReportEndEffector Specification
	QueryEndEffector Effort		sendReportEndEffector Effort
B	*SetEndEffectorEffort*	*management.accessCtrl. isControllingClient*	*setEndEffectorEffort*

TABLE 95 - PRIMITIVE END EFFECTOR SERVICE TRANSITION ACTIONS

Action	Interpretation
sendReportEndEffector Specification	Send a Report End Effector Spec message.
sendReportEndEffectorEffort	Send a Report End Effector Effort message.
setEndEffectorEffort	Set the effort for the end effector.

5. DECLARED TYPES

5.1 CommandClass

5.1.1 ID 0601h: SetJointEffort

This message is used to control joint actuators in an open loop fashion. The command states the percentage level of effort that each actuator should apply. The message must contain effort commands for each joint in the manipulator.

TABLE 96 - SET JOINT EFFORT MESSAGE ENCODING

body
└ **list** name=JointEffortList
(count_field=unsigned byte)
record name=JointEffortRec

record name= JointEffortRec

Field #	Name	Type	Units	Optional?	Interpretation
1	**<fixed_field>** JointEffort	unsigned short integer	one	False	Percent of maximum effort for this joint. Each joint must have a corresponding entry in the list. Scaled Integer Lower Limit= -100 Upper Limit= 100

5.1.2 ID 0602h: SetJointPosition

This message sets the desired joint position values. The message must contain a position command for each joint in the manipulator.

TABLE 97 – SET JOINT POSITION MESSAGE ENCODING

body
└ **list** name=JointPositionList
(count_field=unsigned byte)
record name=JointPositionRec

record name=JointPositionRec

Field #	Name	Index	Type	Units	Optional?	Interpretation
1	**<variable_field>** JointPosition	1 - revolute	unsigned integer	radian	false	Lower limit = -8 π rad Upper limit = +8π rad
		2 - prismatic	unsigned integer	meter		Lower limit = -10 m Upper limit = +10 m
						Scaled integer representing the command position for this joint. Each joint must have a corresponding entry in the list. Units and scale range are based on the joint type.

5.1.3 ID 0603h: SetJointVelocity

This message sets the desired joint velocity values. The message must contain a velocity command for each joint in the manipulator.

TABLE 98 - SET JOINT VELOCITY MESSAGE ENCODING

body
└ **list** name=JointVelocityList
(count_field=unsigned byte)
record name=JointVelocityRec

record name= JointVelocityRec

Field #	Name	Index	Type	Units	Optional?	Interpretation
1	**<variable_field>** JointVelocity	1 - revolute	unsigned integer	radian per sec	false	Lower limit = -10π radian per second Upper limit = +10π radian per second
		2 - prismatic	unsigned integer	meter per sec		Lower limit = -5 m/sec Upper limit = +5 m/sec
						Scaled integer representing the command velocity for this joint. Each joint must have a corresponding entry in the list. Units and scale range are based on the joint type.

5.1.4 ID 0604h: SetToolOffset

This message specifies the coordinates of the end-effector tool point (End Effector Pose) in terms of the End Effector Coordinate System. For a six-axis robot, this coordinate system is defined by having its origin located at the intersection of the S6 joint axis vector and the user defined link vector a67. The Z axis of the coordinate system is along S6 and the X axis is along the a67 vector..

TABLE 99 - SET TOOL OFFSET MESSAGE ENCODING

body
└ **record** name=ToolPointRec

record name= ToolPointRec

Field #	Name	Type	Units	Optional?	Interpretation
1	ToolPointCoordinateX	unsigned integer	meter	false	Scaled Integer Lower limit = -15 m Upper limit = +15 m
2	ToolPointCoordinateY	unsigned integer	meter	false	see field 1
3	ToolPointCoordinateZ	unsigned integer	meter	false	see field 1

5.1.5 ID 0607h: SetJointMotionProfile

This message sets maximum speeds, acceleration, and deceleration rates for each joint of the manipulator. The message must contain a profile for each joint in the manipulator.

TABLE 100 - SET MOTION PROFILE MESSAGE ENCODING

body
└ **list** name=JointMotionProfileList
(count_field=unsigned byte)
variant name=JointType
(vtag_field = unsigned byte)
record name=RevoluteJointMotionProfileRec
record name=PrismaticJointMotionProfileRec

vtag = 1, record name=RevoluteJointMotionProfileRec

Field #	Name	Type	Units	Optional?	Interpretation
1	JointMaxSpeed	unsigned integer	radian per Sec	false	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second
2	JointMaxAccelerationRate	unsigned integer	radian per Sec^2	false	Scaled Integer Lower limit = 0 radian per second^2 Upper limit = +10π radian per second^2
3	JointMaxDecelerationRate	unsigned integer	radian per Sec^2	false	Scaled Integer Lower limit = 0 radian per second^2 Upper limit = +10π radian per second^2
vtag = 2, record name=PrismaticJointMotionProfileRec					
1	JointMaxSpeed	unsigned integer	meter per second	false	Scaled Integer Lower limit = 0 m/sec Upper limit = +5 m/sec
2	JointMaxAccelerationRate	unsigned integer	meter per second^2	false	Scaled Integer Lower limit = 0 m/sec^2 Upper limit = +20 m/sec^2
3	JointMaxDecelerationRate	unsigned integer	meter per second^2	false	Scaled Integer Lower limit = 0 m/sec^2 Upper limit = +20 m/sec^2

5.1.6 ID 0610h: SetEndEffectorPose

This message defines the desired end effector position and orientation. The coordinates of the tool point are defined in terms of the manipulator base coordinate system. The orientation of the end effector is defined by a unit quaternion (d ; a, b, c) which specifies the axis and angle of rotation that was used to establish the orientation of the end effector coordinate system with respect to the manipulator base coordinate system.

TABLE 101 - SET END EFFECTOR POSE MESSAGE ENCODING

body └ **record** name=EndEffectorPoseRec					
record name= EndEffectorPoseRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	ToolPointCoordinateX	unsigned integer	meter	false	Scaled Integer Lower limit = -30 m Upper limit = +30 m
2	ToolPointCoordinateY	unsigned integer	meter	false	see field 1
3	ToolPointCoordinateZ	unsigned integer	meter	false	see field 1
4	dComponentOfUnitQuaternionQ	unsigned integer	one	false	Scaled Integer Lower limit = -1 Upper limit = +1
5	aComponentOfUnitQuaternionQ	unsigned integer	one	false	see field 4
6	bComponentOfUnitQuaternionQ	unsigned integer	one	false	see field 4
7	cComponentOfUnitQuaternionQ	unsigned integer	one	false	see field 4

5.1.7 ID 0612h: SetEndEffectorVelocityState

This message defines the desired end effector velocity state measured with respect to the manipulator base coordinate system.

TABLE 102 - SET END EFFECTOR VELOCITY STATE MESSAGE ENCODING

body └ **record** name=EndEffectorVelocityRec					
record name= EndEffectorVelocityRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	AngularVelocityComponent X	unsigned integer	radian per second	false	Scaled Integer Lower limit = -20 π radian per second Upper limit = +20 π radian per second
2	AngularVelocityComponent Y	unsigned integer	radian per second	false	see field 1
3	AngularVelocityComponent Z	unsigned integer	radian per second	false	see field 1
4	LinearVelocityComponentX	unsigned integer	meter per second	false	Scaled Integer Lower limit = -10 m/sec Upper limit = +10 m/sec
5	LinearVelocityComponentY	unsigned integer	meter per second	false	see field 4
6	LinearVelocityComponentZ	unsigned integer	meter per second	false	see field 4

5.1.8 ID 0613h: SetActuatorForceTorques

This message defines the desired actuator forces (for each prismatic actuator) and the desired actuator torques (for each revolute actuator).

TABLE 103 - SET ACTUATOR FORCE TORQUES MESSAGE ENCODING

body
└ **list** name=ActuatorForceTorqueList
(count_field=unsigned byte)
record name=ActuatorForceTorqueRec

record name= ActuatorForceTorqueRec

Field #	Name	Index	Type	Units	Optional?	Interpretation
1	**<variable_field>** JointForceTorque	1 - revolute	unsigned integer	newton meter	false	Lower limit = -1000 Nm Upper limit = +1000 Nm
		2 - prismatic	unsigned integer	newtons		Lower limit = -500 N Upper limit = +500 N
						Scaled integer representing the command force or torque for this joint. Each joint must have a corresponding entry in the list.

5.1.9 ID 061Eh: ExecuteList

This message is used to begin execution of a sequential list of motion commands. Optionally, the starting element UID can also be specified, where an unspecified value or a value of zero (0) indicates the first (head) element in the list.

TABLE 104 - EXECUTE LIST MESSAGE ENCODING

body
└ **record** name=ExecuteListRec

record name= ExecuteListRec

Field #	Name	Type	Units	Optional?	Interpretation
1	**<presence_vector>**	unsigned byte	one	false	
2	**<fixed_field>** ElementUID	unsigned short integer	one	True	Element UID of the starting element. A value of zero (0) indicates the first (head) element of the list.

5.1.10 ID 0621h: SetPanTiltJointEffort

This message is used to control the two joint actuators for a pan tilt mechanism in an open loop fashion. The command states the percentage level of effort that each actuator should exercise in order to move its corresponding joint.

TABLE 105 - SET PAN TILT JOINT EFFORT MESSAGE ENCODING

body └ **record** name=PanTiltJointEffortRec					
record name= PanTiltJointEffortRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<fixed_field>** Joint1effort	unsigned short integer	percent	False	Percent of maximum. Scaled Integer Lower Limit= -100 Upper Limit= 100
2	**<fixed_field>** Joint2effort	unsigned short integer	percent	False	Percent of maximum. Scaled Integer Lower Limit= -100 Upper Limit= 100

5.1.11 ID 0622h: SetPanTiltJointPosition

This message sets the desired joint position values for a pan tilt mechanism.

TABLE 106 – SET PAN TILT JOINT POSITION MESSAGE ENCODING

body └ **record** name=PanTiltJointPositionRec					
record name= PanTiltJointPositionRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<fixed_field>** Joint1position	unsigned integer	radian	false	Scaled integer: Lower limit = -8 π rad Upper limit = +8π rad
2	**<fixed_field>** Joint2position	unsigned integer	radian	false	see field 1

5.1.12 ID 0623h: SetPanTiltJointVelocity

This message sets the desired joint velocity values for a pan tilt mechanism.

TABLE 107 - SET PAN TILT JOINT VELOCITY MESSAGE ENCODING

body └ **record** name=PanTiltJointVelocityRec					
record name= PanTiltJointVelocityRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<fixed_field>** Joint1velocity	unsigned integer	radian per second	false	Scaled Integer Lower limit = -10π radian per second Upper limit = +10π radian per second
2	**<fixed_field>** Joint2velocity	unsigned integer	radian per second	false	see field 1

5.1.13 ID 0627h: SetPanTiltMotionProfile

This message sets maximum speeds, acceleration, and deceleration rates for each joint of the pan tilt mechanism.

TABLE 108 - SET PAN TILT MOTION PROFILE MESSAGE ENCODING

body └ **record** name= PanTiltMotionProfileRec					
record name= PanTiltMotionProfileRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	Joint1MaxSpeed	unsigned integer	radians per second	false	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second
2	Joint1MaxAccelerationRate	unsigned integer	radian per second squared	false	Scaled Integer Lower limit = 0 radian per second^2 Upper limit = +10π radian per second^2
3	Joint1MaxDecelerationRate	unsigned integer	radian per second squared	false	Scaled Integer Lower limit = 0 radian per second^2 Upper limit = +10π radian per second^2
4	Joint2MaxSpeed	unsigned integer	radian per second	false	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second
5	Joint2MaxAccelerationRate	unsigned integer	radian per second squared	false	Scaled Integer Lower limit = 0 radian per second^2 Upper limit = +10π radian per second^2
6	Joint2MaxDecelerationRate	unsigned integer	radian per second squared	false	Scaled Integer Lower limit = 0 radian per second^2 Upper limit = +10π radian per second^2

5.1.14 ID 0633h: SetEndEffectorEffort

This message is used to control a single degree of freedom end effector in an open loop fashion. The command states the percentage level of effort that the end effector should exercise. The mapping of effort to end effector behavior (open/close gripper, turn a screw driver, activate a welder) is implementation specific.

TABLE 109 - SET END EFFECTOR EFFORT MESSAGE ENCODING

body └ **record** name=EndEffectorEffortRec					
record name= EndEffectorEffortRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<fixed_field>** EndEffectorEffort	unsigned short integer	percent	False	Percent of maximum. Scaled Integer Lower Limit= -100 Upper Limit= 100

5.2 QueryClass

5.2.1 ID 2600h: QueryManipulatorSpecifications

This message shall cause the receiving service to reply to the requestor with a ID 4600h: ReportManipulatorSpecifications message. A logical AND shall be performed on the requested presence vector and that representing the available fields from the responder. The resulting message shall contain the fields indicated by the result of this logical AND operation.

TABLE 110 - QUERY MANIPULATOR SPECIFICATIONS MESSAGE ENCODING

body
└ **record** name= QueryManipulatorSpecificationsRec

record name= QueryManipulatorSpecificationsRec

Field #	Name	Type	Units	Optional?	Interpretation
1	**<fixed_field>** PresenceVector	unsigned byte	one	false	Specifies if the corresponding Report message should include the Manipulator Coordinate System.

5.2.2 ID 2601h: QueryJointEffort

This message shall cause the receiving service to reply to the requestor with a ID 4601h: ReportJointEffort message.

TABLE 111 - QUERY JOINT EFFORT MESSAGE ENCODING

body
└ **empty**

5.2.3 ID 2602h: QueryJointPosition

This message shall cause the receiving service to reply to the requestor with a ID 4602h: ReportJointPosition message.

TABLE 112 - QUERY JOINT POSITION MESSAGE ENCODING

body
└ **empty**

5.2.4 ID 2603h: QueryJointVelocity

This message shall cause the receiving service to reply to the requestor with a ID 4603h: ReportJointVelocity message.

TABLE 113 - QUERY JOINT VELOCITY MESSAGE ENCODING

body
└ **empty**

5.2.5 ID 2604h: QueryToolOffset

This message shall cause the receiving service to reply to the requestor with a ID 4604h: ReportToolOffset message.

TABLE 114 - QUERY TOOL OFFSET MESSAGE ENCODING

body └ **empty**

5.2.6 ID 2605h: QueryJointForceTorque

This message shall cause the receiving service to reply to the requestor with a ID 4605h: ReportJointForceTorque message.

TABLE 115 - QUERY JOINT FORCE TORQUE MESSAGE ENCODING

body └ **empty**

5.2.7 ID 2607h: QueryJointMotionProfile

This message shall cause the receiving service to reply to the requestor with a ID 4607h: ReportJointMotionProfile message.

TABLE 116 - QUERY JOINT MOTION PROFILE MESSAGE ENCODING

body └ **empty**

5.2.8 ID 2608h: QueryCommandedJointPosition

This message shall cause the receiving service to reply to the requestor with a ID 4608h: ReportCommandedJointPosition message.

TABLE 117 - QUERY COMMANDED JOINT POSITION MESSAGE ENCODING

body └ **empty**

5.2.9 ID 2610h: QueryCommandedEndEffectorPose

This message shall cause the receiving service to reply to the requestor with a ID 4610h: ReportCommandedEnd EffectorPose message.

TABLE 118 - QUERY COMMANDED END EFFECTOR POSE MESSAGE ENCODING

body └ **empty**

5.2.10 ID 2611h: QueryCommandedJointVelocity

This message shall cause the receiving service to reply to the requestor with a ID 4611h: ReportCommandedJoint Velocity message.

TABLE 119 - QUERY COMMANDED JOINT VELOCITY MESSAGE ENCODING

body └ **empty**

5.2.11 ID 2612h: QueryCommandedEndEffectorVelocityState

This message shall cause the receiving service to reply to the requestor with a ID 4612h: ReportCommandedEnd EffectorVelocityState message.

TABLE 120 - QUERY COMMANDED END EFFECTOR VELOCITY STATE MESSAGE ENCODING

body └ **empty**

5.2.12 ID 2613h: QueryCommandedActuatorForceTorque

This message shall cause the receiving service to reply to the requestor with a ID 4613h: ReportCommandedActuatorForceTorque message.

TABLE 121 - QUERY COMMANDED ACTUATOR FORCE TORQUE MESSAGE ENCODING

body └ **empty**

5.2.13 ID 2615h: QueryEndEffectorPose

This message shall cause the receiving service to reply to the requestor with a ID 4615h: ReportEndEffectorPose message.

TABLE 122 - QUERY END EFFECTOR POSE MESSAGE ENCODING

body └ **empty**

5.2.14 ID 2616h: QueryEndEffectorVelocityState

This message shall cause the receiving service to reply to the requestor with a ID 4616h: ReportEnd EffectorVelocityState message.

TABLE 123 - QUERY END EFFECTOR VELOCITY STATE MESSAGE ENCODING

body └ **empty**

5.2.15 ID 261Eh: QueryActiveElement

This message is used to query the current active element of an executing list.

TABLE 124 - QUERY ACTIVE ELEMENT MESSAGE ENCODING

body └ **record** name=(empty)

5.2.16 ID 2620h: QueryPanTiltSpecifications

This message shall cause the receiving service to reply to the requestor with a ID 4620h: ReportPanTiltSpecifications message.

TABLE 125 - QUERY PAN TILT SPECIFICATIONS MESSAGE ENCODING

body └ **record** name= QueryPanTiltSpecificationsRec					
record name= QueryPanTiltSpecificationsRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<fixed_field>** PresenceVector	unsigned byte	one	false	See Report Pan Tilt Specifications Message

5.2.17 ID 2621h: QueryPanTiltJointEffort

This message shall cause the receiving service to reply to the requestor with a ID 4621h: ReportPanTiltJointEffort message.

TABLE 126 - QUERY PAN TILT JOINT EFFORT MESSAGE ENCODING

body └ **empty**

5.2.18 ID 2622h: QueryPanTiltJointPositions

This message shall cause the receiving service to reply to the requestor with a ID 4622h: ReportPanTiltJointPositions message.

TABLE 127 - QUERY PAN TILT JOINT POSITIONS MESSAGE ENCODING

body └ **empty**

5.2.19 ID 2623h: QueryPanTiltJointVelocity

This message shall cause the receiving service to reply to the requestor with a ID 4623h: ReportPanTiltJointVelocity message.

TABLE 128 - QUERY PAN TILT JOINT VELOCITY MESSAGE ENCODING

body └ **empty**

5.2.20 ID 2627h: QueryPanTiltMotionProfile

This message shall cause the receiving service to reply to the requestor with a ID 4627h: ReportPanTiltMotionProfile message.

TABLE 129 - QUERY PAN TILT MOTION PROFILE MESSAGE ENCODING

body └ **empty**

5.2.21 ID 2628h: QueryCommandedPanTiltJointPositions

This message shall cause the receiving service to reply to the requestor with a ID 4628h: ReportCommandedPanTiltJointPositions message.

TABLE 130 - QUERY COMMANDED PAN TILT JOINT POSITIONS MESSAGE ENCODING

body └ **empty**

5.2.22 ID 2631h: QueryCommandedPanTiltJointVelocity

This message shall cause the receiving service to reply to the requestor with a ID 4631h: ReportCommandedPanTiltJoint Velocitiy message.

TABLE 131 - QUERY COMMANDED PAN TILT JOINT VELOCITY MESSAGE ENCODING

body └ **empty**

5.2.23 ID 2632h: QueryEndEffectorSpecification

This message shall cause the receiving service to reply to the requestor with a ID 4632h: ReportEndEffectorSpecification message.

TABLE 132 - QUERY END EFFECTOR SPECIFICATION MESSAGE ENCODING

body └ **empty**

5.2.24 ID 2633h: QueryEndEffectorEffort

This message shall cause the receiving service to reply to the requestor with a ID 4633h: ReportEndEffectorEffort message.

TABLE 133 - QUERY END EFFECTOR EFFORT MESSAGE ENCODING

body └ **empty**

5.3 InformClass

5.3.1 ID 4600h: ReportManipulatorSpecifications

This message provides the specification of the manipulator including the number of joints, the link length and twist angle of each link, the joint offset (for revolute joints) or joint angle (for prismatic joints), the minimum and maximum value for each joint, and the minimum and maximum speed for each joint.

The record ManipulatorCoordinateSystemRec establishes the position and orientation relationship of the manipulator base coordinate system to the vehicle coordinate system. It specifies the location (x,y,z) and orientation (described by the quaternion) of the manipulator base coordinate system relative to the vehicle coordinate system.

The variant FirstJointParameters specifies the offset distance from the origin of the manipulator coordinate system to the first link axis (L1 in Figure 1) or the joint angle between the x-axis of the manipulator coordinate system and the line along the first link (φ1 in Figure 1), depending on whether the first joint in JointSpecificationList is revolute or prismatic.

The record RevoluteJointSpecificationRec specifies the link length and twist angle for a link and the joint offset for the revolute joint that comes after the link. The record PrismaticJointSpecificationRec specifies the link length and twist angle for a link and the joint angle for the prismatic joint that comes after the link.

The order in which the joints are encoded into the list called JointSpecificationList must correspond with the order of the joints from the manipulator base to the end effector.

In Revision A, an optional list is appended to this message to allow for a human readable description of each joint. If the list is specified (presence vector bit is set), the size of the list shall be the same as the total number of joints in the manipulator with the order of the names from the manipulator base to the end effector.

TABLE 134 - REPORT MANIPULATOR SPECIFICATIONS MESSAGE ENCODING

body
└ **sequence** name=ReportManipulatorSpecification
(**presence_vector**=unsigned byte)
record name=ManipulatorCoordinateSystemRec **(optional)**
variant name= FirstJointParameters
(vtag_field=unsigned byte)
record name= RevoluteJoint1OffsetRec
record name= PrismaticJoint1AngleRec
list name=JointSpecificationList
(count_field=unsigned byte)
variant name=JointSpecifications
(vtag_field=unsigned byte)
record name=RevoluteJointSpecificationRec
record name=PrismaticJointSpecificationRec
list name=JointNamesList **(optional)**
(count_field=unsigned byte)
record name=JointNameRec

record name= ManipulatorCoordinateSystemRec, **optional=true**

Field #	Name	Type	Units	Optional?	Interpretation
1	**<fixed_field>** manipulator coordinate sys. x	unsigned integer	meter	false	x coordinate of origin of manipulator base coordinate system measured with respect to vehicle coordinate system Scaled integer: Lower limit = -30 m Upper limit = +30 m
2	**<fixed_field>** manipulator coordinate sys. y	unsigned integer	meter	false	y coordinate of origin of manipulator base coordinate system measured with respect to vehicle coordinate system Scaled integer: Lower limit = -30 m Upper limit = +30 m
3	**<fixed_field>** manipulator coordinate sys. z	unsigned integer	meter	false	z coordinate of origin of manipulator base coordinate system measured with respect to vehicle coordinate system Scaled integer: Lower limit = -30 m Upper limit = +30 m
4	**<fixed_field>** d component of unit quaternion q	unsigned integer	one	false	quaternion q = d + ai +bj + ck defines the orientation of the manipulator base coordinate system measured with respect to the vehicle coordinate system Scaled integer: Lower limit = -1 Upper limit = +1
5	**<fixed_field>** a component of unit quaternion q	unsigned integer	one	false	see field 4
6	**<fixed_field>** b component of unit quaternion q	unsigned integer	one	false	see field 4

7	**<fixed_field>** c component of unit quaternion q	unsigned integer	one	false	see field 4
variant name= FirstJointParameters					
vtag= 0, record name= RevoluteJoint1OffsetRec					
1	**<presence_vector>**	unsigned byte	one	false	
2	**< fixed_field> Joint1Offset**	Unsigned short integer	meter	false	Joint offset Scaled value Lower limit = -10 m Upper limit = +10 m
3	**< fixed_field>** Joint 1 – Min value	unsigned integer	radian	true	Scaled value Lower limit = -8 π rad Upper limit = +8π rad Note: This field is omitted by using the presence vector for joints that can rotate continuously without limit.
4	**< fixed_field>** Joint 1 – Max value	unsigned integer	radian	true	Scaled value Lower limit = -8 π rad Upper limit = +8π rad Note: This field is omitted by using the presence vector for joints that can rotate continuously without limit.
5	**< fixed_field>** Joint 1 – Max speed	unsigned integer	radian per second	true	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second
6	**< fixed_field>** Joint 1 – Max torque	unsigned integer	N m	true	Scaled value Lower limit = 0 N m Upper limit = 5000 N m
7	**< fixed_field>** OffsetBoundingCylinderRadius	Unsigned short integer	Meter	true	The radius of an imaginary bounding cylinder whose length lies along the joint offset axis. Scaled value Lower limit = 0 m Upper limit = 10 m
vtag= 1, record name= PrismaticJoint1AngleRec					
1	**<presence_vector>**	unsigned byte	one	false	
2	**< fixed_field> Joint1Angle**	unsigned short Integer	radian	false	Scaled value Lower limit = - π rad Upper limit = +π rad
3	**< fixed_field>** Joint 1 – Min value	unsigned integer	meter	false	Scaled value Lower limit = -10 m Upper limit = +10 m
4	**< fixed_field>** Joint 1 – Max value	unsigned integer	meter	false	Scaled value Lower limit = -10 m Upper limit = +10 m
5	**< fixed_field>** Joint 1 – Max speed	unsigned integer	meter per second	true	Scaled value Lower limit = -5 m/sec Upper limit = +5 m/sec
6	**< fixed_field>** Joint 1 – Max force	unsigned integer	N	true	Scaled value Lower limit = 0 N Upper limit = 5000 N

7	**< fixed_field>** JointBoundingCylinderRadius	Unsigned short integer	Meter	true	The radius of an imaginary bounding cylinder whose length lies along the joints moving axis. Scaled value Lower limit = 0 m Upper limit = 10 m
list name=JointSpecificationList					
vtag= 0, record name= RevoluteJointSpecificationRec					
1	**<presence_vector >**	unsigned byte	one	false	
2	**< fixed_field>** Link a(*i*)(*i+1*) – Link Length	unsigned short Integer	meter	false	Link Length Scaled value Lower limit = -10 m Upper limit = +10 m
3	**< fixed_field>** Link a(*i*)(*i+1*)– Twist Angle	unsigned short Integer	radian	false	Scaled value Lower limit = -π rad Upper limit = +π rad
4	**< fixed_field>** Joint *(i+1)* – Offset	unsigned short Integer	meter	false	Joint offset Scaled value Lower limit = -10 m Upper limit = +10 m
5	**< fixed_field>** Joint *(i+1)* – Min value	unsigned integer	radian	true	Scaled value Lower limit = -8 π rad Upper limit = +8π rad Note: This field is omitted by using the presence vector for joints that can rotate continuously without limit.
6	**< fixed_field>** Joint *(i+1)* – Max value	unsigned integer	radian	true	Scaled value Lower limit = -8 π rad Upper limit = +8π rad Note: This field is omitted by using the presence vector for joints that can rotate continuously without limit.
7	**< fixed_field>** Joint *(i+1)* – Max speed	unsigned integer	radian per second	true	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second
8	**< fixed_field>** Joint *(i+1)* – Max torque	unsigned integer	N m	true	Scaled value Lower limit = 0 N m Upper limit = 5000 N m
9	**< fixed_field>** OffsetBoundingCylinderRadius	Unsigned short integer	Meter	true	The radius of an imaginary bounding cylinder whose length lies along the joint offset axis. Scaled value Lower limit = 0 m Upper limit = 10 m
10	**< fixed_field>** LinkLengthBoundingCylinderRadius	Unsigned short integer	Meter	true	The radius of an imaginary bounding cylinder whose length lies along the link length axis. Scaled value Lower limit = 0 m Upper limit = 10 m
vtag=1, record name= PrismaticJointSpecificationRec					
1	**<presence_vector >**	unsigned byte	one	false	

2	**< fixed_field>** Link a(*i)(i+1)* – Link Length	unsigned short Integer	meter	false	Link Length Scaled value Lower limit = -10 m Upper limit = +10 m .
3	**< fixed_field>** Link a(*i)(i+1)*– Twist Angle	unsigned short Integer	radian	false	Scaled value Lower limit = - π rad Upper limit = +π rad
4	**< fixed_field>** Joint *(i+1)* – Angle	unsigned short Integer	radian	false	Scaled value Lower limit = - π rad Upper limit = +π rad
5	**< fixed_field>** Joint *(i+1)* – Min value	unsigned integer	meter	false	Scaled value Lower limit = -10 m Upper limit = +10 m
6	**< fixed_field>** Joint *(i+1)* – Max value	unsigned integer	meter	false	Scaled value Lower limit = -10 m Upper limit = +10 m
7	**< fixed_field>** Joint *(i+1)* – Max speed	unsigned integer	meter per second	true	Scaled value Lower limit = -5 m/sec Upper limit = +5 m/sec
8	**< fixed_field>** Joint *(i+1)*– Max force	unsigned integer	N	true	Scaled value Lower limit = 0 N Upper limit = 5000 N
9	**< fixed_field>** JointBoundingCylinderRadius	Unsigned short integer	Meter	true	The radius of an imaginary bounding cylinder whose length lies along the joints moving axis. Scaled value Lower limit = 0 m Upper limit = 10 m
10	**< fixed_field>** LinkLengthBoundingCylinderRadius	Unsigned short integer	Meter	true	The radius of an imaginary bounding cylinder whose length lies along the link length axis. Scaled value Lower limit = 0 m Upper limit = 10 m
list name=JointNamesList **(optional)**					
record name=JointNameRec					
1	**<variable_length_str>** Description	count_field= unsigned_byte	one	false	A human-readable string that can be used to label each joint on a user interface.

5.3.2 ID 4601h: ReportJointEffort

This message is used to provide the receiver the percent effort that is currently being applied to the individual manipulator joints. The message data for this message is identical to ID 0601h: SetJointEffort.

TABLE 135 - REPORT JOINT EFFORT MESSAGE ENCODING

body
└ **list** name=JointEffortList
(count_field=unsigned byte)
record name=JointEffortRec

5.3.3 ID 4602h: ReportJointPosition

This message provides the receiver with the current values of the joint positions. The message data and mapping of the presence vector for this message are identical to ID 0602h:SetJointPosition.

TABLE 136 - REPORT JOINT POSITION MESSAGE ENCODING

body
└ **list** name=JointPositionList
(count_field=unsigned byte)
record name=JointPositionRec

5.3.4 ID 4603h: ReportJointVelocity

This message provides the receiver with the current values of the joint velocities. The message data and mapping of the presence vector for this message are identical to ID 0603h:SetJointVelocitiy.

TABLE 137 - REPORT JOINT VELOCITY MESSAGE ENCODING

body
└ **list** name=JointVelocityList
(count_field=unsigned byte)
record name=JointVelocityRec

5.3.5 ID 4604h: ReportToolOffset

This message provides the receiver with the current values for the tool point as measured in the end effector coordinate system. The message data and mapping of the presence vector for this message are identical to ID 0604h:SetToolOffset

TABLE 138 - REPORT TOOL OFFSET MESSAGE ENCODING

body
└ **record** name=ToolPointRec

5.3.6 ID 4605h: ReportJointForceTorques

This message provides the receiver with the current values of the torques applied to revolute joints and forces applied to prismatic joints of the manipulator. The message data are identical to ID 0613h:SetActuatorForceTorques.

TABLE 139 - REPORT JOINT FORCE TORQUE MESSAGE ENCODING

```
body
└─ list name=ActuatorForceTorqueList
       (count_field=unsigned byte)
       record name=ActuatorForceTorqueRec
```

5.3.7 ID 4607h: ReportJointMotionProfile

This message provides the receiver with the current motion profile. The message data for this message are identical to ID 0607h:SetJointMotionProfile.

TABLE 140 - REPORT MOTION PROFILE MESSAGE ENCODING

```
body
└─ list name=JointMotionProfileList
       (count_field=unsigned byte)
       variant name=JointType
           (vtag_field = unsigned byte)
           record name=RevoluteJointMotionProfileRec
           record name=PrismaticJointMotionProfileRec
```

5.3.8 ID 4608h: ReportCommandedJointPosition

This message provides the receiver with the commanded joint positions. The message data for this message are identical to ID 0602h:SetJointPosition.

TABLE 141 - REPORT COMMANDED JOINT POSITION MESSAGE ENCODING

```
body
└─ list name=JointPositionList
       (count_field=unsigned byte)
       record name=JointPositionRec
```

5.3.9 ID 4610h: ReportCommandedEndEffectorPose

This message provides the receiver with the commanded pose of the end effector. The message data for this message are identical to ID 0610h:SetEndEffectorPose.

TABLE 142 - REPORT COMMANDED END EFFECTOR POSE MESSAGE ENCODING

```
body
└─ record name=EndEffectorPoseRec
```

5.3.10 ID 4611h: ReportCommandedJointVelocity

This message provides the receiver with the commanded joint velocities for the manipulator. The message data for this message are identical to ID 0603h:SetJointVelocitiy.

TABLE 143 - REPORT COMMANDED JOINT VELOCITY MESSAGE ENCODING

body
└ **list** name=JointVelocityList
(count_field=unsigned byte)
record name=JointVelocityRec

5.3.11 ID 4612h: ReportCommandedEndEffectorVelocityState

This message provides the receiver with the commanded end effector velocity state for the manipulator. The message data for this message are identical to ID 0612h:SetEndEffectorVelocityState.

TABLE 144 - REPORT COMMANDED END EFFECTOR VELOCITY STATE MESSAGE ENCODING

body
└ **record** name=EndEffectorVelocityRec

5.3.12 ID 4613h: ReportCommandedActuatorForceTorque

This message provides the receiver with the commanded actor force/torques for the manipulator. The message data for this message are identical to ID 0613h: SetActuatorForceTorques.

TABLE 145 - REPORT COMMANDED ACTUATOR FORCE TORQUE MESSAGE ENCODING

body
└ **list** name=ActuatorForceTorqueList
(count_field=unsigned byte)
record name=ActuatorForceTorqueRec

5.3.13 ID 4615h: ReportEndEffectorPose

This message provides the receiver with the current pose of the end effector. The message data for this message are identical to ID 0610h:SetEndEffectorPose.

TABLE 146 - REPORT COMMANDED END EFFECTOR POSE MESSAGE ENCODING

body
└ **record** name=EndEffectorPoseRec

5.3.14 ID 4616h: ReportEndEffectorVelocityState

This message provides the receiver with the current end effector velocity state for the manipulator. The message data for this message are identical to ID 0612h:SetEndEffectorVelocityState.

TABLE 147 - REPORT END EFFECTOR VELOCITY STATE MESSAGE ENCODING

body └ **record** name=EndEffectorVelocityRec

5.3.15 ID 461Eh: ReportActiveElement

This message is used to report the identifier of the current list element being executed.

TABLE 148 - REPORT ACTIVE ELEMENT MESSAGE ENCODING

body └ **record** name=ActiveElementRec					
record name= ActiveElementRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<fixed_field>** ElementUID	unsigned short integer	One	false	UID of the active list element. A value of 0 implies that no lists are executing.

5.3.16 ID 4620h: ReportPanTiltSpecifications

This message provides the base coordinate systems, joint angle limits and joint velocity limits for the pan tilt mechanism.

TABLE 149 - REPORT PAN TILT SPECIFICATIONS MESSAGE ENCODING

body └ **record** name=ReportPanTiltSpecificationsRec					
record name= ReportPanTiltSpecificationsRec					
Field #	**Name**	**Type**	**Units**	**Optional?**	**Interpretation**
1	**<presence_vector>**	unsigned byte	one	false	
2	**<fixed_field>** PanTiltCoordinateSysX	unsigned integer	meter	true	x coordinate of origin of pan tilt coordinate system measured with respect to vehicle coordinate system Scaled integer: Lower limit = -30 m Upper limit = +30 m
3	**<fixed_field>** PanTiltCoordinateSysY	unsigned integer	meter	true	y coordinate of origin of pan tilt coordinate system measured with respect to vehicle coordinate system Scaled integer: Lower limit = -30 m Upper limit = +30 m

4	**<fixed_field>** PanTiltCoordinateSysZ	unsigned integer	meter	true	z coordinate of origin of pan tilt coordinate system measured with respect to vehicle coordinate system Scaled integer: Lower limit = -30 m Upper limit = +30 m
5	**<fixed_field>** DComponentOfUnitQuaternionQ	unsigned integer	one	true	quaternion q = d + ai +bj + ck defines the orientation of the pan tilt coordinate system measured with respect to the vehicle coordinate system Scaled integer: Lower limit = -1 Upper limit = +1
6	**<fixed_field>** AComponentOfUnitQuaternionQ	unsigned integer	one	true	see field 5
7	**<fixed_field>** BComponentOfUnitQuaternionQ	unsigned integer	one	true	see field 5
8	**<fixed_field>** CComponentOfUnitQuaternionQ	unsigned integer	one	true	see field 5
9	**< fixed_field>** Joint1Minvalue	unsigned integer	radian	false	Scaled value Lower limit = -8 π rad Upper limit = +8π rad
10	**< fixed_field>** Joint1Maxvalue	unsigned integer	radian	false	Scaled value Lower limit = -8 π rad Upper limit = +8π rad
11	**< fixed_field>** Joint1MaxSpeed	unsigned integer	radian per second	false	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second
12	**< fixed_field>** Joint2MinValue	unsigned integer	radian	false	Scaled value Lower limit = -8 π rad Upper limit = +8π rad
13	**< fixed_field>** Joint2MaxValue	unsigned integer	radian	false	Scaled value Lower limit = -8 π rad Upper limit = +8π rad
14	**< fixed_field>** Joint2MaxSpeed	unsigned integer	radian per second	false	Scaled Integer Lower limit = 0 radian per second Upper limit = +10π radian per second

5.3.17 ID 4621h: ReportPanTiltJointEffort

This message is used to provide the receiver the percent effort that is currently being applied to the two joints of the pan tilt mechanism. The message data for this message are identical to ID 0621h:SetPanTiltJointEffort.

TABLE 150 - REPORT PAN TILT JOINT EFFORT MESSAGE ENCODING

body └ **record** name=PanTiltJointEffortRec

5.3.18 ID 4622h: ReportPanTiltJointPosition

This message provides the receiver with the current values of the joint positions. The message data and mapping of the presence vector for this message are identical to ID 0622h:SetPanTiltJointPosition.

TABLE 151 - REPORT PAN TILT JOINT POSITION MESSAGE ENCODING

body └ **record** name=PanTiltJointPositionRec

5.3.19 ID 4623h: ReportPanTiltJointVelocity

This message provides the receiver with the current values of the joint velocities for a pan tilt mechanism. The message data for this message are identical to ID 0623h:SetPanTiltJointVelocity.

TABLE 152 - REPORT PAN TILT JOINT VELOCITY MESSAGE ENCODING

body └ **record** name=PanTiltJointVelocityRec

5.3.20 ID 4627h: ReportPanTiltMotionProfile

This message provides the receiver with the current motion profile for the pan tilt mechanism. The message data for this message are identical to ID 0627h:SetPanTiltMotionProfile.

TABLE 153 - REPORT PAN TILT MOTION PROFILE MESSAGE ENCODING

body └ **record** name= PanTiltMotionProfileRec

5.3.21 ID 4628h: ReportCommandedPanTiltJointPositions

This message provides the receiver with the commanded joint positions for the pan tilt mechanism. The message data for this message are identical to ID 0622h:SetPanTiltJointPosition.

TABLE 154 - REPORT COMMANDED PAN TILT JOINT POSITIONS MESSAGE ENCODING

body └ **record** name=PanTiltJointPositionRec

5.3.22 ID 4631h: ReportCommandedPanTiltJointVelocity

This message provides the receiver with the commanded joint velocities for the pan tilt mechanism. The message data for this message are identical to ID 0623h:SetPanTiltJointVelocity.

TABLE 155 - REPORT COMMANDED PAN TILT JOINT VELOCITY MESSAGE ENCODING

body
└ **record** name=PanTiltJointVelocityRec

5.3.23 ID 4632h: ReportEndEffectorSpecification

This message provides the specifications of a one degree of freedom end effector.

TABLE 156 - REPORT END EFFECTOR SPECIFICATION MESSAGE ENCODING

body
└ **record** name=ReportEndEffectorSpecificationsRec

record name= ReportEndEffectorSpecificationsRec

Field #	Name	Type	Units	Optional?	Interpretation
1	**<bit_field>** ParentID	unsigned integer	one	false	JAUS identifier on which this manipulator is mounted. Bits 0-7: Component ID, range 1...255 Bits 8-15: Node ID, range 1...255 Bits 16 – 31: Subsystem ID, range 1...65535

5.3.24 ID 4633h: ReportEndEffectorEffort

This message is used to provide the receiver the percent effort that is currently being applied to the end effector. The message data for this message are identical to ID 0633h:SetEndEffectorEffort.

TABLE 157 - REPORT END EFFECTOR EFFORT MESSAGE ENCODING

body
└ **record** name=EndEffectorEffortRec

6. NOTES

6.1 A change bar (l) located in the left margin is for the convenience of the user in locating areas where technical revisions, not editorial changes, have been made to the previous issue of this document. An (R) symbol to the left of the document title indicates a complete revision of the document, including technical revisions. Change bars and (R) are not used in original publications, nor in documents that contain editorial changes only.

APPENDIX A - XML FOR SERVICE DEFINITIONS

A.1 MANIPULATOR SPECIFICATION SERVICE

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorSpecificationService"
  id="urn:jaus:jss:manipulator:ManipulatorSpecificationService" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    This service is used to describe a manipulator arm.  When queried, the service will
    reply with a description of the manipulator's specification parameters, axes range
    of motion, and axes velocity limits.  The notations used to describe these data are
    documented in many popular text books on robotics and were previously presented in
    Section 3.

    The mechanism specification parameters as reported by the Report Manipulator
    Specifications Message consist of the number of joints, the type of each joint
    (either revolute or prismatic), the link description parameters for each link (link
    length and twist angle as shown in Figure 2), the constant joint parameter value
    (offset for a revolute joint (see Figure 3), and joint angle for a prismatic joint
    (see Figure 4)).  The minimum and maximum allowable value for each joint and the
    maximum velocity for each joint follow this information.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="events" id="urn:jaus:jss:core:Events" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref   name="manipulator"  id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"
    />
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryManipulatorSpecifications"
        declared_type_ref="manipulator.queryClass.QueryManipulatorSpecifications"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportManipulatorSpecifications"
        declared_type_ref="manipulator.informClass.ReportManipulatorSpecifications"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start                                  state_machine_name="events.transport.ReceiveFSM"
state_name="Receiving.Ready"/>
    <state_machine name="events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready">
          <default_state>
            <transition name="events.transport.Receive">
              <parameter type="QueryManipulatorSpecifications" value="msg"
                interpretation="enveloped query manipulator specifications message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="events.transport.Send"
                interpretation="Send a Report Manipulator Specs message">
                <argument value=" 'ReportManipulatorSpecifications' "/>
                <argument value="transportData"/>
```

```
            </action>
          </transition>
        </default_state>
      </state>
    </state>
  </state_machine>
 </protocol_behavior>
</service_def>
```

A.2 PRIMITIVE MANIPULATOR

```
<?xml version="1.0" encoding="UTF-8"?>
<service_def name="PrimitiveManipulator"
  id="urn:jaus:jss:manipulator:PrimitiveManipulator" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    This service is the low level interface to a manipulator arm.  Motion of the arm is
    accomplished via the Set Joint Effort message.  In this message, each actuator is
    commanded to move with a percentage of maximum effort.

    To ensure backward compatibility with 1.0 implementations of this service, it is
    recommended that this service be co-located on the same component as a Manipulator
    Specification Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryJointEffort"
        declared_type_ref="manipulator.queryClass.QueryJointEffort"/>
      <declared_message_def name="SetJointEffort"
        declared_type_ref="manipulator.commandClass.SetJointEffort"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportJointEffort"
        declared_type_ref="manipulator.informClass.ReportJointEffort"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryJointEffort" value="msg"
                  interpretation="enveloped query joint efforts message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
```

```
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a report joint efforts message">
                  <argument value=" 'ReportJointEffort' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetJointEffort" value="msg"
                    interpretation="enveloped set joint efforts message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard

condition="management.accessControl.isControllingClient(transportData)"
                    interpretation="True if the message that triggered the transition is
                    received from the client that is in control of this service"/>
                  <simple/>
                  <action name="setJointEffort"
                    interpretation="Set the joint motion efforts for the manipulator.
The manipulator joints move accordingly">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryJointEffort" value="msg"
                  interpretation="enveloped query joint efforts message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a report joint efforts message">
                  <argument value=" 'ReportJointEffort' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.3 MANIPULATOR JOINT POSITION SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointPositionSensor"
  id="urn:jaus:jss:manipulator:ManipulatorJointPositionSensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Position Sensor Service is to report the values of
    manipulator joint positions when queried.  To ensure backward compatibility
    with 1.0 implementations of this service, it is recommended that this service
```

```
    be co-located on the same component as a Manipulator Specification Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="events" id="urn:jaus:jss:core:Events" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryJointPositions"
        declared_type_ref="manipulator.queryClass.QueryJointPositions"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportJointPositions"
        declared_type_ref="manipulator.informClass.ReportJointPositions"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="events.transport.ReceiveFSM"
state_name="Receiving.Ready"/>
    <state_machine name="events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready">
          <default_state>
            <transition name="events.transport.Receive">
              <parameter type="QueryJointPositions" value="msg"
                interpretation="enveloped query joint positions message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="events.transport.Send"
                interpretation="Send Report Joint Positions message to the service
                that sent the query">
                <argument value=" 'ReportJointPositions' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```


A.4 MANIPULATOR JOINT VELOCITY SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointVelocitySensor"
  id="urn:jaus:jss:manipulator:ManipulatorJointVelocitySensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Velocity Sensor Service is to report the values of
    manipulator joint velocities when queried.  To ensure backward compatibility
    with 1.0 implementations of this service, it is recommended that this service
    be co-located on the same component as a Manipulator Specification Service.
  </description>
```

```
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="events" id="urn:jaus:jss:core:Events" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryJointVelocity"
        declared_type_ref="manipulator.queryClass.QueryJointVelocity"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportJointVelocity"
        declared_type_ref="manipulator.informClass.ReportJointVelocity"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="events.transport.ReceiveFSM"
state_name="Receiving.Ready"/>
    <state_machine name="events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready">
          <default_state>
            <transition name="events.transport.Receive">
              <parameter type="QueryJointVelocity" value="msg"
                interpretation="enveloped query joint velocities message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="events.transport.Send"
                interpretation="Send Report Joint Velocity message to the service that
                sent the query">
                <argument value=" 'ReportJointVelocity' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```


A.5 MANIPULATOR JOINT FORCE/TORQUE SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointForceTorqueSensor"
  id="urn:jaus:jss:manipulator:ManipulatorJointForceTorqueSensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Force/Torque Sensor is to report the values of
    instantaneous torques (for revolute joints) and forces (for prismatic joints)
    that are applied at the individual joints of the manipulator kinematic model
    when queried.  To ensure backward compatibility with 1.0 implementations of
    this service, it is recommended that this service be co-located on the same
    component as a Manipulator Specification Service.
  </description>
```

```
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="events" id="urn:jaus:jss:core:Events" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryJointForceTorques"
        declared_type_ref="manipulator.queryClass.QueryJointForceTorques"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportJointForceTorques"
        declared_type_ref="manipulator.informClass.ReportJointForceTorques"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="events.transport.ReceiveFSM"
state_name="Receiving.Ready"/>
    <state_machine name="events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready">
          <default_state>
            <transition name="events.transport.Receive">
              <parameter type="QueryJointForceTorques" value="msg"
                interpretation="enveloped query joint force torques message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="events.transport.Send"
                interpretation="Send Report Joint Force Torques message to the service
                that sent the query">
                <argument value=" 'ReportJointForceTorques' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```


A.6 MANIPULATOR TOOL OFFSET SERVICE

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorToolOffsetService"
  id="urn:jaus:jss:manipulator:ManipulatorToolOffsetService" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Manipulator Tool Offset Service is to configure the position
    offset of any tool attached to the manipulator flange.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="accessControl" id="urn:jaus:jss:core:AccessControl"
version="1.1"
```

```
    />
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryToolOffset"
        declared_type_ref="manipulator.queryClass.QueryToolOffset"/>
      <declared_message_def name="SetToolOffset"
        declared_type_ref="manipulator.commandClass.SetToolOffset"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportToolOffset"
        declared_type_ref="manipulator.informClass.ReportToolOffset"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryToolOffset" value="msg"
                  interpretation="enveloped query tool offset message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a report tool offset message">
                  <argument value=" 'ReportToolOffset' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryToolOffset" value="msg"
                  interpretation="enveloped query tool offset message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a report tool offset message">
                  <argument value=" 'ReportToolOffset' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="SetToolOffset" value="msg"
                  interpretation="enveloped set tool offset message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
```

```
                interpretation="transport data"/>
              <guard condition="accessControl.isControllingClient(transportData)"
                interpretation="True if the message that triggered the transition is
                received from the client that is in control of this service."/>
              <simple/>
              <action name="setToolOffset"
                interpretation="Set the location of the tool tip as measured in the
                end effector coordinate system.">
                <argument value="msg"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
</protocol_behavior>
</service_def>
```

A.7 MANIPULATOR END EFFECTOR POSE SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorEndEffectorPoseSensor"
  id="urn:jaus:jss:manipulator:ManipulatorEndEffectorPoseSensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the End Effector Pose Sensor Service is to report the position and
    orientation of the tool tip with respect to the manipulator base coordinate system.
    To ensure backward compatibility with 1.0 implementations of this service, it is
    recommended that this service be co-located on the same component as a Manipulator
    Specification Service and a Manipulator Tool Offset Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="accessControl" id="urn:jaus:jss:core:AccessControl"
version="1.1"
    />
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryEndEffectorPose"
        declared_type_ref="manipulator.queryClass.QueryEndEffectorPose"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportEndEffectorPose"
        declared_type_ref="manipulator.informClass.ReportEndEffectorPose"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
```

```
          <state name="NotControlled">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryEndEffectorPose" value="msg"
                  interpretation="enveloped query end effector pose message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a report end effector pose message">
                  <argument value=" 'ReportEndEffectorPose' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryEndEffectorPose" value="msg"
                  interpretation="enveloped query end effector pose message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a report end effector pose message">
                  <argument value=" 'ReportEndEffectorPose' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.8 MANIPULATOR END EFFECTOR VELOCITY STATE SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorEndEffectorVelocityStateSensor"
  id="urn:jaus:jss:manipulator:ManipulatorEndEffectorVelocityStateSensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the End Effector Velocity State Sensor is to report the velocity
    state of the tool tip as defined by two length-three vectors, i.e., ωe and vtool,e.
    These vectors respectively represent the angular velocity of the end effector
    coordinate system and the linear velocity of the tool tip as measured with respect
    to the manipulator base coordinate system.  To ensure backward compatibility with
    1.0 implementations of this service, it is recommended that this service be
    co-located on the same component as a Manipulator Specification Service and a
    Manipulator Tool Offset Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="accessControl" id="urn:jaus:jss:core:AccessControl"
version="1.1"
    />
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
```

```
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryEndEffectorVelocityState"
        declared_type_ref="manipulator.queryClass.QueryEndEffectorVelocityState"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportEndEffectorVelocityState"
        declared_type_ref="manipulator.informClass.ReportEndEffectorVelocityState"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryEndEffectorVelocityState" value="msg"
                  interpretation="enveloped query end effector velocity state message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a report end effector velocity state message">
                  <argument value=" 'ReportEndEffectorVelocityState' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryEndEffectorVelocityState" value="msg"
                  interpretation="enveloped query end effector velocity state message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a report end effector velocity state message">
                  <argument value=" 'ReportEndEffectorVelocityState' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.9 MANIPULATOR JOINT MOTION PROFILE SERVICE

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointMotionProfile"
  id="urn:jaus:jss:manipulator:ManipulatorJointMotionProfile" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Motion Profile Service is to allow for configuration of
    the motion profile for all services co-located on this component.  The Set Motion
    Profile message is used to set maximum velocity and acceleration rates for each of
    the joint parameters.  All motions utilize the motion profile data that was most
    recently sent.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from        name="accessControl"        id="urn:jaus:jss:core:AccessControl"
version="1.1"
    />
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryJointMotionProfile"
        declared_type_ref="manipulator.queryClass.QueryJointMotionProfile"/>
      <declared_message_def name="SetJointMotionProfile"
        declared_type_ref="manipulator.commandClass.SetJointMotionProfile"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportJointMotionProfile"
        declared_type_ref="manipulator.informClass.ReportJointMotionProfile"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryJointMotionProfile" value="msg"
                  interpretation="enveloped query motion profile message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard condition="motionProfileExists()"
                  interpretation="True if a motion profile has already been received."/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a Report Motion Profile message">
                  <argument value=" 'ReportJointMotionProfile' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
```

```
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="QueryJointMotionProfile" value="msg"
                  interpretation="enveloped query motion profile message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard condition="motionProfileExists()"
                  interpretation="True if a motion profile has already been received."/>
                <simple/>
                <action name="accessControl.events.transport.Send"
                  interpretation="Send a Report Motion Profile message">
                  <argument value=" 'ReportJointMotionProfile' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
              <transition name="accessControl.events.transport.Receive">
                <parameter type="SetJointMotionProfile" value="msg"
                  interpretation="enveloped query motion profile message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard condition="accessControl.isControllingClient(transportData)"
                  interpretation="True if the message that triggered the transition is
                  received from the client that is in control of this service."/>
                <simple/>
                <action name="setJointMotionProfile"
                  interpretation="Set the motion profile parameters for the
manipulator.">
                  <argument value="msg"/>
                </action>
              </transition>
            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.10 MANIPULATOR JOINT POSITION DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointPositionDriver"
  id="urn:jaus:jss:manipulator:ManipulatorJointPositionDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Position Driver is to perform closed-loop joint position
    control.  A single target is provided via the Set Joint Position message.  The
    target remains unchanged until a new Set Joint Position message is received.  To
    ensure backward compatibility with 1.0 implementations of this service, it is
    recommended that this service be co-located on the same component as a Manipulator
    Specification Service and a Manipulator Joint Motion Profile Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"
    />
```

```
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryCommandedJointPositions"
        declared_type_ref="manipulator.queryClass.QueryCommandedJointPositions"/>
      <declared_message_def name="SetJointPositions"
        declared_type_ref="manipulator.commandClass.SetJointPositions"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedJointPositions"
        declared_type_ref="manipulator.informClass.ReportCommandedJointPositions"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedJointPositions" value="msg"
                  interpretation="enveloped query commanded Joint Positions message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded Joint Positions message">
                  <argument value=" 'ReportCommandedJointPositions' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion" interpretation="Stop motion of the
manipulator."/>
                </exit>
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetJointPositions" value="msg"
                    interpretation="enveloped set joint efforts message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard

condition="management.accessControl.isControllingClient(transportData)
                    &amp;amp;&amp;amp; motionProfileExists()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                    a motion profile has already been received."/>
                  <simple/>
                  <action name="setJointEfforts"
                    interpretation="Set the joint motion efforts for the manipulator.
                    The manipulator joints move accordingly">
```

```
                  <argument value="msg"/>
                </action>
              </transition>
            </state>
          </state>
          <default_state>
            <transition name="management.accessControl.events.transport.Receive">
              <parameter type="QueryCommandedJointPositions" value="msg"
                interpretation="enveloped query commanded Joint Positions message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="management.accessControl.events.transport.Send"
                interpretation="Send a Report Commanded Joint Positions message">
                <argument value=" 'ReportCommandedJointPositions' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
 </protocol_behavior>
</service_def>
```

A.11 MANIPULATOR LIST DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorListDriver"
  id="urn:jaus:jss:manipulator:ManipulatorListDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Manipulator List Driver is to add support for executing a list
    of waypoints. It is expected that child services will inherit this service to provide
    functionality by overriding the isListValid() guard in the protocol.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="listManager" id="urn:jaus:jss:core:ListManager" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="ExecuteList"
        declared_type_ref="manipulator.commandClass.ExecuteList"/>
      <declared_message_def name="QueryActiveElement"
        declared_type_ref="manipulator.queryClass.QueryActiveElement"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportActiveElement"
        declared_type_ref="manipulator.informClass.ReportActiveElement"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start

state_machine_name="listManager.management.accessControl.events.transport.ReceiveFSM"
```

```
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine
name="listManager.management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">

            <default_state>
              <transition
                name="listManager.management.accessControl.events.transport.Receive">
                <parameter type="QueryActiveElement" value="msg"
                  interpretation="enveloped query Active element message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Report Active Element message">
                  <argument value=" 'ReportActiveElement' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>

          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion"
                    interpretation="Stop motion of the manipulator."/>
                </exit>
                <transition
                  name="listManager.management.accessControl.events.transport.Receive">
                  <parameter type="ExecuteList" value="msg"
                    interpretation="enveloped execute list message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard
                    condition="isControllingClient( transportData ) &amp;amp;&amp;amp;
                    listManager.elementExists( msg ) &amp;amp;&amp;amp; isListValid()"
                    interpretation="This condition is always FALSE.  It must be
                    overridden by derived services to allow the list to be executed."/>
                  <simple/>
                </transition>
              </state>
            </state>
            <default_state>
              <transition
                name="listManager.management.accessControl.events.transport.Receive">
                <parameter type="QueryActiveElement" value="msg"
                  interpretation="enveloped query Active element message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Report Active Element message">
                  <argument value=" 'ReportActiveElement' "/>
                  <argument value="transportData"/>
                </action>
```

```
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
</protocol_behavior>
</service_def>
```

A.12 MANIPULATOR JOINT POSITION LIST DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointPositionListDriver"
  id="urn:jaus:jss:manipulator:ManipulatorJointPositionListDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Position List Driver is to perform closed-loop joint
    position control through a sequence of targets. The sequence of targets is
    specified by one or more SetElement messages, as defined by the List Manager
    Service.  To ensure backward compatibility with 1.0 implementations of this
    service, it is recommended that this service be co-located on the same component
    as a Manipulator Specification Service and a Manipulator Joint Motion Profile
Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="manipulatorListDriver"
id="urn:jaus:jss:manipulator:ManipulatorListDriver"
      version="2.0"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"
    />
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryCommandedJointPositions"
        declared_type_ref="manipulator.queryClass.QueryCommandedJointPositions"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedJointPositions"
        declared_type_ref="manipulator.informClass.ReportCommandedJointPositions"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start

state_machine_name="manipulatorListDriver.listManager.management.accessControl.events.tra
nsport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
FSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
```

```
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
">
                <parameter type="QueryCommandedJointPositions" value="msg"
                  interpretation="enveloped query commanded Joint Positions message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard condition="targetExists()"
                  interpretation="True is a valid target position has been received."/>
                <simple/>
                <action

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded Joint Positions message">
                  <argument value=" 'ReportCommandedJointPositions' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion" interpretation="Stop motion of the
manipulator."/>
                </exit>
                <transition

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
">
                  <parameter type="ExecuteList" value="msg"
                    interpretation="enveloped execute list message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard

condition="manipulatorListDriver.listManager.management.accessControl.isControllingClient
( transportData )
                    &amp;&amp; listManager.elementExists( msg ) &amp;&amp; isListValid()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                      True if the Element UID specified in the message that triggered
                      the transition exists in the list AND True if the list contains
                      Manipulator Joint Position messages."/>
                  <simple/>
                  <action name="executeTargetList"
                    interpretation="Begin sequential execution of the target list
                    starting at the specified element.">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
">
```

```
                <parameter type="QueryCommandedJointPositions" value="msg"
                  interpretation="enveloped query commanded Joint Positions message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard condition="targetExists()"
                  interpretation="True is a valid target position has been received."/>
                <simple/>
                <action

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded Joint Positions message">
                  <argument value=" 'ReportCommandedJointPositions' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
              <transition

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
">
                <parameter type="SetElement" value="msg"
                  interpretation="enveloped set element message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard
                  condition="isControllingClient(transportData) &amp;&amp;
                  listManager.isValidElementRequest(msg) &amp;&amp;
                  isElementSupported(msg)"
                  interpretation="True if the message that triggered the transition is
                  received from the client that is in control of this service AND True
                  if the resulting list will not be invalid as defined by the List
                  Manager Service description and the receiving entity has sufficient
                  memory to store the element(s) AND True if the message that triggered
                  the transition contains payload(s) of valid serialized Set Joint
                  Positions message(s)."/>
                <simple/>
                <action name="setElement"
                  interpretation="Store the given targets(s) in the target list with
                  sequence specified by the previous and next element UIDs.  If this
                  action represents an insert or append into an existing list, the
                  service should modify the NextUID of the previous element and/or the
                  PreviousUID of the next element to reflect the updated sequence">
                  <argument value="msg"/>
                </action>
                <action

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Confirm Element Request message">
                  <argument value=" 'ConfirmElementRequest' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.13 MANIPULATOR END EFFECTOR POSE DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorEndEffectorPoseDriver"
  id="urn:jaus:jss:manipulator:ManipulatorEndEffectorPoseDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the End Effector Pose Driver is to perform closed-loop position and
    orientation control of the tool tip.  The input is the desired position and
    orientation of the end effector pose specified in the manipulator base coordinate
    system. It is assumed that the manipulator begins motion immediately after receiving
    the “Set End Effector Pose” message. To ensure backward compatibility with 1.0
    implementations of this service, it is recommended that this service be co-located
    on the same component as a Manipulator Specification Service, a Manipulator Tool
Offset
    Service, and a Manipulator Joint Motion Profile Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"
    />
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="SetEndEffectorPose"
        declared_type_ref="manipulator.commandClass.SetEndEffectorPose"/>
      <declared_message_def name="QueryCommandedEndEffectorPose"
        declared_type_ref="manipulator.queryClass.QueryCommandedEndEffectorPose"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedEndEffectorPose"
        declared_type_ref="manipulator.informClass.ReportCommandedEndEffectorPose"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedEndEffectorPose" value="msg"
                  interpretation="enveloped query commanded End Effector Pose message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded End Effector Pose message">
                  <argument value=" 'ReportCommandedEndEffectorPose' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
```

```
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion" interpretation="Stop motion of the
manipulator."/>
                </exit>
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetEndEffectorPose" value="msg"
                    interpretation="enveloped set end effector pose message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard

condition="management.accessControl.isControllingClient(transportData)
                    &amp;&amp; motionProfileExists()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                    True if a motion profile has already been received"/>
                  <simple/>
                  <action name="setEndEffectorPose"
                    interpretation="Set the desired position and orientation for the
                    manipulator end-effector">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedEndEffectorPose" value="msg"
                  interpretation="enveloped query commanded End Effector Pose message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded End Effector Pose message">
                  <argument value=" 'ReportCommandedEndEffectorPose' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```


A.14 MANIPULATOR END EFFECTOR POSE LIST DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorEndEffectorPoseListDriver"
  id="urn:jaus:jss:manipulator:ManipulatorEndEffectorPoseListDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the End Effector Pose List Driver is to perform closed-loop control
    of a sequence of positions and orientations of the tool tip specified in the
    manipulator base coordinate system. The sequence of targets is specified by one or
```

```
    more SetElement messages, as defined by the List Manager Service [AS6009]. To ensure
    backward compatibility with 1.0 implementations of this service, it is recommended
    that this service be co-located on the same component as a Manipulator Specification
    Service, a Manipulator Tool Offset Service, and a Manipulator Joint Motion Profile
    Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="manipulatorListDriver"
id="urn:jaus:jss:manipulator:ManipulatorListDriver"
      version="2.0"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"
    />
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryCommandedEndEffectorPose"
        declared_type_ref="manipulator.queryClass.QueryCommandedEndEffectorPose"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedEndEffectorPose"
        declared_type_ref="manipulator.informClass.ReportCommandedEndEffectorPose"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start

state_machine_name="manipulatorListDriver.listManager.management.accessControl.events.tra
nsport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
FSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">

            <default_state>
              <transition

name="manipulatorListDriver.listManager.management.accessControl.events.transport.Receive
">
                <parameter type="QueryCommandedEndEffectorPose" value="msg"
                  interpretation="enveloped query commanded end effector pose message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name=
"manipulatorListDriver.listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded End Effector Pose message">
                  <argument value=" 'ReportCommandedEndEffectorPose' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
```

```
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion"
                     interpretation="Stop motion of the manipulator."/>
                </exit>
                <transition name=
"manipulatorListDriver.listManager.management.accessControl.events.transport.Receive">
                  <parameter type="ExecuteList" value="msg"
                    interpretation="enveloped execute list message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard condition=
"manipulatorListDriver.listManager.management.accessControl.isControllingClient(transport
Data)
                    &amp;&amp; listManager.elementExists(msg) &amp;&amp; isListValid()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                      True if the Element UID specified in the message that triggered
                      the transition exists in the list AND True if the list contains
                      End Effector Pose messages."/>
                  <simple/>
                  <action name="executeTargetList"
                    interpretation="Begin sequential execution of the target list
starting at the specified element.">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition name=
"manipulatorListDriver.listManager.management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedEndEffectorPose" value="msg"
                  interpretation="enveloped query commanded end effector pose message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name=
"manipulatorListDriver.listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded End Effector Pose message">
                  <argument value=" 'ReportCommandedEndEffectorPose' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
              <transition name=
"manipulatorListDriver.listManager.management.accessControl.events.transport.Receive">
                <parameter type="SetElement" value="msg"
                  interpretation="enveloped set element message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard condition=
"manipulatorListDriver.listManager.management.accessControl.isControllingClient(transport
Data) &amp;&amp; listManager.isValidElementRequest(msg) &amp;&amp;
isElementSupported(msg)"
                  interpretation="True if the message that triggered the transition is
                  received from the client that is in control of this service AND True
                  if the resulting list will not be invalid as defined by the List
                  Manager Service description and the receiving entity has sufficient
```

```
                  memory to store the element(s) AND True if the message that triggered
                  the transition contains payload(s) of valid serialized Set End Effector
                  Pose message(s)."/>
                <simple/>
                <action name="setElement"
                  interpretation="Store the given targets(s) in the target list with
                  sequence specified by the previous and next element UIDs.  If this
                  action represents an insert or append into an existing list, the
service should modify the NextUID of the previous element and/or the PreviousUID
                  of the next element to reflect the updated sequence">
                  <argument value="msg"/>
                </action>
                <action name=
"manipulatorListDriver.listManager.management.accessControl.events.transport.Send"
                  interpretation="Send a Confirm Element Request message">
                  <argument value=" 'ConfirmElementRequest' "/>
                  <argument value="transportData"/>
                </action>
              </transition>

            </default_state>
          </state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.15 MANIPULATOR JOINT VELOCITY DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorJointVelocityDriver"
  id="urn:jaus:jss:manipulator:ManipulatorJointVelocityDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Joint Velocity Driver is to perform closed-loop joint velocity
    control.  The input is the desired instantaneous joint velocities.  It is assumed
    that the manipulator begins motion immediately after receiving the “SET JOINT
    VELOCITY” message. To ensure backward compatibility with 1.0 implementations of
    this service, it is recommended that this service be co-located on the same
    component as a Manipulator Specification Service and a Manipulator Joint Motion
    Profile Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="SetJointVelocity"
        declared_type_ref="manipulator.commandClass.SetJointVelocity"/>
      <declared_message_def name="QueryCommandedJointVelocity"
        declared_type_ref="manipulator.queryClass.QueryCommandedJointVelocity"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedJointVelocity"
        declared_type_ref="manipulator.informClass.ReportCommandedJointVelocity"/>
    </output_set>
```

```
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedJointVelocity" value="msg"
                  interpretation="enveloped query commanded joint velocities message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded joint velocities message">
                  <argument value=" 'ReportCommandedJointVelocity' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion" interpretation="Stop motion of the
manipulator."/>
                </exit>
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetJointVelocity" value="msg"
                    interpretation="enveloped set joint velocities message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard
                   condition="management.accessControl.isControllingClient(transportData)
                    &amp;&amp; motionProfileExists()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                    True if a motion profile has already been received"/>
                  <simple/>
                  <action name="setJointVelocity"
                    interpretation="Set the desired velocities for the individual joints
                    of the manipulator">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedJointVelocity" value="msg"
                  interpretation="enveloped query commanded joint velocities message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
```

```
                interpretation="Send a Report Commanded joint velocities message">
                <argument value=" 'ReportCommandedJointVelocity' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
</protocol_behavior>
</service_def>
```

A.16 MANIPULATOR END EFFECTOR VELOCITY STATE DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorEndEffectorVelocityStateDriver"
  id="urn:jaus:jss:manipulator:ManipulatorEndEffectorVelocityStateDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the End Effector Velocity State Driver is to perform closed-loop
    velocity control of the tool tip.  The velocity state of the tool tip is defined
    by two length-three vectors, i.e., ωe and vtool,e.  These vectors respectively
    represent the angular velocity of the end effector coordinate system and the linear
    velocity of the tool tip as measured with respect to the manipulator base coordinate
    system.  It is assumed that the manipulator begins motion immediately after receiving
    the “Set End Effector Velocity State” message. To ensure backward compatibility with
    1.0 implementations of this service, it is recommended that this service be co-
located
    on the same component as a Manipulator Specification Service, a Manipulator Tool
Offset
    Service, and a Manipulator Joint Motion Profile Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="SetEndEffectorVelocityState"
        declared_type_ref="manipulator.commandClass.SetEndEffectorVelocityState"/>
      <declared_message_def name="QueryCommandedEndEffectorVelocityState"
      declared_type_ref="manipulator.queryClass.QueryCommandedEndEffectorVelocityState"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedEndEffectorVelocityState"
declared_type_ref="manipulator.informClass.ReportCommandedEndEffectorVelocityState"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
```

```
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedEndEffectorVelocityState" value="msg"
                  interpretation="enveloped query commanded End effector velocity
                  state message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded End effector velocity
                  state message">
                  <argument value=" 'ReportCommandedEndEffectorVelocityState' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion" interpretation="Stop motion of the
manipulator."/>
                </exit>
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetEndEffectorVelocityState" value="msg"
                    interpretation="enveloped set End effector velocity state message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard
                   condition="management.accessControl.isControllingClient(transportData)
                    &amp;&amp; motionProfileExists()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                    True if a motion profile has already been received"/>
                  <simple/>
                  <action name="setEndEffectorVelocityState"
                    interpretation="Set the desired velocity state for the end-effector">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedEndEffectorVelocityState" value="msg"
                  interpretation="enveloped query commanded End effector velocity
                  state message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded End effector velocity
                  state message">
                  <argument value=" 'ReportCommandedEndEffectorVelocityState' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
```

```
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.17 MANIPULATOR ACTUATOR FORCE/TORQUE DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="ManipulatorActuatorForceTorqueDriver"
  id="urn:jaus:jss:manipulator:ManipulatorActuatorForceTorqueDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Actuator Force/Torque Driver is to perform closed-loop force
    control (for a prismatic actuator) and closed-loop torque control (for a revolute
    actuator). To ensure backward compatibility with 1.0 implementations of this service,
    it is recommended that this service be co-located on the same component as a
    Manipulator Specification Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryCommandedActuatorForceTorques"
        declared_type_ref="manipulator.queryClass.QueryCommandedActuatorForceTorques"/>
      <declared_message_def name="SetActuatorForceTorques"
        declared_type_ref="manipulator.commandClass.SetActuatorForceTorques"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedActuatorForceTorques"
        declared_type_ref="manipulator.informClass.ReportCommandedActuatorForceTorques"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedActuatorForceTorques" value="msg"
                  interpretation="enveloped Query Commanded Actuator Force Torque
message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded Actuator Force Torque message">
                  <argument value=" 'ReportCommandedActuatorForceTorques' "/>
                  <argument value="transportData"/>
```

```
              </action>
            </transition>
          </default_state>
        </state>
        <state name="Controlled" interpretation="redefine state in order to extend">
          <state name="Available" initial_state="Standby">
            <state name="Ready">
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="SetActuatorForceTorques" value="msg"
                  interpretation="enveloped set Actuator Force Torque message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard
                condition="management.accessControl.isControllingClient(transportData)"
                  interpretation="True if the message that triggered the transition is
                  received from the client that is in control of this service"/>
                <simple/>
                <action name="setActuatorForceTorques"
                  interpretation="Set the desired actuator forces and torques">
                  <argument value="msg"/>
                </action>
              </transition>
            </state>
          </state>
          <default_state>
            <transition name="management.accessControl.events.transport.Receive">
              <parameter type="QueryCommandedActuatorForceTorques" value="msg"
                interpretation="enveloped Query Commanded Actuator Force Torque
message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="management.accessControl.events.transport.Send"
                interpretation="Send a Report Commanded Actuator Force Torque message">
                <argument value=" 'ReportCommandedActuatorForceTorques' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
</protocol_behavior>
</service_def>
```

A.18 PRIMITIVE PAN TILT

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="PrimitivePanTilt" id="urn:jaus:jss:manipulator:PrimitivePanTilt"
  version="2.0" xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
  The Primitive Pan Tilt Service is the low level interface to a pan tilt mechanism.
  Motion of the pan tilt mechanism is accomplished via the Set Pan Tilt Joint Effort
  message.  In this message, each actuator is commanded to move with a percentage of
  maximum effort.  To ensure backward compatibility with 1.0 implementations of this
  service, it is recommended that this service be co-located on the same component
  as a Pan Tilt Specification Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
```

```
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryPanTiltJointEffort"
        declared_type_ref="manipulator.queryClass.QueryPanTiltJointEffort"/>
      <declared_message_def name="SetPanTiltJointEffort"
        declared_type_ref="manipulator.commandClass.SetPanTiltJointEffort"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportPanTiltJointEffort"
        declared_type_ref="manipulator.informClass.ReportPanTiltJointEffort"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryPanTiltJointEffort" value="msg"
                  interpretation="enveloped query Pan Tilt joint efforts message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a report Pan Tilt joint efforts message">
                  <argument value=" 'ReportPanTiltJointEffort' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetPanTiltJointEffort" value="msg"
                    interpretation="enveloped set Pan Tilt joint efforts message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard
                  condition="management.accessControl.isControllingClient(transportData)"
                    interpretation="True if the message that triggered the transition is
                    received from the client that is in control of this service"/>
                  <simple/>
                  <action name="setPanTiltJointEffort"
                    interpretation="Set the joint motion efforts for the two joints of
the pan tilt mechanism">
                    <argument value="msg"/>
                  </action>
                </transition>
```

```
            </state>
          </state>
          <default_state>
            <transition name="management.accessControl.events.transport.Receive">
              <parameter type="QueryPanTiltJointEffort" value="msg"
                interpretation="enveloped query Pan Tilt joint efforts message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="management.accessControl.events.transport.Send"
                interpretation="Send a report Pan Tilt joint efforts message">
                <argument value=" 'ReportPanTiltJointEffort' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
</protocol_behavior>
</service_def>
```

A.19 PAN TILT JOINT POSITION SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="PanTiltJointPositionSensor"
  id="urn:jaus:jss:manipulator:PanTiltJointPositionSensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Pan Tilt Joint Position Sensor Service is to report the values
    of the two joint angles of the pan tilt mechanism when queried. To ensure backward
    compatibility with 1.0 implementations of this service, it is recommended that this
    service be co-located on the same component as a Pan Tilt Specification Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="events" id="urn:jaus:jss:core:Events" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryPanTiltJointPositions"
        declared_type_ref="manipulator.queryClass.QueryPanTiltJointPositions"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportPanTiltJointPositions"
        declared_type_ref="manipulator.informClass.ReportPanTiltJointPositions"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="events.transport.ReceiveFSM"
state_name="Receiving.Ready"/>
    <state_machine name="events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready">
```

```
          <default_state>
            <transition name="events.transport.Receive">
              <parameter type="QueryPanTiltJointPositions" value="msg"
                interpretation="enveloped query Pan Tilt joint positions message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="events.transport.Send"
                interpretation="Send Report Pan Tilt Joint Positions message to the
                service that sent the query">
                <argument value=" 'ReportPanTiltJointPositions' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.20 PAN TILT JOINT VELOCITY SENSOR

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="PanTiltJointVelocitySensor"
  id="urn:jaus:jss:manipulator:PanTiltJointVelocitySensor" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Pan Tilt Joint Velocity Sensor Service is to report the values
    of the two joint velocities of the pan tilt mechanism when queried. To ensure
    backward compatibility with 1.0 implementations of this service, it is recommended
    that this service be co-located on the same component as a Pan Tilt Specification
    Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="events" id="urn:jaus:jss:core:Events" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryPanTiltJointVelocity"
        declared_type_ref="manipulator.queryClass.QueryPanTiltJointVelocity"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportPanTiltJointVelocity"
        declared_type_ref="manipulator.informClass.ReportPanTiltJointVelocity"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="events.transport.ReceiveFSM"
state_name="Receiving.Ready"/>
    <state_machine name="events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready">
          <default_state>
```

```
            <transition name="events.transport.Receive">
              <parameter type="QueryPanTiltJointVelocity" value="msg"
                interpretation="enveloped query Pan Tiltjoint velocities message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="events.transport.Send"
                interpretation="Send Report Pan Tilt Joint Velocity message to the
                service that sent the query">
                <argument value=" 'ReportPanTiltJointVelocity' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state_machine>
  </protocol_behavior>
</service_def>
```

A.21 PAN TILT JOINT POSITION DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="PanTiltJointPositionDriver"
  id="urn:jaus:jss:manipulator:PanTiltJointPositionDriver" version="2.0"
  xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of the Pan Tilt Joint Position Driver is to perform closed-loop joint
    position control.  A single target is provided via the Set Pan Tilt Joint Position
    message.  The target remains unchanged until a new Set Pan Tilt Joint Position
    message is received.  To ensure backward compatibility with 1.0 implementations of
    this service, it is recommended that this service be co-located on the same component
    as a Pan Tilt Specification Service and a Pan Tilt Motion Profile Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
      version="2.0"/>
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="QueryCommandedPanTiltJointPositions"
        declared_type_ref="manipulator.queryClass.QueryCommandedPanTiltJointPositions"/>
      <declared_message_def name="SetPanTiltJointPositions"
        declared_type_ref="manipulator.commandClass.SetPanTiltJointPositions"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedPanTiltJointPositions"

declared_type_ref="manipulator.informClass.ReportCommandedPanTiltJointPositions"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
```

```
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">

            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedPanTiltJointPositions" value="msg"
                  interpretation="enveloped query commanded Pan Tilt Joint Positions
message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded Pan Tilt Joint Positions
message">
                  <argument value=" 'ReportCommandedPanTiltJointPositions' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </default_state>
          </state>
          <state name="Controlled" interpretation="redefine state in order to extend">
            <state name="Available" initial_state="Standby">
              <state name="Ready">
                <exit>
                  <action name="stopMotion"
                    interpretation="Stop motion of the pan tilt unit."/>
                </exit>
                <transition name="management.accessControl.events.transport.Receive">
                  <parameter type="SetPanTiltJointPositions" value="msg"
                    interpretation="enveloped set Pan Tilt joint efforts message"/>
                  <parameter type="Receive.Body.ReceiveRec" value="transportData"
                    interpretation="transport data"/>
                  <guard
                   condition="management.accessControl.isControllingClient(transportData)
                    &amp;&amp; panTiltMotionProfileExists()"
                    interpretation="True if the message that triggered the transition
                    is received from the client that is in control of this service AND
                    True if a Pan Tilt motion profile has already been received."/>
                  <simple/>
                  <action name="setPanTiltJointEfforts"
                    interpretation="Set the desired joint values for the pan tilt
mechanism">
                    <argument value="msg"/>
                  </action>
                </transition>
              </state>
            </state>
            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedPanTiltJointPositions" value="msg"
                  interpretation="enveloped query commanded Pan Tilt Joint Positions
message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <simple/>
                <action name="management.accessControl.events.transport.Send"
                  interpretation="Send a Report Commanded Pan Tilt Joint Positions
message">
                  <argument value=" 'ReportCommandedPanTiltJointPositions' "/>
                  <argument value="transportData"/>
```

```
              </action>
            </transition>
          </default_state>
        </state>
      </state>
    </state>
  </state_machine>
 </protocol_behavior>
</service_def>
```

A.22 PAN TILT JOINT VELOCITY DRIVER

```
<?xml version="1.1" encoding="UTF-8"?>
<service_def name="PanTiltJointVelocityDriver"
  id="urn:jaus:jss:manipulator:PanTiltJointVelocityDriver" version="2.0"
xmlns="urn:jaus:jsidl:1.1">
  <description xml:space="preserve">
    The function of The Pan Tilt Joint Velocity Driver is to perform closed-loop joint
    velocity control.  The input is the desired instantaneous desired joint velocities
    for the pan tilt mechanism.  It is assumed that the pan tilt mechanism begins motion
    immediately after receiving the Set Pan Tilt Joint Velocity message. To ensure
    backward compatibility with 1.0 implementations of this service, it is recommended
    that this service be co-located on the same component as a Pan Tilt Specification
    Service and a Pan Tilt Motion Profile Service.
  </description>
  <assumptions> Messages may be delayed, lost, or reordered. </assumptions>
  <references>
    <inherits_from name="management" id="urn:jaus:jss:core:Management" version="1.1"/>
  </references>
  <declared_type_set name="types">
    <declared_type_set_ref name="manipulator" id="urn:jaus:jss:manipulator:MessageSet"
version="2.0"
    />
  </declared_type_set>
  <message_set>
    <input_set>
      <declared_message_def name="SetPanTiltJointVelocity"
        declared_type_ref="manipulator.commandClass.SetPanTiltJointVelocity"/>
      <declared_message_def name="QueryCommandedPanTiltJointVelocity"
        declared_type_ref="manipulator.queryClass.QueryCommandedPanTiltJointVelocity"/>
    </input_set>
    <output_set>
      <declared_message_def name="ReportCommandedPanTiltJointVelocity"
        declared_type_ref="manipulator.informClass.ReportCommandedPanTiltJointVelocity"/>
    </output_set>
  </message_set>
  <internal_events_set/>
  <protocol_behavior is_stateless="false">
    <start state_machine_name="management.accessControl.events.transport.ReceiveFSM"
      state_name="Receiving.Ready.NotControlled"/>
    <state_machine name="management.accessControl.events.transport.ReceiveFSM"
      interpretation="extending ReceiveFSM of base service (transport)">
      <state name="Receiving" initial_state="Ready"
        interpretation="redefine state in order to extend">
        <state name="Ready" initial_state="NotControlled"
          interpretation="redefine state in order to extend">
          <state name="NotControlled" interpretation="redefine state in order to extend">

            <default_state>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="QueryCommandedPanTiltJointVelocity" value="msg"
```

```
                interpretation="enveloped query commanded Pan Tilt joint velocities
message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="management.accessControl.events.transport.Send"
                interpretation="Send a Report Commanded Pan Tilt joint velocities
message">
                <argument value=" 'ReportCommandedPanTiltJointVelocity' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>

        </state>
        <state name="Controlled" interpretation="redefine state in order to extend">
          <state name="Available" initial_state="Standby">
            <state name="Ready">
              <exit>
                <action name="stopMotion" interpretation="Stop motion of the pan tilt
unit."/>
              </exit>
              <transition name="management.accessControl.events.transport.Receive">
                <parameter type="SetPanTiltJointVelocity" value="msg"
                  interpretation="enveloped set Pan Tilt joint velocities message"/>
                <parameter type="Receive.Body.ReceiveRec" value="transportData"
                  interpretation="transport data"/>
                <guard

condition="management.accessControl.isControllingClient(transportData)
                  &amp;&amp; panTiltMotionProfileExists()"
                  interpretation="True if the message that triggered the transition
                  is received from the client that is in control of this service AND
                  True if a motion profile has already been received"/>
                <simple/>
                <action name="setPanTiltJointVelocity"
                  interpretation="Set the desired velocities for the individual joints
                  of the Pan Tilt mechanism">
                  <argument value=" 'SetPanTiltJointVelocity' "/>
                  <argument value="transportData"/>
                </action>
              </transition>
            </state>
          </state>
          <default_state>
            <transition name="management.accessControl.events.transport.Receive">
              <parameter type="QueryCommandedPanTiltJointVelocity" value="msg"
                interpretation="enveloped query commanded Pan Tilt joint velocities
message"/>
              <parameter type="Receive.Body.ReceiveRec" value="transportData"
                interpretation="transport data"/>
              <simple/>
              <action name="management.accessControl.events.transport.Send"
                interpretation="Send a Report Commanded Pan Tilt joint velocities
message">
                <argument value=" 'ReportCommandedPanTiltJointVelocity' "/>
                <argument value="transportData"/>
              </action>
            </transition>
          </default_state>
        </state>
      </state>
```