



AEROSPACE STANDARD

AS6111™

Issued

2023-09

J AUS Unmanned Maritime Vehicle Service Set

RATIONALE

This service set provides the unmanned maritime vehicle specific capabilities that are not supported by the higher level platform-independent services. This service set enables interoperability on common elements specific to unmanned surface vehicles.

INTRODUCTION

This SAE Aerospace Standard (AS) defines a message-passing interface for services commonly found in unmanned maritime vehicles (UMVs). These services provide the platform-specific information and capabilities unique to UMVs. Two common categories of UMVs are unmanned surface vehicles (USVs) and unmanned underwater vehicles (UUVs). Additional capabilities are specified in the JAUS Core Service Set (refer to AS5710) and are frequently referenced herein. This document aims to fill maritime-domain specific gaps not covered by AS6009, the JAUS Mobility Service Set that is domain- and platform-independent.

The primary goal of the JAUS UMV 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 (refer to AS5669), is used in conjunction with this specification.

Each service in the JAUS UMV Service Set is described using the JAUS Service Interface Definition Language (JSIDL; refer to AS5684). 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 document, it is highly recommended for supporting context. For convenience, the JAUS UMV Service Set contains both a text-based and XML-based representation for each service.

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 [APPENDIX A](#). In case of transcription errors, the XML specification should be considered correct.

SAE Executive Standards Committee Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2023 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

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
<http://www.sae.org>

SAE WEB ADDRESS:

Full document details: SAE AS6111

For more information on this standard, visit
<https://www.sae.org/standards/content/AS6111>

Documents

Document Sharing Hub

TABLE OF CONTENTS

1.	SCOPE.....	6
1.1	Purpose.....	6
1.2	J AUS Core Service Set.....	7
1.3	Compliance.....	7
1.4	Document Organization.....	7
2.	REFERENCES.....	7
2.1	Applicable Documents.....	7
2.1.1	SAE Publications.....	7
2.1.2	Other Publications.....	8
2.2	Acronyms.....	8
3.	SERVICE DEFINITIONS.....	9
3.1	Annunciator.....	9
3.1.1	Description.....	9
3.1.2	Assumptions.....	9
3.1.3	References.....	10
3.1.4	Vocabulary.....	10
3.1.5	Encoding.....	11
3.1.6	Protocol Behavior.....	18
3.2	BallastControl.....	20
3.2.1	Description.....	20
3.2.2	Assumptions.....	20
3.2.3	References.....	20
3.2.4	Vocabulary.....	20
3.2.5	Encoding.....	21
3.2.6	Protocol Behavior.....	28
3.3	BilgePumpControl.....	29
3.3.1	Description.....	29
3.3.2	Assumptions.....	29
3.3.3	References.....	29
3.3.4	Vocabulary.....	29
3.3.5	Encoding.....	30
3.3.6	Protocol Behavior.....	32
3.4	BucketDriver.....	32
3.4.1	Description.....	32
3.4.2	Assumptions.....	33
3.4.3	References.....	33
3.4.4	Vocabulary.....	33
3.4.5	Encoding.....	33
3.4.6	Protocol Behavior.....	35
3.5	CTDSensor.....	36
3.5.1	Description.....	36
3.5.2	Assumptions.....	36
3.5.3	References.....	36
3.5.4	Vocabulary.....	37
3.5.5	Encoding.....	37
3.5.6	Protocol Behavior.....	38
3.6	CompartmentSensor.....	39
3.6.1	Description.....	39
3.6.2	Assumptions.....	39
3.6.3	References.....	39
3.6.4	Vocabulary.....	39
3.6.5	Encoding.....	40
3.6.6	Protocol Behavior.....	41

3.7	ControlSurface	42
3.7.1	Description	42
3.7.2	Assumptions.....	42
3.7.3	References.....	42
3.7.4	Vocabulary	42
3.7.5	Encoding	43
3.7.6	Protocol Behavior.....	47
3.8	SeaStateSensor	48
3.8.1	Description	48
3.8.2	Assumptions.....	48
3.8.3	References.....	48
3.8.4	Vocabulary	48
3.8.5	Encoding	48
3.8.6	Protocol Behavior.....	49
3.9	AnchorService.....	50
3.9.1	Description	50
3.9.2	Assumptions.....	50
3.9.3	References.....	50
3.9.4	Vocabulary	50
3.9.5	Encoding	51
3.9.6	Protocol Behavior.....	55
3.10	PlatformSpecifications.....	56
3.10.1	Description	56
3.10.2	Assumptions.....	56
3.10.3	References.....	57
3.10.4	Vocabulary	57
3.10.5	Encoding	57
3.10.6	Protocol Behavior.....	62
3.11	PowerPlant.....	62
3.11.1	Description	62
3.11.2	Assumptions.....	62
3.11.3	References.....	62
3.11.4	Vocabulary	63
3.11.5	Encoding	63
3.11.6	Protocol Behavior.....	69
3.12	PropulsionService	70
3.12.1	Description	70
3.12.2	Assumptions.....	71
3.12.3	References.....	71
3.12.4	Vocabulary	71
3.12.5	Encoding	72
3.12.6	Protocol Behavior.....	81
4.	NOTES.....	82
4.1	Revision Indicator.....	82
APPENDIX A	XML FOR SERVICE DEFINITIONS	83
Figure 1	Annunciator service protocol behavior.....	18
Figure 2	BallastControl service protocol behavior.....	28
Figure 3	BilgePumpControl service protocol behavior.....	32
Figure 4	BucketDriver service protocol behavior	35
Figure 5	CTDSensor service protocol behavior	38
Figure 6	CompartmentSensor service protocol behavior.....	41
Figure 7	ControlSurface service protocol behavior	47
Figure 8	SeaStateSensor service protocol behavior.....	49
Figure 9	AnchorService protocol behavior.....	55
Figure 10	Vehicle dimension measurements	58
Figure 11	PlatformSpecifications service protocol behavior	62
Figure 12	PowerPlant service protocol behavior.....	69



Figure 13	Propulsor offset example	71
Figure 14	Propulsion service protocol behavior	81
Table 1	Annunciator service message vocabulary	10
Table 2	SetLight message encoding	11
Table 3	SetWhistle message encoding	12
Table 4	SetBell message encoding	12
Table 5	SetGong message encoding	13
Table 6	QueryAnnunciatorCapabilities message encoding	13
Table 7	QueryLightStatus message encoding	13
Table 8	QueryWhistle message encoding	14
Table 9	QueryBell message encoding	14
Table 10	QueryGong message encoding	14
Table 11	ReportAnnunciatorCapabilities message encoding	15
Table 12	ReportLightStatus message encoding	16
Table 13	ReportWhistle message encoding	17
Table 14	ReportBell message encoding	17
Table 15	ReportGong message encoding	18
Table 16	Annunciator service state transitions	19
Table 17	Annunciator service conditions	19
Table 18	Annunciator service transition actions	19
Table 19	BallastControl service message vocabulary	20
Table 20	BallastPumpControl message encoding	21
Table 21	BallastTankControl message encoding	22
Table 22	QueryBallastTankSpecifications message encoding	22
Table 23	QueryBallastPumpStatus message encoding	23
Table 24	QueryBallastTankStatus message encoding	23
Table 25	ReportBallastTankSpecifications message encoding	24
Table 26	ReportBallastPumpStatus message encoding	26
Table 27	ReportBallastTankStatus message encoding	27
Table 28	BallastControl service state transitions	28
Table 29	BallastControl service conditions	28
Table 30	BallastControl service transition actions	29
Table 31	BilgePumpControl service message vocabulary	29
Table 32	SetBilgePumpState message encoding	30
Table 33	QueryBilgePumpIdentification message encoding	30
Table 34	QueryBilgePumpStatus message encoding	30
Table 35	ReportBilgePumpIdentification message encoding	31
Table 36	ReportBilgePumpStatus message encoding	31
Table 37	BilgePumpControl service state transitions	32
Table 38	BilgePumpControl service conditions	32
Table 39	BilgePumpControl service transition actions	32
Table 40	BucketDriver service message vocabulary	33
Table 41	SetBucketPosition message encoding	33
Table 42	QueryBucketIdentification message encoding	34
Table 43	QueryBucketPosition message encoding	34
Table 44	ReportBucketIdentification message encoding	34
Table 45	ReportBucketPosition message encoding	35
Table 46	BucketDriver service state transitions	36
Table 47	BucketDriver service conditions	36
Table 48	BucketDriver service transition actions	36
Table 49	CTDSensor service message vocabulary	37
Table 50	QueryCTDState message encoding	37
Table 51	ReportCTDState message encoding	38
Table 52	CTDSensor service state transitions	38
Table 53	CTDSensor service transition actions	39
Table 54	CompartmentSensor service message vocabulary	39
Table 55	QueryCompartmentIdentification message encoding	40
Table 56	QueryCompartmentStatus message encoding	40
Table 57	ReportCompartmentIdentification message encoding	40

Table 58	ReportCompartmentStatus message encoding	41
Table 59	CompartmentSensor service state transitions	41
Table 60	CompartmentSensor service transition actions	42
Table 61	ControlSurface service message vocabulary	42
Table 62	SetControlSurface message encoding	43
Table 63	QueryControlSurfaceSpecifications message encoding	43
Table 64	QueryControlSurface message encoding	44
Table 65	ReportControlSurfaceSpecifications message encoding	44
Table 66	ReportControlSurface message encoding	46
Table 67	ControlSurface service state transitions	47
Table 68	ControlSurface service conditions	47
Table 69	ControlSurface service transition actions	47
Table 70	SeaStateSensor service message vocabulary	48
Table 71	QuerySeaState message encoding	48
Table 72	ReportSeaState message encoding	49
Table 73	SeaStateSensor service state transitions	49
Table 74	SeaStateSensor service transition actions	49
Table 75	AnchorService message vocabulary	50
Table 76	SetAnchorConfiguration message encoding	51
Table 77	QueryAnchorSpecifications message encoding	51
Table 78	QueryAnchorStatus message encoding	52
Table 79	QueryAnchorConfiguration message encoding	52
Table 80	ReportAnchorSpecifications message encoding	53
Table 81	ReportAnchorStatus message encoding	54
Table 82	ReportAnchorConfiguration message encoding	55
Table 83	AnchorService state transitions	56
Table 84	AnchorService conditions	56
Table 85	AnchorService transition actions	56
Table 86	PlatformSpecifications service message vocabulary	57
Table 87	QueryPlatformSpecifications message encoding	57
Table 88	ReportPlatformSpecifications message encoding	59
Table 89	PlatformSpecifications service state transitions	62
Table 90	PlatformSpecifications service transition actions	62
Table 91	PowerPlant service message vocabulary	63
Table 92	SetPowerPlantConfiguration message encoding	63
Table 93	QueryPowerPlantSpecifications message encoding	63
Table 94	QueryPowerPlantConfiguration message encoding	64
Table 95	QueryPowerPlantStatus message encoding	64
Table 96	ReportPowerPlantSpecifications message encoding	65
Table 97	ReportPowerPlantConfiguration message encoding	67
Table 98	ReportPowerPlantStatus message encoding	68
Table 99	PowerPlant service state transitions	70
Table 100	PowerPlant service conditions	70
Table 101	PowerPlant service transition actions	70
Table 102	Propulsion service message vocabulary	71
Table 103	SetPropulsorConfiguration message encoding	72
Table 104	QueryPropulsorSpecifications message encoding	73
Table 105	QueryPropulsorStatus message encoding	73
Table 106	QueryPropulsorConfiguration message encoding	74
Table 107	ReportPropulsorSpecifications message encoding	74
Table 108	ReportPropulsorStatus message encoding	78
Table 109	ReportPropulsorConfiguration message encoding	80
Table 110	Propulsion service state transitions	81
Table 111	Propulsion service conditions	82
Table 112	Propulsion service transition actions	82

1. SCOPE

This SAE Aerospace Standard (AS) defines a set of standard application layer interfaces called JAUS UMV 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 UMV Services represent the platform-specific capabilities commonly found in UMVs and augment the Mobility Service Set (refer to AS6009) which is platform-agnostic. At present, 12 services are defined in this document. While these services are presented alphabetically within this document, they also can be logically categorized as:

- Platform Description Services: This service provides information about the vehicle platform, including mobility limits and geometric properties:
 - Platform Specification Service.
- Propulsion and Driver Services: These services provide the control and monitoring capabilities to the vehicle's propulsion system(s). These services can be applied selectively to facilitate primitive mobility teleoperation.
 - Propulsion Service,
 - Bucket Driver Service, and
 - Control Surface Service.
- Auxiliary Services: These services provide an interface to auxiliary sensors and devices, including depth sensors, annunciating devices, components such as generators, bilge pumps, ballast pumps, etc.
 - Annunciator Service,
 - CTD Sensor Service,
 - Compartment Sensor Service,
 - Sea State Sensor Service,
 - Bilge Pump Control Service,
 - Ballast Tank Control Service,
 - Power Plant Service, and
 - Anchor Service.

NOTE: There are two aspects of the maritime domain that are candidates for a future revision of this specification. The first is the incorporation of a Sonar Service that would set up the identification, location, and capabilities of sonars installed on the UMV along with high-level control and feedback services for them, but no standardization of the sonar detection data. If there is a community appetite for standardizing the detection data, this would likely be incorporated as a separate service in a future Environmental Sensing revision. The other aspect is adding additional granularity to the association of UMV componentry to UMV compartments with the aim of providing additional diagnostic/troubleshooting insights to other services and operators. Community feedback on these two topics is encouraged.

1.1 Purpose

The purpose of this document is to facilitate interoperation of unmanned vehicle systems, subsystems, command and control systems, and payloads by standardization of the message sets and associated protocols.

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 (refer to 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 UMV 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 document; it is not necessary that a single entity realize every Service to be considered compliant.

1.4 Document Organization

The layout of this document is as follows: Section [2](#) specifies external references and provides a list of acronyms. Section [3](#) specifies the JAUS Service Definition for each of the services, with particular emphasis on the description, assumptions, message definition, and protocol behavior. Finally, [APPENDIX A](#) contains 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 latest issue of SAE publications shall apply. The applicable issue of other publications shall be the issue in effect on the date of the purchase order. 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 unless a specific exemption has been obtained.

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 +1 724-776-4970 (outside USA), www.sae.org.

AIR5665	Architecture Framework for Unmanned Systems
AS5669	JAUS/SDP Transport Specification
AS5684	JAUS Service Interface Definition Language
AS5710	JAUS Core Service Set
AS6009	JAUS Mobility Service Set

2.1.2 Other Publications

Wikipedia. (2022). *Douglas sea scale*. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Douglas_sea_scale.

COLREGs. (n.d., a). *Rule 34 (Manoeuvring and warning signals)*. ACTS Project Consortium. Retrieved from https://www.ecolregs.com/index.php?option=com_k2&view=item&layout=item&id=66&Itemid=406&lang=en.

COLREGs. (n.d., b.). *Rule 35 (Sound signals in restricted visibility)*. ACTS Project Consortium. Retrieved from https://www.ecolregs.com/index.php?option=com_k2&view=item&layout=item&id=67&Itemid=407&lang=en.

2.2 Acronyms

BWL	Beam at waterline
CB	Center of buoyancy
CG	Center of gravity
CMD	Command (as in commanded fin deflection)
COLREGs	International Regulations for Preventing Collisions at Sea 1972, published by International Maritime Organization (IMO)
CTD	Conductivity, temperature, and depth (used to compute speed of sound in water)
FB	Feedback value (as in measured fin deflection)
ID	Identifier
JAUS	Joint Architecture for Unmanned Systems
JSIDL	JAUS Service (Interface) Definition Language
LWL	Length at waterline
POSIX	Portable Operating System Interface
RPM	Revolutions per minute
UMV	Unmanned maritime vehicle
USV	Unmanned surface vehicle
UUV	Unmanned underwater vehicle
WMO	World Meteorological Organization
XML	Extensible Markup Language

3. SERVICE DEFINITIONS

The following subsections provide a textual definition for each Service Definition in the JAUS UMV Service Set. Corresponding JSIDL definitions are offered in [APPENDIX A](#). Additional information on interpreting the service definition elements may be found in AS5684.

3.1 Annunciator

name=Annunciator

version=1.0

id=urn:jaus:jss:umv:Annunciator

3.1.1 Description

The function of this service is to control and monitor the vessel's sound and light signaling devices. This service supports the independent control of sound and light signaling devices, allowing clients to implement patterns such as those required by the International Regulations for Preventing Collisions at Sea 1972 (COLREGs), published by International Maritime Organization (IMO). COLREGs defines six common types of lights:

- Masthead light: A frontal unbroken white light placed over the fore and aft centerline of the vessel.
- Side lights: Unbroken green light(s) on the starboard (right) side and unbroken red light(s) on the port (left) side.
- Stern light: A posterior unbroken white light placed as nearly as practicable at the stern.
- Towing light: A yellow light having the same characteristics as the stern light.
- All-round light: an unbroken light over an arc of horizon of 360 degrees.
- Flashing light: A light flashing at a frequency of 120 flashes or more per minute.

COLREGs also defines three types of sounds:

- Whistle: On vessels over 12 meters (39.4 feet) in length.
- Bell: On vessels over 12 meters (39.4 feet) in length.
- Gong: On vessels over 100 meters (328 feet) in length.

This service provides means to individually control and monitor each type of sounds and lights. For example, the command message SetLight can power on or off the masthead light, whereas QueryLightStatus and ReportLightStatus provides the typical query and report of each light's on off status. This straightforward control mechanism can quickly change each sound or light device's functional status. Note that this service does not implement COLREG compliant signals but allows a client to do so through independent control of the sound and light devices. A client should first send a QueryAnnunciatorCapabilities to receive its response ReportAnnunciatorCapabilities in order to determine what sounds and lights are supported by the platform.

3.1.2 Assumptions

Messages may be delayed, lost, or reordered.

3.1.3 References

Inherits-from=AccessControl

version=1.1

id=urn:jaus:jss:core:AccessControl

3.1.4 Vocabulary

Table 1 - Annunciator service message vocabulary

Message ID (hex)	Name	Command
Input Set		
0A11	SetLight (see 3.1.5.1.1)	true
0A12	SetWhistle (see 0)	true
0A13	SetBell (see 3.1.5.1.3)	true
0A14	SetGong (see 0)	true
2A10	QueryAnnunciatorCapabilities (see 3.1.5.1.5)	false
2A11	QueryLightStatus (see 3.1.5.1.6)	false
2A12	QueryWhistle (see 0)	false
2A13	QueryBell (see 3.1.5.1.8)	false
2A14	QueryGong (see 3.1.5.1.9)	false
Output Set		
4A10	ReportAnnunciatorCapabilities (see 3.1.5.2.1)	false
4A11	ReportLightStatus (see 0)	false
4A12	ReportWhistle (see 0)	false
4A13	ReportBell (see 3.1.5.2.4)	false
4A14	ReportGong (see 0)	false

3.1.5 Encoding

3.1.5.1 Input Set

3.1.5.1.1 ID 0A11: SetLight

This message is used to switch the lights on or off.

Table 2 - SetLight message encoding

body └ record name = LightSwitchRec					
Record Name = LightSwitchRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<bit_field> Switch	unsigned byte	bit_field	false	Bits to turn lights on or off Bits 0..0, MastheadLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 1..1, PortSideLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 2..2, StarboardSideLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 3..3, SternLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 4..4, TowingLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 5..5, AllAroundLight: Value set, offset=false, ranges/enums: 1= ON 0= OFF Bits 6..6, FlashingLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 7..7, UserDefinedLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> FlashDuration	float	second	true	Flash duration in seconds, when flashing light is activated
4	<fixed_field> FlashNumber	unsigned byte	one	true	1 to 255: The number of times to flash 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	float	second	true	The time in seconds between flashes

3.1.5.1.2 ID 0A12: SetWhistle

This message is used to turn on or off the whistle sound. The message can command a blast for a number of times with a specified duration and time period in between each blast. When the number of times to blast is set to 0, the component shall continue to generate a blast after each time period specified.

Table 3 - SetWhistle message encoding

body └ record name = SoundControlRec					
Record Name = SoundControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Switch	unsigned byte	one	false	Command enumerations Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> BlastDuration	unsigned byte	second	true	Blast duration in seconds
4	<fixed_field> BlastNumber	unsigned byte	one	true	1 to 255: The number of times to blast 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	unsigned byte	second	true	The time in seconds between blasts

3.1.5.1.3 ID 0A13: SetBell

This message is used to turn on or off the bell sound. The message can command a blast for a number of times with a specified duration and time period in between each blast. When the number of times to blast is set to 0, the component shall continue to generate a blast after each time period specified.

Table 4 - SetBell message encoding

body └ record name = SoundControlRec					
Record Name = SoundControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Switch	unsigned byte	one	false	Command enumerations Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> BlastDuration	unsigned byte	second	true	Blast duration in seconds
4	<fixed_field> BlastNumber	unsigned byte	one	true	1 to 255: The number of times to blast 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	unsigned byte	second	true	The time in seconds between blasts

3.1.5.1.4 ID 0A14: SetGong

This message is used to turn on or off the gong sound. The message can command a blast for a number of times with a specified duration and time period in between each blast. When the number of times to blast is set to 0, the component shall continue to generate a blast after each time period specified.

Table 5 - SetGong message encoding

body └ record name = SoundControlRec					
Record Name = SoundControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Switch	unsigned byte	one	false	Command enumerations Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> BlastDuration	unsigned byte	second	true	Blast duration in seconds
4	<fixed_field> BlastNumber	unsigned byte	one	true	1 to 255: The number of times to blast 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	unsigned byte	second	true	The time in seconds between blasts

3.1.5.1.5 ID 2A10: QueryAnnunciatorCapabilities

This message shall cause the receiving component to reply to the requestor with a ID 4A10h: ReportAnnunciatorCapabilities.

Table 6 - QueryAnnunciatorCapabilities message encoding

body └ (empty)
Empty message body

3.1.5.1.6 ID 2A11: QueryLightStatus

This message shall cause the receiving component to reply to the requestor with a ID 4A11h: ReportLightStatus.

Table 7 - QueryLightStatus message encoding

body └ (empty)
Empty message body

3.1.5.1.7 ID 2A12: QueryWhistle

This message shall cause the receiving component to reply to the requestor with a ID 4A12h: ReportWhistle.

Table 8 - QueryWhistle message encoding

body └ (empty)
Empty message body

3.1.5.1.8 ID 2A13: QueryBell

This message shall cause the receiving component to reply to the requestor with a ID 4A13h: ReportBell.

Table 9 - QueryBell message encoding

body └ (empty)
Empty message body

3.1.5.1.9 ID 2A14: QueryGong

This message shall cause the receiving component to reply to the requestor with a ID 4A14h: ReportGong.

Table 10 - QueryGong message encoding

body └ (empty)
Empty message body

3.1.5.2 Output Set

3.1.5.2.1 ID 4A10: ReportAnnunciatorCapabilities

This message is used to provide the receiver the platform's annunciator capabilities; i.e., the sounds, lights, and situational signals supported.

Table 11 - ReportAnnunciatorCapabilities message encoding

body └ record name = AnnunciatorCapabilitiesRec					
Record Name = AnnunciatorCapabilitiesRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<bit_field> LightsSupported	unsigned byte	bit_field	false	Bitfield of supported lights Bits 0..0, MastheadLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 1..1, PortSideLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 2..2, StarboardSideLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 3..3, SternLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 4..4, TowingLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 5..5, AllRoundLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 6..6, FlashingLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 7..7, UserDefinedLight: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED
2	<bit_field> SoundsSupported	unsigned byte	bit_field	false	Bitfield of supported sounds Bits 0..0, Whistle: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 1..1, Bell: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED Bits 2..2, Gong: Value set, offset=false, ranges/enums: 0= UNSUPPORTED 1= SUPPORTED

3.1.5.2.2 ID 4A11: ReportLightStatus

This message is used to provide the receiver the lights' on or off status.

Table 12 - ReportLightStatus message encoding

body └ record name = LightSwitchRec					
Record Name = LightSwitchRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<bit_field> Switch	unsigned byte	bit_field	false	Status of each light Bits 0..0, MastheadLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 1..1, PortSideLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 2..2, StarboardSideLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 3..3, SternLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 4..4, TowingLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 5..5, AllAroundLight: Value set, offset=false, ranges/enums: 1= ON 0= OFF Bits 6..6, FlashingLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON Bits 7..7, UserDefinedLight: Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> FlashDuration	float	second	true	Flash duration in seconds, when flashing light is activated
4	<fixed_field> FlashNumber	unsigned byte	one	true	1 to 255: The number of times remaining to flash 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	float	second	true	The time in seconds between flashes

3.1.5.2.3 ID 4A12: ReportWhistle

This message is used to provide the receiver the running status of the whistle sound.

Table 13 - ReportWhistle message encoding

body └ record name = SoundControlRec					
Record Name = SoundControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Switch	unsigned byte	one	false	Status enumerations Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> BlastDuration	unsigned byte	second	true	Blast duration in seconds
4	<fixed_field> BlastNumber	unsigned byte	one	true	1 to 255: The number of times remaining to blast 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	unsigned byte	second	true	The time in seconds between blasts

3.1.5.2.4 ID 4A13: ReportBell

This message is used to provide the receiver the running status of the bell sound.

Table 14 - ReportBell message encoding

body └ record name = SoundControlRec					
Record Name = SoundControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Switch	unsigned byte	one	false	Status enumerations Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> BlastDuration	unsigned byte	second	true	Blast duration in seconds
4	<fixed_field> BlastNumber	unsigned byte	one	true	1 to 255: The number of times remaining to blast 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	unsigned byte	second	true	The time in seconds between blasts

3.1.5.2.5 ID 4A14: ReportGong

This message is used to provide the receiver the running status of the gong sound.

Table 15 - ReportGong message encoding

body
└ record name = SoundControlRec

Record Name = SoundControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Switch	unsigned byte	one	false	Status enumerations Value set, offset=false, ranges/enums: 0= OFF 1= ON
3	<fixed_field> BlastDuration	unsigned byte	second	true	Blast duration in seconds
4	<fixed_field> BlastNumber	unsigned byte	one	true	1 to 255: The number of times remaining to blast 0: Continue indefinitely Value set, offset=false, ranges/enums: 0= INDEFINITELY [1,255]
5	<fixed_field> RestTime	unsigned byte	second	true	The time in seconds between blasts

3.1.6 Protocol Behavior

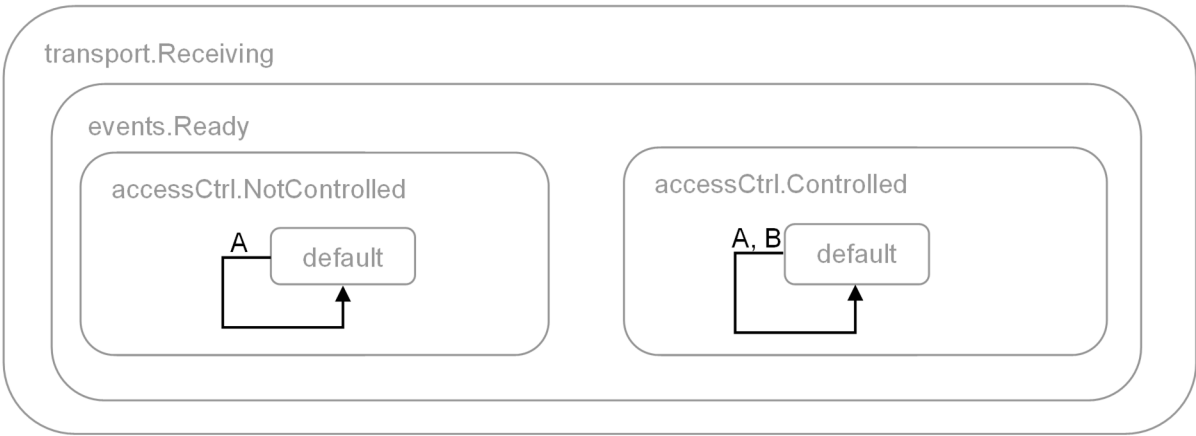


Figure 1 - Annunciator service protocol behavior

Table 16 - Annunciator service state transitions

Start State	Trigger	Conditions	Actions
B	SetLight	isControllingClient(transportData) && isSupportedRequest(msg)	updateLightControl (msg)
	SetWhistle	isControllingClient(transportData) && isSupportedRequest(msg)	updateWhistleControl (msg)
	SetBell	isControllingClient(transportData) && isSupportedRequest(msg)	updateBellControl (msg)
	SetGong	isControllingClient(transportData) && isSupportedRequest(msg)	updateGongControl (msg)
A	QueryAnnunciatorCapabilities		SendAnnunciatorCapabilities (msg, transportData)
	QueryLightStatus		SendLightStatus (msg, transportData)
	QueryWhistle		SendReportWhistle (msg, transportData)
	QueryBell		SendReportBell (msg, transportData)
	QueryGong		SendReportGong (msg, transportData)

Table 17 - Annunciator service conditions

Condition	Interpretation
isControllingClient(transportData) && isSupportedRequest(msg)	True if the message that triggered the transition is received from the client that is in control of this service AND the request is valid and supported.

Table 18 - Annunciator service transition actions

Action	Interpretation
updateLightControl	Update the lights' power control for the component
updateWhistleControl	Update the whistle control for the component
updateBellControl	Update the bell control for the component
updateGongControl	Update the gong control for the component
SendAnnunciatorCapabilities	Send a ReportAnnunciatorCapabilities message to the component that sent the query
SendLightStatus	Send a ReportLightStatus message to the component that sent the query
SendReportWhistle	Send a ReportWhistle message to the component that sent the query
SendReportBell	Send a ReportBell message to the component that sent the query
SendReportGong	Send a ReportGong message to the component that sent the query

3.2 BallastControl

name=BallastControl

version=1.0

id=urn:jaus:jss:umv:BallastControl

3.2.1 Description

The function of this service is to control and monitor the ballast tanks and their supporting pumps on the vehicle. The ballast tank control is measured by either volume or mass. A client initially finds out the identifications of all the ballast tanks and their associated ballast pumps via QueryBallastTankSpecifications. The controller then uses each reported BallastTankID or BallastPumpID to command the corresponding ballast tank. The filled level of each tank can be commanded and monitored. Four modes of operation (Off, Fill, Empty, and Trim) are supported per ballast pump. When trim mode is set, the BallastControl service is trying to keep the vehicle steady in a particular orientation at a desired depth.

3.2.2 Assumptions

Messages may be delayed, lost, or reordered.

3.2.3 References

Inherits-from=Management

version=1.1

id=urn:jaus:jss:core:Management

3.2.4 Vocabulary

Table 19 - BallastControl service message vocabulary

Message ID (hex)	Name	Command
Input Set		
0A86	BallastPumpControl (see 3.2.5.1.1)	true
0A87	BallastTankControl (see 3.2.5.1.2)	true
2A85	QueryBallastTankSpecifications (see 3.2.5.1.3)	false
2A86	QueryBallastPumpStatus (see 3.2.5.1.4)	false
2A87	QueryBallastTankStatus (see 3.2.5.1.5)	false
Output Set		
4A85	ReportBallastTankSpecifications (see 3.2.5.2.1)	false
4A86	ReportBallastPumpStatus (see 3.2.5.2.2)	false
4A87	ReportBallastTankStatus (see 3.2.5.2.3)	false

3.2.5 Encoding

3.2.5.1 Input Set

3.2.5.1.1 ID 0A86: BallastPumpControl

This message is used to set the active state of the ballast pumps associated with the specified BallastPumpIDs.

Table 20 - BallastPumpControl message encoding

<pre> body └─ list name = BallastPumpControlList └─ (count_field = unsigned byte) └─ sequence name = BallastPumpControlSeq └─ record name = BallastPumpControlRec └─ variant name = PumpFlowRateVar (optional=true) └─ (vtag_field = unsigned byte) └─ record name = VolumeFlowRateRec └─ record name = MassFlowRateRec </pre>					
Record Name = BallastPumpControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BallastPumpID	unsigned byte	one	False	1...255: the ballast pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
2	<fixed_field> StateCMD	unsigned byte	one	False	Value set, offset=false, ranges/enums: 0= OFF 1= FILL 2= EMPTY 3= TRIM
Record Name = VolumeFlowRateRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> VolumeFlowRateCMD	unsigned integer	liters per second	False	Flow rate to fill or empty. (scaled range = [0,1000000], round)
Record Name = MassFlowRateRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> MassFlowRateCMD	unsigned integer	kilograms per second	False	Flow rate to fill or empty. (scaled range = [0,1000000], round)

3.2.5.1.2 ID 0A87: BallastTankControl

This message is used to set the filled level of the ballast tanks associated with the specified BallastTankIDs.

Table 21 - BallastTankControl message encoding

<pre> body └─ list name = BallastTankList └─ (count_field = unsigned byte) └─ sequence name = BallastTankSeq └─ record name = BallastTankIDRec └─ variant name = BallastTankVar └─ (vtag_field = unsigned byte) └─ record name = LiquidTankRec └─ record name = MassTankRec └─ record name = TrimRec </pre>					
Record Name = BallastTankIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BallastTankID	unsigned byte	one	false	1...255: the ballast tank ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
Record Name = LiquidTankRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> LevelCMD	unsigned integer	percent	false	Tank filled percentage level, where 0 is empty and 100 is full. (scaled range = [0,100], round)
Record Name = MassTankRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> MassCMD	unsigned integer	kilogram	false	Tank filled mass (scaled range = [0,100000000], round)
Record Name = TrimRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> TrimCMD	unsigned byte	one	false	Value set, offset=false, ranges/enums: 0= NO_TRIM 1= ACTIVE

3.2.5.1.3 ID 2A85: QueryBallastTankSpecifications

This message shall cause the receiving component to reply to the requestor with a ID 4A85h: ReportBallastTankSpecifications. This message shall request the specifications of all the ballast tanks.

Table 22 - QueryBallastTankSpecifications message encoding

<pre> body └─ (empty) </pre>					
Empty message body					

3.2.5.1.4 ID 2A86: QueryBallastPumpStatus

This message shall cause the receiving component to reply to the requestor with a ID 4A86h: ReportBallastPumpStatus. This message shall request the current status of all the ballast pumps associated with the BallastPumpID.

Table 23 - QueryBallastPumpStatus message encoding

body └─ list name = BallastPumpIDList └─ (count_field = unsigned byte) └─ record name = BallastPumpIDRec					
Record Name = BallastPumpIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BallastPumpID	unsigned byte	one	false	1...255: the ballast tank ID. Note: 0 is used to query all ballast pumps. Value set, offset=false, ranges/enums: 0= ALL_PUMPS [1,255]

3.2.5.1.5 ID 2A87: QueryBallastTankStatus

This message shall cause the receiving component to reply to the requestor with a ID 4A87h: ReportBallastTankStatus. This message shall request the status of all the ballast tanks associated with the BallastTankID.

Table 24 - QueryBallastTankStatus message encoding

body └─ list name = BallastTankIDList └─ (count_field = unsigned byte) └─ record name = BallastTankIDRec					
Record Name = BallastTankIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BallastTankID	unsigned byte	one	false	1...255: the ballast tank ID. Note: 0 is used to query all ballast tanks. Value set, offset=false, ranges/enums: 0= ALL_TANKS [1,255]

3.2.5.2 Output Set

3.2.5.2.1 ID 4A85: ReportBallastTankSpecifications

This message is used to provide the receiver the ballast tank specifications values. Each ballast tank has a list of supporting ballast pumps. Note: To simplify control, both BallastTankID and BallastPumpID are required to be unique within this Ballast Control service.

Table 25 - ReportBallastTankSpecifications message encoding

<pre> body └─ list name = BallastTankSpecsList (count_field = unsigned byte) └─ sequence name = BallastTankSpecsSeq └─ record name = BallastTankSpecsRec └─ list name = BallastPumpList (count_field = unsigned byte) └─ variant name = BallastPumpSpecsVar (vtag_field = unsigned byte) └─ record name = VolumePumpSpecsRec └─ record name = MassPumpSpecsRec </pre>					
Record Name = BallastTankSpecsRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> BallastTankID	unsigned byte	one	false	1...255: the ballast tank ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
3	<variable_length_string> Description	variable length string (byte[])	N/A	false	Human-recognizable name of the ballast tank (Length min..max = 0..255)
4	<fixed_field> VolumeCapacity	unsigned integer	liter	true	Tank volume (scaled range = [0,100000000], round)
5	<fixed_field> MassCapacity	unsigned integer	kilogram	true	Tank mass capacity (scaled range = [0,100000000], round)
6	<fixed_field> TrimTank	unsigned byte	one	false	True if this is a trim tank Value set, offset=false, ranges/enums: 0= FALSE 1= TRUE
Record Name = VolumePumpSpecsRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> BallastPumpID	unsigned byte	one	false	1...255: the ballast pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
3	<fixed_field> MinEmptyRate	unsigned integer	liters per second	true	Minimum flow rate to empty (scaled range = [0,1000000], round)
4	<fixed_field> MaxEmptyRate	unsigned integer	liters per second	true	Maximum flow rate to empty (scaled range = [0,1000000], round)
5	<fixed_field> MinFillRate	unsigned integer	liters per second	true	Minimum flow rate to fill (scaled range = [0,1000000], round)
6	<fixed_field> MaxFillRate	unsigned integer	liters per second	true	Maximum flow rate to fill (scaled range = [0,1000000], round)

Record Name = MassPumpSpecsRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> BallastPumpID	unsigned byte	one	false	1...255: the ballast pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
3	<fixed_field> MinEmptyRate	unsigned integer	kilograms per second	true	Minimum flow rate to empty (scaled range = [0,1000000], round)
4	<fixed_field> MaxEmptyRate	unsigned integer	kilograms per second	true	Maximum flow rate to empty (scaled range = [0,1000000], round)
5	<fixed_field> MinFillRate	unsigned integer	kilograms per second	true	Minimum flow rate to fill (scaled range = [0,1000000], round)
6	<fixed_field> MaxFillRate	unsigned integer	kilograms per second	true	Maximum flow rate to fill (scaled range = [0,1000000], round)

3.2.5.2.2 ID 4A86: ReportBallastPumpStatus

This message is used to report the current status of the ballast pumps associated with the specified BallastPumpID. If a reported pump is not executing a flow rate command, do not report the corresponding VolumeFlowRateCMD or MassFlowRateCMD value.

Table 26 - ReportBallastPumpStatus message encoding

<pre> body └─ list name = BallastPumpStatusList └─ (count_field = unsigned byte) └─ sequence name = BallastPumpStatusSeq └─ record name = BallastPumpStatusRec └─ variant name = PumpFlowRateStatusVar (optional=true) └─ (vtag_field = unsigned byte) └─ record name = VolumeFlowRateStatusRec └─ record name = MassFlowRateStatusRec </pre>					
Record Name = BallastPumpStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BallastPumpID	unsigned byte	one	false	1...255: the ballast pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
2	<fixed_field> StateCMD	unsigned byte	one	false	Value set, offset=false, ranges/enums: 0= OFF 1= FILL 2= EMPTY 3= TRIM
3	<fixed_field> StateFB	unsigned byte	one	false	Value set, offset=false, ranges/enums: 0= OFF 1= FILL 2= EMPTY 3= TRIM 4= FAULT
Record Name = VolumeFlowRateStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> VolumeFlowRateCMD	unsigned integer	liters per second	true	Commanded flow rate (scaled range = [0,1000000], round)
3	<fixed_field> VolumeFlowRateFB	unsigned integer	liters per second	false	Current running flow rate (scaled range = [0,1000000], round)
Record Name = MassFlowRateStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> MassFlowRateCMD	unsigned integer	kilograms per second	true	Commanded flow rate (scaled range = [0,1000000], round)
3	<fixed_field> MassFlowRateFB	unsigned integer	kilograms per second	false	Current running flow rate (scaled range = [0,1000000], round)

3.2.5.2.3 ID 4A87: ReportBallastTankStatus

This message is used to provide the filled level of the ballast tanks associated with the specified BallastTankID. If a reported tank is not executing a filled level or filled mass command, do not report the corresponding LevelCMD or MassCMD value.

Table 27 - ReportBallastTankStatus message encoding

<pre> body └─ list name = BallastTankStatusList └─ (count_field = unsigned byte) └─ sequence name = BallastTankStatusSeq └─ record name = BallastTankIDRec └─ record name = TrimStatusRec (optional=true) └─ variant name = BallastTankStatusVar └─ (vtag_field = unsigned byte) └─ record name = LiquidTankStatusRec └─ record name = MassTankStatusRec </pre>					
Record Name = BallastTankIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BallastTankID	unsigned byte	one	false	1...255: the ballast tank ID. 0 is not a valid ID Value set, offset=false, ranges/enums: [1,255]
Record Name = TrimStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> TrimCMD	unsigned byte	one	true	Value set, offset=false, ranges/enums: 0= NO_TRIM 1= ACTIVE
3	<fixed_field> TrimFB	unsigned byte	one	true	Value set, offset=false, ranges/enums: 0= NO_TRIM 1= ACTIVE
Record Name = LiquidTankStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> LevelCMD	unsigned integer	percent	true	Commanded tank filled percentage level (scaled range = [0,100], round)
3	<fixed_field> LevelFB	unsigned integer	percent	false	Current tank filled percentage level (scaled range = [0,100], round)
Record Name = MassTankStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> MassCMD	unsigned integer	kilogram	true	Commanded tank filled mass (scaled range = [0,100000000], round)
3	<fixed_field> MassFB	unsigned integer	kilogram	false	Current tank filled mass (scaled range = [0,100000000], round)

3.2.6 Protocol Behavior

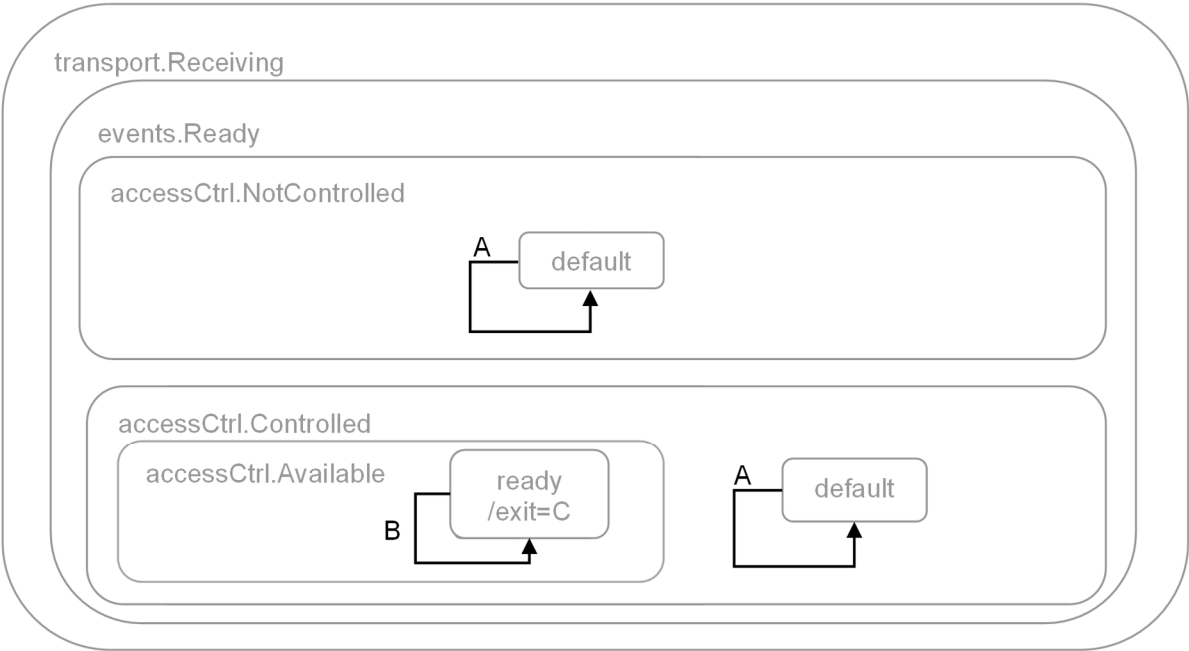


Figure 2 - BallastControl service protocol behavior

Table 28 - BallastControl service state transitions

Table 28A

Label	State	Type	Guard	Actions
C	management.ready	exit		stop

Table 28B

Start State	Trigger	Conditions	Actions
B	BallastPumpControl	isControllingClient(transportData)	setBallastPumpState (msg)
	BallastTankControl	isControllingClient(transportData)	setBallastTankLevel (msg)
A	QueryBallastTankSpecifications		sendReportBallastTankSpecifications (msg, transportData)
	QueryBallastPumpStatus		sendReportBallastPumpStatus (msg, transportData)
	QueryBallastTankStatus		sendReportBallastTankStatus (msg, transportData)

Table 29 - BallastControl service conditions

Condition	Interpretation
isControllingClient(transportData)	True if the message that triggered the transition is received from the client that is in control of this service.

Table 30 - BallastControl service transition actions

Action	Interpretation
setBallastPumpState	Set the ballast pump state as commanded in BallastPumpControl message.
setBallastTankLevel	Set the ballast tank level as commanded in BallastTankControl message.
sendReportBallastTankSpecifications	Send a ReportBallastTankSpecifications message to the component that sent the query
sendReportBallastPumpStatus	Send a ReportBallastPumpStatus message to the component that sent the query
sendReportBallastTankStatus	Send a ReportBallastTankStatus message to the component that sent the query
stop	stop all ballast pumps

3.3 BilgePumpControl

name=BilgePumpControl

version=1.0

id=urn:jaus:jss:umv:BilgePumpControl

3.3.1 Description

The function of this service is to control and monitor the bilge pumps on the vehicle. A client initially finds out the identifications of all the bilge pumps via QueryBilgePumpIdentification. The client then uses each reported BilgePumpID to command the corresponding bilge pump. Three modes of operation (Off, On, and Auto) are supported per bilge pump. The auto mode means the pump will automatically be turned on by the service when flood is detected in its area of responsibility. Note: Flood and leak reporting is provided by the Compartment Sensor Service.

3.3.2 Assumptions

Messages may be delayed, lost, or reordered.

3.3.3 References

Inherits-from=AccessControl

version=1.1

id=urn:jaus:jss:core:AccessControl

3.3.4 Vocabulary

Table 31 - BilgePumpControl service message vocabulary

Message ID (hex)	Name	Command
Input Set		
0A81	SetBilgePumpState (see 3.3.5.1.1)	true
2A80	QueryBilgePumpIdentification (see 3.3.5.1.2)	false
2A81	QueryBilgePumpStatus (see 3.3.5.1.3)	false
Output Set		
4A80	ReportBilgePumpIdentification (see 3.3.5.2.1)	false
4A81	ReportBilgePumpStatus (see 3.3.5.2.2)	false

3.3.5 Encoding

3.3.5.1 Input Set

3.3.5.1.1 ID 0A81: SetBilgePumpState

This message is used to set the active state of the bilge pumps associated with the specified BilgePumpIDs.

Table 32 - SetBilgePumpState message encoding

body └─ list name = BilgePumpControlList (count_field = unsigned byte) └─ record name = BilgePumpControlRec					
Record Name = BilgePumpControlRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BilgePumpID	unsigned byte	one	false	1...255: the bilge pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
2	<fixed_field> StateCMD	unsigned byte	one	false	Value set, offset=false, ranges/enums: 0= OFF 1= ON 2= AUTO

3.3.5.1.2 ID 2A80: QueryBilgePumpIdentification

This message shall cause the receiving component to reply to the requestor with a ID 4A80h: ReporBilgePumpIdentification. This message shall request the identifications of all the bilge pumps.

Table 33 - QueryBilgePumpIdentification message encoding

body └─ (empty)					
Empty message body					

3.3.5.1.3 ID 2A81: QueryBilgePumpStatus

This message shall cause the receiving component to reply to the requestor with a ID 4A81h: ReportBilgePumpStatus. This message shall request the current status of all the bilge pumps associated with the specified BilgePumpID.

Table 34 - QueryBilgePumpStatus message encoding

body └─ list name = BilgePumpIDList (count_field = unsigned byte) └─ record name = BilgePumpIDRec					
Record Name = BilgePumpIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BilgePumpID	unsigned byte	one	false	The bilge pump ID. Note: 0 is used to query all bilge pumps. Value set, offset=false, ranges/enums: 0= ALL_PUMPS [1,255]

3.3.5.2 Output Set

3.3.5.2.1 ID 4A80: ReportBilgePumpIdentification

This message is used to provide the receiver the bilge pump identification values.

Table 35 - ReportBilgePumpIdentification message encoding

<pre> body └─ list name = BilgePumpIdentificationList └─ (count_field = unsigned byte) └─ record name = BilgePumpIdentificationRec </pre>					
Record Name = BilgePumpIdentificationRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> BilgePumpID	unsigned byte	one	false	1...255: the bilge pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
3	<variable_length_string> Description	variable length string (byte[])	N/A	false	Human-recognizable name of the bilge pump. (Length min..max = 0..255)
4	<fixed_field> CompartmentID	unsigned byte	one	true	The compartment ID associated with this bilge pump. Zero (0) is not a valid ID Value set, offset=false, ranges/enums: [1,255]

3.3.5.2.2 ID 4A81: ReportBilgePumpStatus

This message is used to report the current status of the bilge pumps.

Table 36 - ReportBilgePumpStatus message encoding

<pre> body └─ list name = BilgePumpStatusList └─ (count_field = unsigned byte) └─ record name = BilgePumpStatusRec </pre>					
Record Name = BilgePumpStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BilgePumpID	unsigned byte	one	false	1...255: the bilge pump ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
2	<fixed_field> StateCMD	unsigned byte	one	false	Most recently requested command state Value set, offset=false, ranges/enums: 0= OFF 1= ON 2= AUTO
3	<fixed_field> StateFB	unsigned byte	one	false	Currently active state Value set, offset=false, ranges/enums: 0= OFF 1= ON 2= AUTO 3= FAULT

3.3.6 Protocol Behavior

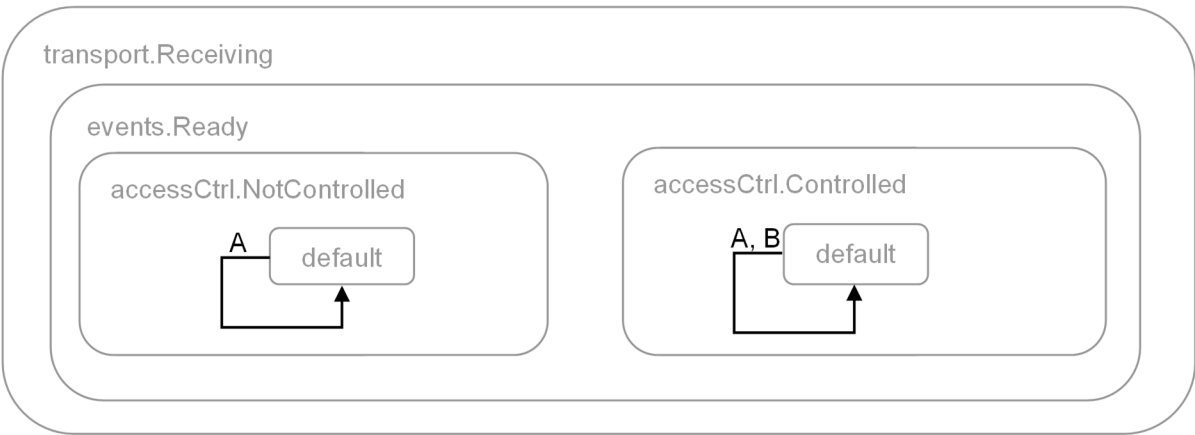


Figure 3 - BilgePumpControl service protocol behavior

Table 37 - BilgePumpControl service state transitions

Start State	Trigger	Conditions	Actions
B	SetBilgePumpState	isControllingClient(transportData)	setBilgePumpState (msg)
A	QueryBilgePumpIdentification		sendReportBilgePumpIdentification(msg, transportData)
	QueryBilgePumpStatus		sendReportBilgePumpStatus (msg, transportData)

Table 38 - BilgePumpControl service conditions

Condition	Interpretation
isControllingClient(transportData)	True if the message that triggered the transition is received from the client that is in control of this service.

Table 39 - BilgePumpControl service transition actions

Action	Interpretation
setBilgePumpState	Set the bilge pump state as commanded in SetBilgePumpState message.
sendReportBilgePumpIdentification	Send a ReportBilgePumpIdentification message to the component that sent the query.
sendReportBilgePumpStatus	Send a ReportBilgePumpStatus message to the component that sent the query.

3.4 BucketDriver

name=BucketDriver

version=1.0

id=urn:jaus:jss:umv:BucketDriver

3.4.1 Description

The function of this service is to report bucket identifications, set and report bucket positions. A water jet driven boat typically has a bucket for controlling the forward/reverse thrust. To control and monitor buckets on the platform, a client initially finds out the identifications of all the buckets via QueryBucketIdentification. The client then uses each reported bucket's name to query and set the corresponding bucket's position.

3.4.2 Assumptions

Messages may be delayed, lost, or reordered.

3.4.3 References

Inherits-from=management

version=1.1

id=urn:jaus:jss:core:Management

3.4.4 Vocabulary

Table 40 - BucketDriver service message vocabulary

Message ID (hex)	Name	Command
Input Set		
0A31	SetBucketPosition (see 3.4.5.1.1)	true
2A30	QueryBucketIdentification (see 3.4.5.1.2)	false
2A31	QueryBucketPosition (see 3.4.5.1.3)	false
Output Set		
4A30	ReportBucketIdentification (see 3.4.5.2.1)	false
4A31	ReportBucketPosition (see 3.4.5.2.2)	false

3.4.5 Encoding

3.4.5.1 Input Set

3.4.5.1.1 ID 0A31: SetBucketPosition

This message is used to provide position control on the bucket associated with the specified BucketID.

Table 41 - SetBucketPosition message encoding

<pre> body └─ list name = BucketPositionList └─ (count_field = unsigned byte) └─ record name = BucketPositionRec </pre>					
Record Name = BucketPositionRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BucketID	unsigned byte	one	false	The ID of the bucket being commanded. Zero (0) is not a valid ID Value set, offset=false, ranges/enums: [1,255]
2	<fixed_field> BucketPositionCMD	unsigned byte	percent	false	Command as a percent (scaled range = [-100,100], round)

3.4.5.1.2 ID 2A30: QueryBucketIdentification

This message shall cause the receiving component to reply to the requestor with a ID 4A30h: ReportBucketIdentification. This message shall request the identifications of all the buckets.

Table 42 - QueryBucketIdentification message encoding

body └ (empty)
Empty message body

3.4.5.1.3 ID 2A31: QueryBucketPosition

This message shall cause the receiving component to reply to the requestor with a ID 4A31h: ReportBucketPosition. This message shall request the current positions of all the buckets associated with the specified BucketID.

Table 43 - QueryBucketPosition message encoding

body - list name = BucketIDList (count_field = unsigned byte) - record name = BucketIDRec					
Record Name = BucketIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BucketID	unsigned byte	one	false	The bucket ID. Note: Zero (0) is used to query all buckets Value set, offset=false, ranges/enums: 0= ALL [1,255]

3.4.5.2 Output Set

3.4.5.2.1 ID 4A30: ReportBucketIdentification

This message is used to provide the receiver the bucket identification values.

Table 44 - ReportBucketIdentification message encoding

body - list name = BucketIdentificationList (count_field = unsigned byte) - record name = BucketIdentificationRec					
Record Name = BucketIdentificationRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> BucketID	unsigned byte	one	false	The bucket ID. Zero (0) is not a valid ID Value set, offset=false, ranges/enums: [1,255]
2	<variable_length_string> Name	variable length string (byte[])	N/A	false	Human-recognizable name of the bucket (Length min..max = 0..255)

3.4.5.2.2 ID 4A31: ReportBucketPosition

This message is used to provide the receiver the current bucket position values.

Table 45 - ReportBucketPosition message encoding

body

list

name = ReportBucketPositionList

(count_field = unsigned byte)

record

name = ReportBucketPositionRec

Record Name = ReportBucketPositionRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> BucketID	unsigned byte	one	false	The bucket ID. Zero (0) is not a valid ID Value set, offset=false, ranges/enums: [1,255]
3	<fixed_field> BucketPositionCMD	unsigned byte	percent	false	Commanded value as a percent (scaled range = [-100,100], round)
4	<fixed_field> BucketPositionFB	unsigned byte	one	false	Feedback as a percent (scaled range = [-100,100], round)
5	<fixed_field> TimeSeconds	unsigned integer	second	true	UTC POSIX time in seconds since 1 January 1970 of measured data
6	<fixed_field> TimeNanoseconds	unsigned integer	one	true	Nanoseconds component of POSIX time of measured data

3.4.6 Protocol Behavior

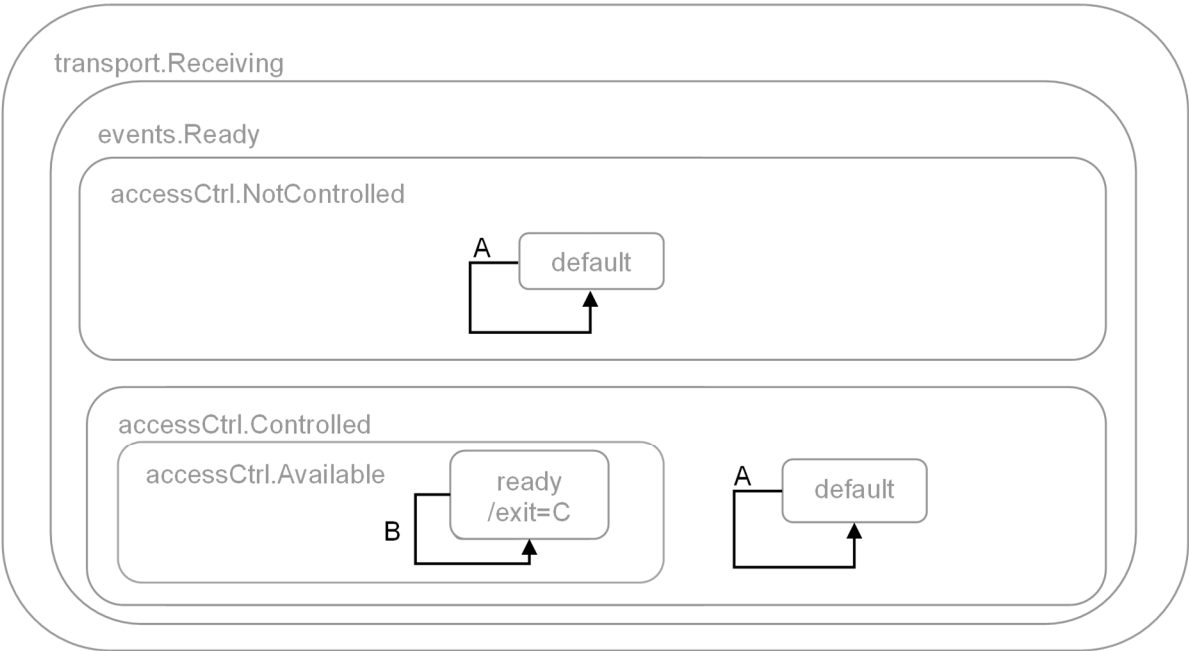


Figure 4 - BucketDriver service protocol behavior

Table 46 - BucketDriver service state transitions**Table 46A**

Label	State	Type	Guard	Actions
C	management.ready	exit		stop

Table 46B

Start State	Trigger	Conditions	Actions
B	SetBucketPosition	isControllingClient(transportData)	setBucketPosition (msg, transportData)
A	QueryBucketIdentification		sendReportBucketIdentification (msg, transportData)
	QueryBucketPosition		sendReportBucketPosition (msg, transportData)

Table 47 - BucketDriver service conditions

Condition	Interpretation
isControllingClient(transportData)	True if the message that triggered the transition is received from the client that is in control of this service.

Table 48 - BucketDriver service transition actions

Action	Interpretation
setBucketPosition	Set the bucket position
sendReportBucketIdentification	Send ReportBucketIdentification message to the component that sent the query
sendReportBucketPosition	Send ReportBucketPosition message to the component that sent the query
stop	Stop movement of all buckets

3.5 CTDSensor

name=CTDSensor

version=1.0

id=urn:jaus:jss:umv:CTDSensor

3.5.1 Description

The Conductivity, Temperature and Depth (CTD) Sensor has the responsibility of reporting the instantaneous water column CTD values measured at the vehicle's depth. It may also optionally report calculated values such as: sound velocity, salinity, and density.

3.5.2 Assumptions

Messages may be delayed, lost, or reordered.

3.5.3 References

Inherits-from=events

version=1.1

id=urn:jaus:jss:core:Events

3.5.4 Vocabulary

Table 49 - CTDSENSOR service message vocabulary

Message ID (hex)	Name	Command
Input Set		
2A45	QueryCTDState (see 3.5.5.1.1)	false
Output Set		
4A45	ReportCTDState (see 3.5.5.2.1)	false

3.5.5 Encoding

3.5.5.1 Input Set

3.5.5.1.1 ID 2A45: QueryCTDState

This message shall cause the receiving component to reply to the requestor with a ID 4A45h: ReportCTDState message.

Table 50 - QueryCTDState message encoding

body └ (empty)
Empty message body

3.5.5.2 Output Set

3.5.5.2.1 ID 4A45: ReportCTDState

This message is used to provide the receiver with the current set of CTD sensor information. Reported time is for the measured data.

Table 51 - ReportCTDState message encoding

body					
└ record name = CTDStateRec					
Record Name = CTDStateRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> Conductivity	unsigned integer	one	false	Siemens per meter (scaled range = [0,100], round)
3	<fixed_field> Temperature	unsigned integer	degree Celsius	false	(scaled range = [-10,60], round)
4	<fixed_field> Depth	unsigned integer	meter	false	(scaled range = [0,100000], round)
5	<fixed_field> EstimatedSoundVelocity	unsigned integer	meter per second	true	(scaled range = [0,2000], round)
6	<fixed_field> EstimatedSalinity	unsigned integer	parts per thousand	true	(scaled range = [0,1000], round)
7	<fixed_field> EstimatedDensity	unsigned integer	kilogram per cubic meter	true	(scaled range = [0,10000], round)
8	<fixed_field> TimeSeconds	unsigned integer	second	true	UTC POSIX time in seconds since 1 January 1970 for the measured data
9	<fixed_field> TimeNanoSeconds	unsigned integer	one	true	Nanoseconds component of POSIX time of measured data

3.5.6 Protocol Behavior

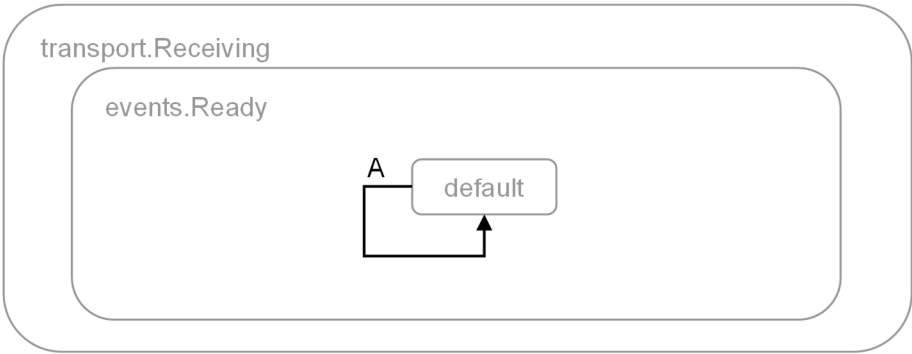


Figure 5 - CTDService service protocol behavior

Table 52 - CTDService service state transitions

Start State	Trigger	Conditions	Actions
A	QueryCTDState		sendReportCTDState (msg, transportData)

Table 53 - CTDSensor service transition actions

Action	Interpretation
sendReportCTDState	Send Report CTD State message to the component that sent the query.

3.6 CompartmentSensor

name=CompartmentSensor

version=1.0

id=urn:jaus:jss:umv:CompartmentSensor

3.6.1 Description

The function of this service is to report compartment identifications and current compartment status on the vehicle. It allows client service(s) to monitor essential compartment status such as temperature, humidity, flood and leak detection.

3.6.2 Assumptions

Messages may be delayed, lost, or reordered.

3.6.3 References

Inherits-from=events

version=1.1

id=urn:jaus:jss:core:Events

3.6.4 Vocabulary

Table 54 - CompartmentSensor service message vocabulary

Message ID (hex)	Name	Command
Input Set		
2A50	QueryCompartmentIdentification (see 3.6.5.1.1)	false
2A51	QueryCompartmentStatus (see 3.6.5.1.2)	false
Output Set		
4A50	ReportCompartmentIdentification (see 3.6.5.2.1)	false
4A51	ReportCompartmentStatus (see 3.6.5.2.2)	false

3.6.5 Encoding

3.6.5.1 Input Set

3.6.5.1.1 ID 2A50: QueryCompartmentIdentification

This message shall cause the receiving component to reply to the requestor with a ID 4A50h: ReportCompartmentIdentification. This message shall request the identification of every compartment monitored.

Table 55 - QueryCompartmentIdentification message encoding

body └ (empty)
Empty message body

3.6.5.1.2 ID 2A51: QueryCompartmentStatus

This message shall cause the receiving component to reply to the requestor with a ID 4A51h: ReportCompartmentStatus. This message shall request the current status of every compartment associated with the specified CompartmentID.

Table 56 - QueryCompartmentStatus message encoding

body └ list name = CompartmentIDList (count_field = unsigned byte) record name = CompartmentIDRec					
Record Name = CompartmentIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> CompartmentID	unsigned byte	one	false	The compartment ID. Note: Zero (0) is used to query all compartments Value set, offset=false, ranges/enums: 0= ALL [1,255]

3.6.5.2 Output Set

3.6.5.2.1 ID 4A50: ReportCompartmentIdentification

This message is used to provide the receiver the compartment identification values.

Table 57 - ReportCompartmentIdentification message encoding

body └ list name = CompartmentIdentificationList (count_field = unsigned byte) record name = CompartmentIdentificationRec					
Record Name = CompartmentIdentificationRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> CompartmentID	unsigned byte	one	false	The compartment ID. Zero (0) is not a valid ID Value set, offset=false, ranges/enums: [1,255]
2	<variable_length_string> Description	variable length string (byte[])	N/A	false	Human-recognizable name of the compartment (Length min..max = 0..255)

3.6.5.2.2 ID 4A51: ReportCompartmentStatus

This message is used to report the current status of the compartments specified in the query.

Table 58 - ReportCompartmentStatus message encoding

body list name = CompartmentStatusList (count_field = unsigned byte) record name = CompartmentStatusRec					
Record Name = CompartmentStatusRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> CompartmentID	unsigned byte	one	false	The compartment ID. Zero (0) is not a valid ID Value set, offset=false, ranges/enums: [1,255]
3	<fixed_field> FloodDetected	unsigned byte	one	true	Value set, offset=false, ranges/enums: 0= NOT_DETECTED 1= DETECTED
4	<fixed_field> LeakDetected	unsigned byte	one	true	Value set, offset=false, ranges/enums: 0= NOT_DETECTED 1= DETECTED
5	<fixed_field> Humidity	unsigned byte	percent	true	(scaled range = [0,100], round)
6	<fixed_field> Temperature	unsigned short integer	degree Celsius	true	(scaled range = [-75,180], round)

3.6.6 Protocol Behavior

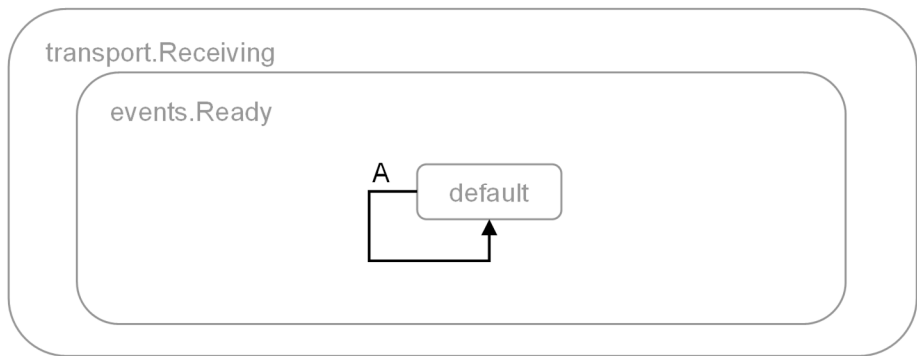


Figure 6 - CompartmentSensor service protocol behavior

Table 59 - CompartmentSensor service state transitions

Start State	Trigger	Conditions	Actions
A	QueryCompartmentIdentification		sendReportCompartmentIdentification(msg,transportData)
	QueryCompartmentStatus		sendReportCompartmentStatus (msg,transportData)

Table 60 - CompartmentSensor service transition actions

Action	Interpretation
sendReportCompartmentIdentification	Send a ReportCompartmentIdentification message to the component that sent the query
sendReportCompartmentStatus	Send a ReportCompartmentStatus message to the component that sent the query

3.7 ControlSurface

name=ControlSurface

version=1.0

id=urn:jaus:jss:umv:ControlSurface

3.7.1 Description

The function of the Control Driver Service is to provide mechanisms to directly control three different types of individual control surfaces as well as obtain their current position. Direct control is provided by the Set Control Surface message if sent by a controlling agent, the Query Control Surface message causes a Report Control Surface message to be returned to the requestor that contains the current command and if available the measured control surface positions. The Query Control Surface Specifications message causes a Report Control Surface Specifications message to be returned to the requestor.

3.7.2 Assumptions

Messages may be delayed, lost, or reordered.

3.7.3 References

Inherits-from=Management

version=1.1

id=urn:jaus:jss:core:Management

3.7.4 Vocabulary

Table 61 - ControlSurface service message vocabulary

Message ID (hex)	Name	Command
Input Set		
0A36	SetControlSurface (see 3.7.5.1.1)	true
2A35	QueryControlSurfaceSpecifications (see 3.7.5.1.2)	false
2A36	QueryControlSurface (see 3.7.5.1.3)	false
Output Set		
4A35	ReportControlSurfaceSpecifications (see 3.7.5.2.1)	false
4A36	ReportControlSurface (see 3.7.5.2.2)	false

3.7.5 Encoding

3.7.5.1 Input Set

3.7.5.1.1 ID 0A36: SetControlSurface

This message is used to set the desired control surface position on the surface associated with the specified ControlSurfaceID. If a commanded deflection rate exceeds the maximum or minimum allowed rate for that control surface, the control surface will deflect at its maximum or minimum rate, respectively. The commanded position is taken from each surface zero or null position. Positive values increase the surface vector direction with respect to NED coordinate system with origin in the center of buoyancy of the vehicle. Offset of control surfaces use the same definition as the offset for propulsors defined in PropulsionService.

Table 62 - SetControlSurface message encoding

body └ list name = ControlSurfaceList └ (count_field = unsigned byte) └ record name = ControlSurfaceRec					
Record Name = ControlSurfaceRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> ControlSurfaceID	unsigned byte	one	false	Control Surface ID. 0 is used to set all control surfaces Value set, offset=false, ranges/enums: 0= ALL_SURFACES [1,255]
3	<fixed_field> DeflectionCMD	unsigned short integer	percent	false	(scaled range = [-100,100], round)
4	<variable_field> DeflectionRateCMD	variable	N/A	True	Variable field for the requested deflection rate. Type and units should match that offered in the Report Control Surface Specifications message Index = 1 radian_per_second Field Type = float Field Units = radian per second Index = 2 meter_per_second Field Type = float Field Units = meter per second Index = 3 percent_per_second Field Type = float Field Units = one

3.7.5.1.2 ID 2A35: QueryControlSurfaceSpecifications

This message shall cause the receiving component to reply to the requestor with an ID 4A35h: ReportControlSurfaceSpecifications message. This message shall request the specifications of all the surfaces.

Table 63 - QueryControlSurfaceSpecifications message encoding

body └ (empty)
Empty message body

3.7.5.1.3 ID 2A36: QueryControlSurface

This message shall cause the receiving component to reply to the requestor with an ID 4A36h: ReportControlSurface message. A ControlSurfaceID of zero shall cause the deflections for all control surface to be returned.

Table 64 - QueryControlSurface message encoding

<pre> body └─ list name = ControlSurfaceIDList (count_field = unsigned byte) └─ record name = ControlSurfaceIDRec </pre>					
Record Name = ControlSurfaceIDRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<fixed_field> ControlSurfaceID	unsigned byte	one	false	Control Surface ID. 0 is used to request all surfaces. Value set, offset=false, ranges/enums: 0= ALL_SURFACES [1,255]

3.7.5.2 Output Set

3.7.5.2.1 ID 4A35: ReportControlSurfaceSpecifications

This message is used to provide the receiver with the specifications of the fins on the UMV. Position offset values have the same interpretation as the one described in Propulsion Service.

Table 65 - ReportControlSurfaceSpecifications message encoding

<pre> body └─ list name = ControlSurfaceSpecificationsList (count_field = unsigned byte) └─ record name = ControlSurfaceSpecificationsRec </pre>					
Record Name = ControlSurfaceSpecificationsRec					
Field #	Name	Type	Units	Optional	Interpretation
1	<presence_vector>	unsigned byte			
2	<fixed_field> ControlSurfaceID	unsigned byte	one	false	Control Surface ID. 0 is not a valid ID. Value set, offset=false, ranges/enums: [1,255]
3	<variable_length_string> Description	variable length string (byte[])	N/A	false	Human readable name of this control surface (Length min..max = 0..255)
4	<fixed_field> SurfaceType	unsigned byte	one	false	Value set, offset=false, ranges/enums: 1= Rudder 2= Fin 3= TrimTab 4= DivePlane 5= FinStabilizer 6= Sail
5	<fixed_field> PositionOffsetX	unsigned integer	meter	false	(scaled range = [-1000,1000], round)
6	<fixed_field> PositionOffsetY	unsigned integer	meter	false	(scaled range = [-500,500], round)
7	<fixed_field> PositionOffsetZ	unsigned integer	meter	false	(scaled range = [-500,500], round)