AEROSPACE STANDARD

AS6062™

REV. A

Issued 2010-07
Revised 2018-08
Reaffirmed 2024-03

Superseding AS6062

(R) JAUS Mission Spooling Service Set

RATIONALE

This document, the JAUS Mission Spooling Service Set (AS6062A), defines a task spooling interface that could be employed by mission planning services commonly found in unmanned systems. Additional capabilities are specified in the JAUS Core Service Set (AS5710A) and are frequently referenced herein. The reason for this revision is to correct technical problems and reduce the complexity of the Mission Spooler Service that has significantly hampered its adoption and use. The revision was so substantial that effectively no backward compatibility was maintained. It is recommended that all users of the Mission Spooling Service Set adopt revision A or later.

AS6062A has been reaffirmed to comply with the SAE Five-Year Review policy.

INTRODUCTION

The primary goal of the JAUS Mission Spooling 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 (AS5669A), is used in conjunction with this specification.

Each service in the JAUS Mission Spooling 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 Mission Spooling 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 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**

For more information on this standard, visit
https://www.sae.org/standards/content/AS6062A/

TABLE OF CONTENTS

1. SCOPE

This document defines a set of standard application layer interfaces called JAUS Mission Spooling 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 Mission Spooling Services represent the physical platform-independent capabilities commonly found across all domains and types of unmanned systems. At present, one service is defined in this document (more services are planned for future versions of this document):

- Mission Spooler: Stores, manages, and executes lists of tasks

The Mission Spooler service is described by a JAUS Service Definition (JSD) which specifies the message set and message protocol required for compliance. The 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 message protocol with regard to the planning and execution of missions, tasks and activities.

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 (versions compatible with "`urn:jaus:jss:core:MessageSet`" `version="1.1"` as defined in AS5710A). These documents must be used together to define a complete service.

It is important to note that details related to message serialization and protocol can be found in the JAUS Core Service Set document, in 2.2.6 “Serialization” and Section 4 “Understanding Protocol Descriptions,” respectively, and are not repeated here.

1.3 Compliance

The JAUS Mission Spooling 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. All implementations of the Mission Spooler Service with major version number two should be compliant to the Mission Spooler version 2.0 service specification herein.

1.4 Document Organization

The layout of this document is as follows: Section 2 lists external references and acronyms used throughout the specification. Section 3 describes common conventions used in the service description. Section 4 specifies the JAUS Service Definition for each of the Mission Spooling services, with particular emphasis on the description, assumptions, message set, and protocol behavior. Section 5 describes the message encoding for each message set. Section 6 has some notes about document revision. 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.

AS5669A JAUS/SDP Transport Specification

AS5684 JAUS Service Interface Definition Language

AS5710A JAUS Core Service Set

AS6009A JAUS Mobility Service Set

2.2 List of Acronyms

ID Identifier

JAUS Joint Architecture for Unmanned Systems

JSD JAUS Service Definition

JSIDL JAUS Service (Interface) Definition Language

XML Extensible Markup Language

3. COMMON CONVENTIONS

3.1 Defining a Mission

The Mission Spooler Service acts as an intermediary between a high-level planner (human or machine) and other SAE JAUS Services. The mission spooler may reside on-board the planner, the platform/payload, or a different subsystem altogether. Notional data flow is shown in Figure 1.

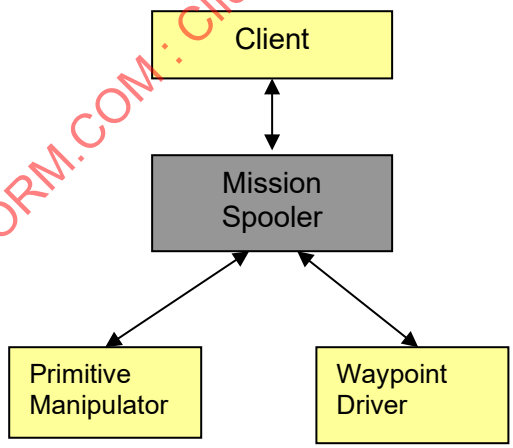


Figure 1 - Notional mission spooler configuration

The Mission Spooler represents a mission spool as one or more tasks. A task, in turn, is composed of a SAE JAUS message and parameters related to its execution. The two task parameters, ControlledNeeded and ReadyNeeded, indicate whether the service receiving the message must first be put into a particular state: AccessControl:Controlled and Management:Ready, respectively. In order to put a service into the AccessControl:Controlled state, a control request is made to the AccessControl service with an authority code that ranges from 0 to 255. The mission spooler assumes the access control authority of its client when requesting control. All messages associated with tasks are sent using the Mission Spooler as the source as opposed to the client.

Each list of tasks is linear and flat. When executing a list of tasks, you can optionally specify both the starting task in the list as well as whether to abort or continue execution of the task list if access control of the Mission Spooler is not maintained by its client.

The fundamental actions of any mission plan are based on SAE JAUS messages supported by the underlying services. For example, the GlobalPathSegmentDriver Service specified by the JAUS Mobility Service Set (AS6009A) defines the Set GlobalPathSegmentExt message for driving along a specified route. In this section, it is important to note that the mission actions are generalizations of behaviors offered by SAE JAUS services and are not new messages introduced by the Mission Spooler service. For instance, the SetGlobalPathSegmentExt message of the GlobalPathSegmentDriver service has been called “Drive Path Segment” in subsequent examples.

Following the execution of the message associated with a task and any specified AccessControl actions, a delay may be introduced using the optional Wait field. This delay will occur before the execution of the next task begins when there are additional tasks or before the spool is considered completed when there are no more tasks in the spool.

3.2 Blocking and Concurrent Task Execution

Blocking tasks are tasks that must be performed strictly sequentially and should contain a command message from the message set of a service that inherits from Events version 1.1 or compatible. In the mission spooler, a task is made into a blocking task simply by setting its blocking flag to true. When a blocking task is present, the task must be executed to successful completion (or unsuccessful termination) before subsequent tasks in the list can be executed. Conversely, concurrent tasks have their blocking flag set to false. Once a concurrent task is executed, the next task can begin immediately (after the executed task’s specified wait time), approximating parallel execution behavior. The examples below, describe both blocking and concurrent execution.

Assume that an unmanned system needs to traverse through two ordered path segments before it performs its first task with a manipulator. The manipulator task must be performed while the unmanned system is stationary. Once the manipulator task is completed, the unmanned system must traverse a third path segment. Only after completing the third path segment, must it perform a second task with a manipulator, change its camera pose and begin traversal of a fourth path segment. The actions that follow the traversal of the third path segment are not causally or temporally related to each other. That is, those tasks may be performed in any order. The mission for such a scenario is shown in Figure 2. The first four actions are flagged as blocking actions, implying synchronous execution. The last three actions are not flagged as blocking actions and therefore can be executed immediately after its predecessor’s specified wait time (nearly concurrent if the wait time is zero).

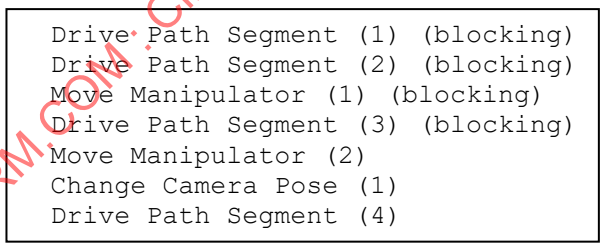


Figure 2 - Example of synchronous and asynchronous tasks

When a task is designated as blocking, the Mission Spooler service will send a CreateCommandEvent formed from the task’s JAUS message and using the optional Duration field for the MaximumAllowedDuration when it is specified. A blocking task’s message execution is then considered successful or unsuccessful when the corresponding CommandEvent message is received with a CommandResult of SUCCESSFUL or UNSUCCESSFUL, respectively. If the specified duration in the CreateCommandEvent is exceeded, then the CommandResult will be UNSUCCESSFUL in accordance with the urn:jaus:jss:core:Events (version compatible with 1.1) service protocol.

3.3 Spool Status

The status of a spool as reported by ID 4E05: ReportSpoolStatus changes based upon where that task list is in its creation and run cycle, as illustrated in Figure 3.

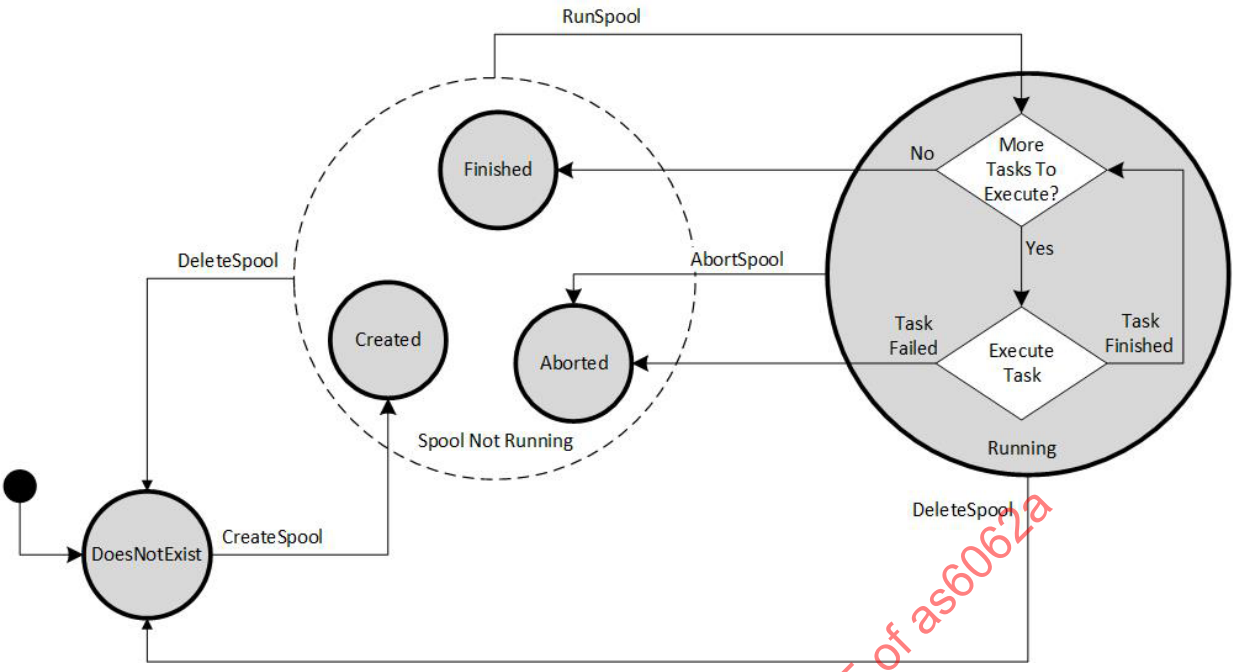


Figure 3 - Spool status

As shown in the figure, the initial status of a spool is DoesNotExist (i.e., no spool with a given ID is currently being stored). From the DoesNotExist status, a spool will have the Created status once the Mission Spooler service receives the ID 0E09: CreateSpool message defining its list of tasks. From any status where the spool has been created, but is not running (Created, Finished, or Aborted), a spool will transition into the Running status when the ID 0E0B: RunSpool message begins the execution of its task list. The spool can then enter either the Aborted or Finished status depending on whether the execution aborts or finishes all tasks, respectively. Addionally, a spool that has been created can later be deleted with the ID 0E0A: DeleteSpool message and return to the DoesNotExist status.

3.4 Task Status

The status of the current task as reported by ID 4E05: ReportSpoolStatus likewise changes based upon where that task is in its creation and run cycle. The definition of the different task statuses are detailed in Table 1, leaving flexibility for implementation whenever possible.

Table 1 - Task status definitions

Task Status	Description
NotCreated	No tasks have been created for this spool.
Created	The spool/task has been created, but not run.
Aborted	The execution of a task was aborted or failed before it finished and there is no further action (like releasing control) remaining.
ControlPending	The Mission Spooler service is attempting to put the destination component[1] into the AccessControl:Controlled state.
ReadyPending	The Mission Spooler service is attempting to put the destination component into the Management:Ready state.
Blocking	The Mission Spooler service has sent the CreateCommandEvent corresponding to a blocking task and is waiting for either a CommandEvent indicating the success or failure of the command or a RejectEventRequest when the command event creation fails.
ReleaseControlPending	The Mission Spooler service is attempting to release control of the destination component.
Waiting	A task has finished all actions associated with its execution and is waiting for the period specified in its Wait field before becoming finished.
Finished	A task has successfully completed execution.

4. SERVICE DEFINITIONS

The following subsections provide a textual definition for each Service Definition in the JAUS Mission Spooling Service Set (currently only the Mission Spooler service is defined, additional services will be defined in future versions of this document). Corresponding JSIDL definitions are offered in the Appendix.

4.1 Mission Spooler Service

Name=MissionSpooler
version=2.0
id= urn:jaus:jss:missionSpooler:MissionSpooler

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

Client-of Management
  name=management
  id= urn:jaus:jss:core:Management
  version=1.1

[1] For the purposes of this discussion, the term ‘”destination component” will refer to the the component that is the destination of a task’s JAUS message.

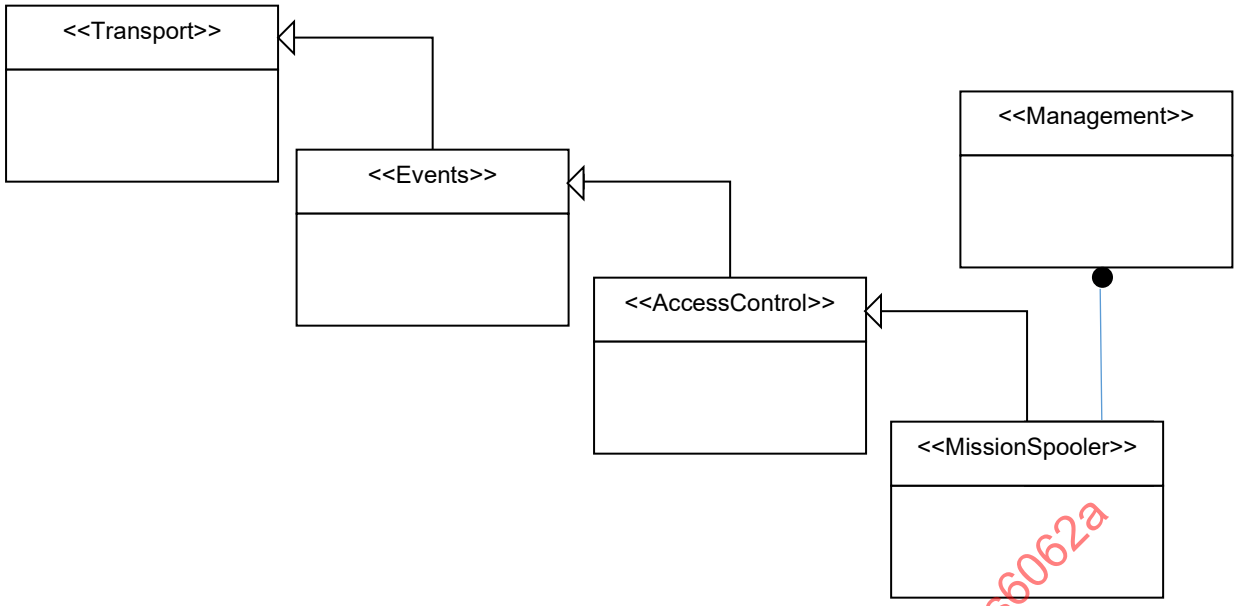


Figure 1 - Mission spooler service

4.1.1 Description

The Mission Spooler Service is responsible for storing and spooling linear lists of tasks. Each task list has a unique ID allowing for the storage of multiple lists. Each task in the list encompasses a SAE JAUS message (e.g., Set Global WaypointExt [AS6009A]) and parameters about its execution using one or more Services of one or more unmanned systems. Each task can be blocking or non-blocking, where a task designated as blocking should contain a command message from a service that inherits from Events version 1.1 or compatible. The Mission Spooler shall not spool tasks beyond a blocking task unless the unmanned system has completed the associated action. An action immediately preceding the use of a Payload is a good example of where blocking messages may be useful. Some payloads can only perform their functions when the unmanned system is stationary (e.g., soil sampling, video image) while other payloads can perform their functions (e.g., start mine flail) while in motion. The blocking flag ensures that no other messages are spooled until the blocking message is complete. If an optional wait time is specified for a task, the Mission Spooler shall wait for the specified period of time as the final step before the task is finished and the execution of additional tasks (when present) can begin.

4.1.2 Assumptions

Messages may be delayed, lost, or reordered.

4.1.3 Vocabulary

Table 2 lists the vocabulary of the Mission Spooler Service.

Table 2 - Mission spooler service vocabulary

Message ID (hex)	Name	Command
Input Set		
0E09	CreateSpool	true
0E0A	DeleteSpool	true
0E0B	RunSpool	true
0E0C	AbortSpool	true
2E04	QueryStoredSpools	false
2E05	QuerySpoolStatus	false
Output Set		
4E09	CreateSpoolResponse	false
4E0A	DeleteSpoolResponse	false
4E0B	RunSpoolResponse	false
4E0C	AbortSpoolResponse	false
4E04	ReportStoredSpools	false
4E05	ReportSpoolStatus	false

4.1.4 Protocol Behavior

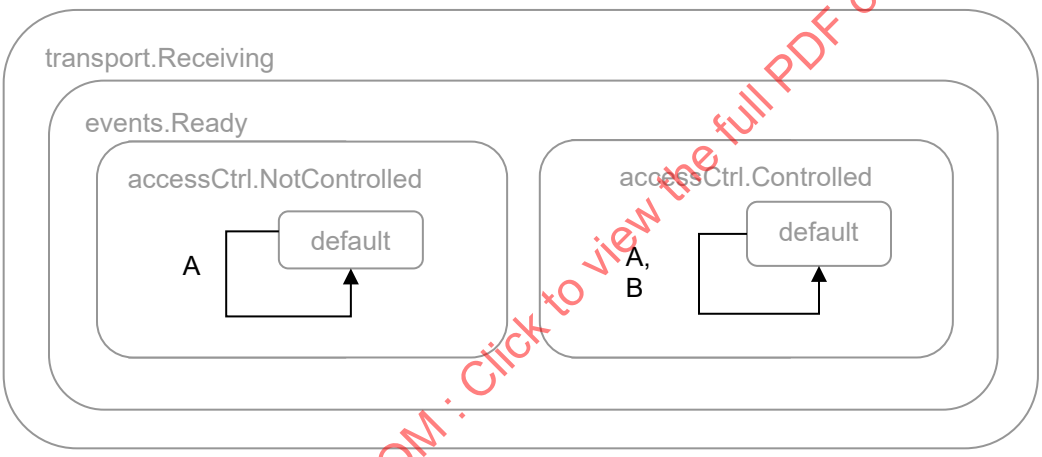


Figure 2 - Mission spooler service protocol behavior

Table 3 - Mission spooler service state transitions

Label	Trigger	Guard	*Actions*
A	QueryStoredSpools		*sendReportStoredSpools*
	QuerySpoolStatus		*sendReportSpoolStatus*
	CommandEvent	commandSuccessful && blocking	*continueSpool*
	CommandEvent	!commandSuccessful && blocking	*abortSpool*
	ConfirmEventRequest	blocking	*storeEventID*
	RejectEventRequest	blocking	*abortSpool*
B	CreateSpool	isControllingClient || isSelf	*createSpool*
	DeleteSpool	(isControllingClient || isSelf) && spoolExists && isExecuting	*abortSpool* *deleteSpool*
	DeleteSpool	(isControllingClient || isSelf) && spoolExists	*deleteSpool*
	DeleteSpool	(isControllingClient || isSelf) && !spoolExists	*sendDeleteSpoolResponse('SpoolDoesNotExist')*
	RunSpool	(isControllingClient || isSelf) && isExecuting	*sendRunSpoolResponse('SpoolAlreadyExecuting')*
	RunSpool	(isControllingClient || isSelf) && spoolExists	*executeSpool*
	RunSpool	(isControllingClient || isSelf) && !spoolExists	*sendRunSpoolResponse('SpoolDoesNotExist')*
	AbortSpool	(isControllingClient || isSelf) && spoolExists && isExecuting	*abortSpool*
	AbortSpool	(isControllingClient || isSelf) && spoolExists	*sendAbortSpoolResponse('SpoolNotExecuting')*
	AbortSpool	(isControllingClient || isSelf) && !spoolExists	*sendAbortSpoolResponse('SpoolDoesNotExist')*

Table 4 - Mission Spooler service conditions table

Condition	Interpretation
spoolExists	True if a spool with the given ID has been created.
commandSuccessful	True if the Command Event has a CommandResult of SUCCESSFUL.
blocking	True if a currently executing task list is waiting for the result of this Command Event to be reported.
isControllingClient	True if the command message was received from the client currently controlling this component.
isSelf	True if the command message was received from this component.

Table 5 - Mission spooler service transition actions

Action	Interpretation
sendReportStoredSpools	Send a ReportStoredSpools message to requesting client.
sendReportSpoolStatus	Send a ReportSpoolStatus message to requesting client.
abortSpool	Attempt to abort the execution of a task list with a given spool ID and call sendAbortSpoolResponse with either “SpoolAborted” or “SpoolNotAborted.”
continueSpool	Continue executing a task list that has been paused while a blocking command was pending.
createSpool	Attempt to create a new spool with the given task list and parameters and call sendCreateSpoolResponse with either “SpoolCreated” or “SpoolNotCreated.”
sendCreateSpoolResponse	Send a CreateSpoolResponse message to the client.
deleteSpool	Attempt to delete the spool indicated by the given spool ID and call sendDeleteSpoolResponse with either “SpoolDeleted” or “SpoolNotDeleted.”
sendDeleteSpoolResponse	Send a DeleteSpoolResponse message to the client.
sendRunSpoolResponse	Send a RunSpoolResponse message to the client.
sendAbortSpoolResponse	Send an AbortSpoolResponse message to the client.
executeSpool	Attempt to begin sequentially executing the tasks in a spool (at the given task index, if specified). When the next task is non-blocking, the service will send the message and immediately proceed to the next task action. If the next task is blocking, the service will create and send a CreateCommandEvent message with duration set as the maximum allowable duration and wait for a CommandEvent response or RejectEventRequest before proceeding to the next task action. Call sendRunSpoolResponse with either “SpoolStarted” or “SpoolNotStarted.”
storeEventID	Store the EventID assigned to the Command Event so that future responses can be matched to this request.

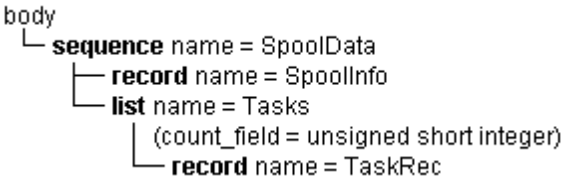
5. DECLARED TYPES

5.1 CommandClass

5.1.1 ID 0E09: CreateSpool

This message is used to communicate spools to a Mission Spooler service. The Request ID is a locally unique ID that the spooler returns in the create response message. The remaining fields define the spool, where the spool structure is a linear list called Tasks. Each task in the list can be either blocking or non-blocking, where blocking tasks will be realized using the Command Event mechanism. An optional duration field will specify the maximum time for a blocking task and an optional wait field will specify the delay between tasks. The remaining task parameters specify the Access Control and Management needs associated with executing the task.

Table 6 - CreateSpool message encoding



Record Name = SpoolInfo

Field #	Name	Type	Units	Optional	Interpretation
1	**<fixed_field>** RequestID	unsigned byte	one	false	Locally unique request ID for tracking the status of creating a spool
2	**<variable_length_string>** SpoolName	variable length string (byte[])	N/A	false	Human readable name for this spool (Length min..max = 0..255)

Record Name = TaskRec

Field #	Name	Type	Units	Optional	Interpretation
1	**<presence_vector>**	unsigned byte			
2	**<bit_field>** DestinationID	unsigned integer	bit_field	false	Identifier of destination to send the message Bits 0..7, ComponentID: Component ID where a value of 255 represents all components. Value set, offset=false, ranges/enums: [1,255] Bits 8..15, NodeID: Node ID where a value of 255 represents all nodes. Value set, offset=false, ranges/enums: [1,255] Bits 16..31, SubsystemID: Subsystem ID, where a value of 65535 represents all subsystems Value set, offset=false, ranges/enums: [1,65535]
3	**<variable_length_field>** message	JAUS MESSAGE	N/A	false	The JAUS message to be sent by the spooler. This includes the JAUS message header and footer but not the transport header. Count field type = unsigned integer (min/max count = 2..2147483647)
4	**<bit_field>** TaskParameters	unsigned byte	bit_field	false	Parameters dealing with the execution of this message including blocking and Access Control and Management information Bits 0..0, Blocking: Whether or not the message is blocking. Value set, offset=false, ranges/enums: 0= False 1= True Bits 1..1, ControlledNeeded: Whether or not the message requires the target component to be in Access Control Controlled state. Value set, offset=false, ranges/enums: 0= False 1= True Bits 2..2, ReadyNeeded: Whether or not the message requires the target component to be in Management Ready state. Value set, offset=false, ranges/enums: 0= False 1= True Bits 3..3, MaintainControlAfterTaskCompletes: Whether or not control of the target component should be maintained after the message is acted on. Value set, offset=false, ranges/enums: 0= False 1= True

5	**<fixed_field>** Duration	unsigned integer	second	true	Maximum allowed duration of a blocking message, assumed to be the maximum if not specified. When this time is exceeded, the blocking message times out and the spool will be aborted.
6	**<fixed_field>** Wait	float	second	true	Wait time between the completion of this task and the beginning of the next in seconds, assumed to be 0 if not specified

5.1.2 ID 0E0A: DeleteSpool

Removes a spool from those stored by the service. If the targeted spool is currently executing, this message will cause that execution to be aborted.

Table 7 - DeleteSpool message encoding

body └ **record** name = DeleteSpoolRec					
Record Name = DeleteSpoolRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool

5.1.3 ID 0E0B: RunSpool

This message is used to begin execution of the spool with the given Spool ID

Table 8 - RunSpool message encoding

body └ **record** name = RunSpoolRec					
Record Name = RunSpoolRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<presence_vector>**	unsigned byte			
2	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool
3	**<fixed_field>** ControlLostBehavior	unsigned byte	one	true	Defines what to do with active spool execution if control is lost by the client that initiated the execution, assumed to be ContinueSpoolExecution when not specified. Value set, offset=false, ranges/enums: 0= AbortSpoolExecution 1= ContinueSpoolExecution
4	**<fixed_field>** StartingTaskStep	unsigned byte	one	true	Specifies which task in the zero-indexed task list to start execution with, assumed to be 0 if not specified

5.1.4 ID 0E0C: AbortSpool

This message is used to abort the execution of the spool with the given Spool ID.

Table 9 - AbortSpool message encoding

body └ **record** name = AbortSpoolRec					
Record Name = AbortSpoolRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool

5.2 QueryClass

5.2.1 ID 2E04: QueryStoredSpools

This message shall cause the receiving component to reply to the requestor with a ReportStoredSpools message. The requester can specify whether to include task information in the returned report or just the names and IDs of the stored spool(s). If the QueryStoredSpoolsList is empty, all stored spools will be returned.

Table 10 - QueryStoredSpools message encoding

body └ **sequence** name = QueryStoredSpoolsSeq ├ **record** name = IncludeTasksRec └ **list** name = QueryStoredSpoolsList │ (count_field = unsigned byte) └ **record** name = SpoolIDRec					
Record Name = IncludeTasksRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** IncludeTasks	unsigned byte	one	false	Whether or not to include task information in the report or just the spool name and ID. Value set, offset=false, ranges/enums: 0= False 1= True
Record Name = SpoolIDRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of a stored spool

5.2.2 ID 2E05: QuerySpoolStatus

This message shall cause the receiving component to reply to the requestor with the ReportSpoolStatus message. An empty QuerySpoolStatusList will result in all known spools being returned.

Table 11 - QuerySpoolStatus message encoding

body └ **list** name = QuerySpoolStatusList │ (count_field = unsigned byte) └ **record** name = SpoolIDRec					
Record Name = SpoolIDRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool

5.3 InformClass

5.3.1 ID 4E09: CreateSpoolResponse

Response to the 0x0E09 CreateSpool message with the spool ID and success/failure of the action indicated.

Table 12 - CreateSpoolResponse message encoding

body └ **record** name = CreateSpoolResponseRec					
Record Name = CreateSpoolResponseRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<presence_vector>**	unsigned byte			
2	**<fixed_field>** RequestID	unsigned byte	one	false	Locally unique request ID for tracking the status of creating a spool
3	**<fixed_field>** Response	unsigned byte	one	false	Indicates whether the list was successfully created Value set, offset=false, ranges/enums: 0= SpoolCreated 1= SpoolNotCreated
4	**<fixed_field>** SpoolID	unsigned byte	one	true	Locally unique ID for the saved spool that is assigned if the spool is successfully created

5.3.2 ID 4E0A: DeleteSpoolResponse

Response to the 0x0E0A DeleteSpool message with the spool ID and success/failure of the action indicated.

Table 13 - DeleteSpoolResponse message encoding

body └ **record** name = DeleteSpoolResponseRec					
Record Name = DeleteSpoolResponseRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool
2	**<fixed_field>** Response	unsigned byte	one	false	Indicates whether the list was successfully deleted Value set, offset=false, ranges/enums: 0= SpoolDeleted 1= SpoolNotDeleted 2= SpoolDoesNotExist

5.3.3 ID 4E0B: RunSpoolResponse

Response to the 0x0E0B RunSpool message with the spool ID and success/failure of the action indicated.

Table 14 - RunSpoolResponse message encoding

body └ **record** name = RunSpoolResponseRec					
Record Name = RunSpoolResponseRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool
2	**<fixed_field>** Response	unsigned byte	one	false	Indicates whether the list was successfully started Value set, offset=false, ranges/enums: 0= SpoolStarted 1= SpoolNotStarted 2= SpoolDoesNotExist 3= SpoolAlreadyExecuting

5.3.4 ID 4E0C: AbortSpoolResponse

Response to the 0x0E0C AbortSpool message with the spool ID and success/failure of the action indicated.

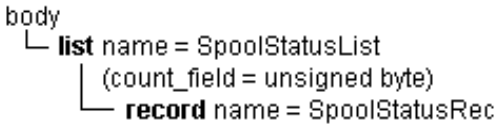
Table 15 - AbortSpoolResponse message encoding

body └ **record** name = AbortSpoolResponseRec					
Record Name = AbortSpoolResponseRec					
Field #	**Name**	**Type**	**Units**	**Optional**	**Interpretation**
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool
2	**<fixed_field>** Response	unsigned byte	one	false	Indicates whether the spool execution was successfully aborted Value set, offset=false, ranges/enums: 0= SpoolAborted 1= SpoolNotAborted 2= SpoolNotExecuting 3= SpoolDoesNotExist

5.3.5 ID 4E05: ReportSpoolStatus

This message is used to provide the receiver a list containing the status of each queried spool and well as the status of its current task. The available spool statuses are: “DoesNotExist” for an ID that does not have a spool associated with it; “Created” for a spool that has been created but not run; “Running” for a spool being actively executed; “Finished” for a spool that has run to completion; and “Aborted” for a spool that has aborted its execution before finishing. The available task statuses are: “NotCreated” for when a spool with the given ID has not been created; “Created” for a task that has been created but not run; “Aborted” for a task that was aborted or failed before finishing; “ControlPending” for a task attempting to gain access control; “ReadyPending” for a task attempting to set the Management::Ready state; ‘‘Blocking’ for a task waiting for a blocking command event to finish execution; ReleaseControlPending for a task attempting to release access control; “Waiting” for a task that is currently waiting the specified period before finishing; and “Finished” for a task that has successfully run to completion.

Table 16 - ReportSpoolStatus message encoding



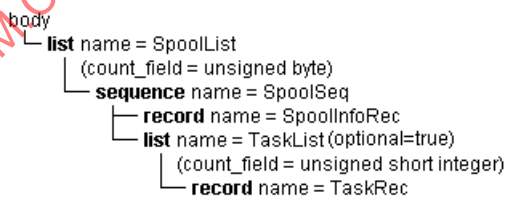
Record Name = SpoolStatusRec

Field #	Name	Type	Units	Optional	Interpretation
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool
2	**<fixed_field>** CurrentSpoolIndex	unsigned byte	one	false	The index of the task in the task list that corresponds to either the currently executing task when the spool is running, the task on which the execution was aborted, or the first task for both finished and not-started executions.
3	**<fixed_field>** SpoolStatus	unsigned byte	one	false	Value set, offset=false, ranges/enums: 0= DoesNotExist 1= Created 2= Running 3= Finished 4= Aborted
4	**<fixed_field>** TaskStatus	unsigned byte	one	false	Value set, offset=false, ranges/enums: 0= Created 1= NotCreated 2= Aborted 3= ControlPending 4= ReadyPending 5= Blocking 6= ReleaseControlPending 7= Waiting 8= Finished

5.3.6 ID 4E04: ReportStoredSpools

This message is used to provide the receiver with a list of spools stored in the Mission Spooler.

Table 17 - ReportStoredSpools message encoding



Record Name = SpoolInfoRec

Field #	Name	Type	Units	Optional	Interpretation
1	**<fixed_field>** SpoolID	unsigned byte	one	false	ID of the saved spool
2	**<variable_length_string>** SpoolName	variable length string (byte[])	N/A	false	Human readable name for this spool (Length min..max = 0..255)

Record Name = TaskRec

Field #	Name	Type	Units	Optional	Interpretation
1	**<presence_vector>**	unsigned byte			
2	**<bit_field>** DestinationID	unsigned integer	bit_field	false	Identifier of destination to send the message Bits 0..7, ComponentID: Component ID where a value of 255 represents all components. Value set, offset=false, ranges/enums:

					[1,255] Bits 8..15, NodeID: Node ID where a value of 255 represents all nodes. Value set, offset=false, ranges/enums: [1,255] Bits 16..31, SubsystemID: Subsystem ID, where a value of 65535 represents all subsystems Value set, offset=false, ranges/enums: [1,65535]
3	**<variable_length_field>** message	JAUS MESSAGE	N/A	false	The JAUS message to be sent by the spooler. Count field type = unsigned integer (min/max count = 2..2147483647)
4	**<bit_field>** TaskParameters	unsigned byte	bit_field	false	Parameters dealing with the execution of this message including blocking and Access Control and Management information Bits 0..0, Blocking: Whether or not the message is blocking. Value set, offset=false, ranges/enums: 0= False 1= True Bits 1..1, ControlledNeeded: Whether or not the message requires the target component to be in Access Control Controlled state. Value set, offset=false, ranges/enums: 0= False 1= True Bits 2..2, ReadyNeeded: Whether or not the message requires the target component to be in Management Ready state. Value set, offset=false, ranges/enums: 0= False 1= True Bits 3..3, MaintainControlAfterTaskCompletes: Whether or not control of the target component should be maintained after the message is acted on. Value set, offset=false, ranges/enums: 0= False 1= True
5	**<fixed_field>** Duration	unsigned integer	second	true	Maximum allowed duration of a blocking message, assumed to be the maximum if not specified. When this time is exceeded, the blocking message times out and the spool will be aborted.
6	**<fixed_field>** Wait	float	second	true	Wait time between the completion of this task and the beginning of the next in ms.

6. NOTES

6.1 Revision Indicator

A change bar (I) 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 MISSION SPOOLER

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<service_def name="MissionSpooler" id="urn:jaus:jss:missionSpooler:MissionSpooler" version="2.0" xmlns:ns2="urn:jaus:jsidl:1.1" xmlns="urn:jaus:jsidl:1.0" xmlns:ns3="urn:jaus:jsidl:plus">

  <description xml:space="preserve">The Mission Spooler Service is responsible for storing and spooling linear lists of tasks. Each task list has a unique ID allowing for the storage of multiple lists. Each task in the list encompasses a SAE JAUS message (e.g., Set Global WaypointExt [AS6009A]) and parameters about its execution using one or more Services of one or more unmanned systems. Each task can be blocking or non-blocking, where a task designated as blocking should contain a command message from a service that inherits from Events version 1.1 or compatible. The Mission Spooler shall not spool tasks beyond a blocking task unless the unmanned system has completed the associated action. An action immediately preceding the use of a Payload is a good example of where blocking messages may be useful. Some payloads can only perform their functions when the unmanned system is stationary (e.g. soil sampling, video image) while other payloads can perform their functions (e.g. start mine flail) while in motion. The blocking flag ensures that no other messages are spooled until the blocking message is complete. If an optional wait time is specified for a task, the Mission Spooler shall wait for the specified period of time as the final step before the task is finished and the execution of additional tasks (when present) can begin.</description>

  <assumptions xml:space="preserve">Messages may be delayed, lost or reordered.</assumptions>

  <references>

    <inherits_from name="AccessControl" id="urn:jaus:jss:core:AccessControl" version="1.1"/>

    <client_of name="Management" id="urn:jaus:jss:core:Management" version="1.1"/>

  </references>

  <message_set>

    <input_set>

      <message_def name="CreateSpool" message_id="0E09" is_command="true">

        <description xml:space="preserve">This message is used to communicate spools to a Mission Spooler service. The Request ID is a locally unique ID that the spooler returns in the create response message. The remaining fields define the spool, where the spool structure is a linear list called Tasks. Each task in the list can be either blocking or non-blocking, where blocking tasks will be realized using the Command Event mechanism. An optional duration field will specify the maximum time for a blocking task and an optional wait field will specify the delay between tasks. The remaining task parameters specify the Access Control and Management needs associated with executing the task.</description>

        <header name="JTS_DefaultHeader">

          <record name="DefaultHeaderRec" optional="false">

            <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

          </record>

        </header>

        <body name="Body">

          <sequence name="SpoolData" optional="false">
```

```
<record name="SpoolInfo" optional="false">
<fixed_field name="RequestID" optional="false" interpretation="Locally unique request ID for tracking the status of creating a spool" field_type="unsigned byte" field_units="one"/>
<variable_length_string name="SpoolName" optional="false" interpretation="Human readable name for this spool">
<count_field min_count="0" max_count="255" field_type_unsigned="unsigned byte"/>
</variable_length_string>
</record>
<list name="Tasks" optional="false" interpretation="List of JAUS Messages for this spool">
<count_field min_count="0" max_count="65535" field_type_unsigned="unsigned short integer"/>
<record name="TaskRec" optional="false">
<presence_vector field_type_unsigned="unsigned byte"/>
<bit_field name="DestinationID" optional="false" interpretation="Identifier of destination to send the message" field_type_unsigned="unsigned integer">
<sub_field name="ComponentID" interpretation="Component ID where a value of 255 represents all components.">
<bit_range from_index="0" to_index="7"/>
<value_set offset_to_lower_limit="false">
<value_range lower_limit="1" lower_limit_type="inclusive" upper_limit="255" upper_limit_type="inclusive"/>
</value_set>
</sub_field>
<sub_field name="NodeID" interpretation="Node ID where a value of 255 represents all nodes.">
<bit_range from_index="8" to_index="15"/>
<value_set offset_to_lower_limit="false">
<value_range lower_limit="1" lower_limit_type="inclusive" upper_limit="255" upper_limit_type="inclusive"/>
</value_set>
</sub_field>
<sub_field name="SubsystemID" interpretation="Subsystem ID, where a value of 65535 represents all subsystems">
<bit_range from_index="16" to_index="31"/>
<value_set offset_to_lower_limit="false">
```

<value_range lower_limit="1" lower_limit_type="inclusive" upper_limit="65535" upper_limit_type="inclusive"/>

</value_set>

</sub_field>

</bit_field>

<variable_length_field name="message" optional="false" interpretation="The JAUS message to be sent by the spooler. This includes the JAUS message header and footer but not the transport header." field_format="JAUS MESSAGE">

<count_field min_count="2" max_count="2147483647" field_type_unsigned="unsigned integer"/>

</variable_length_field>

<bit_field name="TaskParameters" optional="false" interpretation="Parameters dealing with the execution of this message including blocking and Access Control and Management information" field_type_unsigned="unsigned byte">

<sub_field name="Blocking" interpretation="Whether or not the message is blocking.">

<bit_range from_index="0" to_index="0"/>

<value_set offset_to_lower_limit="false">

<value_enum enum_index="0" enum_const="False"/>

<value_enum enum_index="1" enum_const="True"/>

</value_set>

</sub_field>

<sub_field name="ControlledNeeded" interpretation="Whether or not the message requires the target component to be in Access Control Controlled state.">

<bit_range from_index="1" to_index="1"/>

<value_set offset_to_lower_limit="false">

<value_enum enum_index="0" enum_const="False"/>

<value_enum enum_index="1" enum_const="True"/>

</value_set>

</sub_field>

<sub_field name="ReadyNeeded" interpretation="Whether or not the message requires the target component to be in Management Ready state.">

<bit_range from_index="2" to_index="2"/>

<value_set offset_to_lower_limit="false">

<value_enum enum_index="0" enum_const="False"/>

<value_enum enum_index="1" enum_const="True"/>

</value_set>

</sub_field>

<sub_field name="MaintainControlAfterTaskCompletes" interpretation="Whether or not control of the target component should be maintained after the message is acted on.">

<bit_range from_index="3" to_index="3"/>

<value_set offset_to_lower_limit="false">

<value_enum enum_index="0" enum_const="False"/>

<value_enum enum_index="1" enum_const="True"/>

</value_set>

</sub_field>

</bit_field>

<fixed_field name="Duration" optional="true" interpretation="Maximum allowed duration of a blocking message, assumed to be the maximum if not specified. When this time is exceeded, the blocking message times out and the spool will be aborted." field_type="unsigned integer" field_units="second"/>

<fixed_field name="Wait" optional="true" interpretation="Wait time between the completion of this task and the beginning of the next in seconds, assumed to be 0 if not specified" field_type="float" field_units="second"/>

</record>

</list>

</sequence>

</body>

<footer name="Footer"/>

</message_def>

<message_def name="DeleteSpool" message_id="0E0A" is_command="true">

<description xml:space="preserve">Removes a spool from those stored by the service. If the targeted spool is currently executing, this message will cause that execution to be aborted.</description>

<header name="JTS_DefaultHeader">

<record name="DefaultHeaderRec" optional="false">

<fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

</record>

</header>

<body name="Body">

```
        <record name="DeleteSpoolRec" optional="false">
          <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>
        </record>
      </body>
      <footer name="Footer"/>
    </message_def>
    <message_def name="RunSpool" message_id="0E0B" is_command="true">
      <description xml:space="preserve">This message is used to begin execution of the spool with the given Spool ID.</description>
      <header name="JTS_DefaultHeader">
        <record name="DefaultHeaderRec" optional="false">
          <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>
        </record>
      </header>
      <body name="Body">
        <record name="RunSpoolRec" optional="false">
          <presence_vector field_type_unsigned="unsigned byte"/>
          <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>
          <fixed_field name="ControlLostBehavior" optional="true" interpretation="Defines what to do with active spool execution if control is lost by the client that initiated the execution, assumed to be ContinueSpoolExecution when not specified." field_type="unsigned byte" field_units="one">
            <value_set offset_to_lower_limit="false">
              <value_enum enum_index="0" enum_const="AbortSpoolExecution"/>
              <value_enum enum_index="1" enum_const="ContinueSpoolExecution"/>
            </value_set>
          </fixed_field>
          <fixed_field name="StartingTaskStep" optional="true" interpretation="Specifies which task in the zero-indexed task list to start execution with, assumed to be 0 if not specified" field_type="unsigned byte" field_units="one"/>
        </record>
      </body>
      <footer name="Footer"/>
```

```
        </message_def>

        <message_def name="AbortSpool" message_id="0E0C" is_command="true">

            <description xml:space="preserve">This message is used to abort the execution of the spool with the given Spool ID.</description>

            <header name="JTS_DefaultHeader">

                <record name="DefaultHeaderRec" optional="false">

                    <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

                </record>

            </header>

            <body name="Body">

                <record name="AbortSpoolRec" optional="false">

                    <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>

                </record>

            </body>

            <footer name="Footer"/>

        </message_def>

        <message_def name="QueryStoredSpools" message_id="2E04" is_command="false">

            <description xml:space="preserve">This message shall cause the receiving component to reply to the requestor with a ReportStoredSpools message. The requester can specify whether to include task information in the returned report or just the names and IDs of the stored spool(s). If the QueryStoredSpoolsList is empty, all stored spools will be returned.</description>

            <header name="JTS_DefaultHeader">

                <record name="DefaultHeaderRec" optional="false">

                    <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

                </record>

            </header>

            <body name="Body">

                <sequence name="QueryStoredSpoolsSeq" optional="false">

                    <record name="IncludeTasksRec" optional="false">

                        <fixed_field name="IncludeTasks" optional="false" field_type="unsigned byte" field_units="one" interpretation="Whether or not to include task information in the report or just the spool name and ID.">

                            <value_set offset_to_lower_limit="false">
```

```
                    <value_enum enum_index="0" enum_const="False"/>
                    <value_enum enum_index="1" enum_const="True"/>
                  </value_set>
                </fixed_field>
              </record>
              <list name="QueryStoredSpoolsList" optional="false" interpretation="A list of spool IDs to report on.  A zero size list indicates that all spools are to be reported.">
                <count_field min_count="0" max_count="255" field_type_unsigned="unsigned byte"/>
                <record name="SpoolIDRec" optional="false">
                  <fixed_field name="SpoolID" optional="false" interpretation="ID of a stored spool" field_type="unsigned byte" field_units="one"/>
                </record>
              </list>
            </sequence>
          </body>
          <footer name="Footer"/>
        </message_def>
        <message_def name="QuerySpoolStatus" message_id="2E05" is_command="false">
          <description xml:space="preserve">This message shall cause the receiving component to reply to the requestor with the ReportSpoolStatus message. An empty QuerySpoolStatusList will result in all known spools being returned.</description>
          <header name="JTS_DefaultHeader">
            <record name="DefaultHeaderRec" optional="false">
              <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>
            </record>
          </header>
          <body name="Body">
            <list name="QuerySpoolStatusList" optional="false" interpretation="A list of spool IDs to report on.  A zero size list indicates that all spools are to be reported.">
<count_field min_count="0" max_count="255" field_type_unsigned="unsigned byte"/>
<record name="SpoolIDRec" optional="false">
<fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>
```

```
</record>

</list>

            </body>

            <footer name="Footer"/>

         </message_def>

      </input_set>

      <output_set>

         <message_def name="CreateSpoolResponse" message_id="4E09" is_command="false">

            <description xml:space="preserve">Response to the 0x0E09 CreateSpool message with the spool ID and success/failure of the action indicated.</description>

            <header name="JTS_DefaultHeader">

               <record name="DefaultHeaderRec" optional="false">

                  <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

               </record>

            </header>

            <body name="Body">

               <record name="CreateSpoolResponseRec" optional="false">

                  <presence_vector field_type_unsigned="unsigned byte"/>

                  <fixed_field name="RequestID" optional="false" interpretation="Locally unique request ID for tracking the status of creating a spool" field_type="unsigned byte" field_units="one"/>

                  <fixed_field name="Response" optional="false" interpretation="Indicates whether the list was successfully created" field_type="unsigned byte" field_units="one">

                     <value_set offset_to_lower_limit="false">

                        <value_enum enum_index="0" enum_const="SpoolCreated"/>

                        <value_enum enum_index="1" enum_const="SpoolNotCreated"/>

                     </value_set>

                  </fixed_field>

<fixed_field name="SpoolID" optional="true" interpretation="Locally unique ID for the saved spool that is assigned if the spool is successfully created" field_type="unsigned byte" field_units="one"/>

               </record>

            </body>

            <footer name="Footer"/>
```

```
</message_def>

<message_def name="DeleteSpoolResponse" message_id="4E0A" is_command="false">

  <description xml:space="preserve">Response to the 0x0E0A DeleteSpool message with the spool ID and success/failure of the action indicated.</description>

  <header name="JTS_DefaultHeader">

    <record name="DefaultHeaderRec" optional="false">

      <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

    </record>

  </header>

  <body name="Body">

    <record name="DeleteSpoolResponseRec" optional="false">

      <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>

      <fixed_field name="Response" optional="false" interpretation="Indicates whether the list was successfully deleted" field_type="unsigned byte" field_units="one">

        <value_set offset_to_lower_limit="false">

          <value_enum enum_index="0" enum_const="SpoolDeleted"/>

          <value_enum enum_index="1" enum_const="SpoolNotDeleted"/>

          <value_enum enum_index="2" enum_const="SpoolDoesNotExist"/>

        </value_set>

      </fixed_field>

    </record>

  </body>

  <footer name="Footer"/>

</message_def>

<message_def name="RunSpoolResponse" message_id="4E0B" is_command="false">

  <description xml:space="preserve">Response to the 0x0E0B RunSpool message with the spool ID and success/failure of the action indicated.</description>

  <header name="JTS_DefaultHeader">

    <record name="DefaultHeaderRec" optional="false">

      <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>

    </record>
```

```
        </header>
        <body name="Body">
          <record name="RunSpoolResponseRec" optional="false">
            <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>
            <fixed_field name="Response" optional="false" interpretation="Indicates whether the list was successfully started" field_type="unsigned byte" field_units="one">
              <value_set offset_to_lower_limit="false">
                <value_enum enum_index="0" enum_const="SpoolStarted"/>
                <value_enum enum_index="1" enum_const="SpoolNotStarted"/>
                <value_enum enum_index="2" enum_const="SpoolDoesNotExist"/>
                <value_enum enum_index="3" enum_const="SpoolAlreadyExecuting"/>
              </value_set>
            </fixed_field>
          </record>
        </body>
        <footer name="Footer"/>
      </message_def>
      <message_def name="AbortSpoolResponse" message_id="4E0C" is_command="false">
        <description xml:space="preserve">Response to the 0x0E0C AbortSpool message with the spool ID and success/failure of the action indicated.</description>
        <header name="JTS_DefaultHeader">
          <record name="DefaultHeaderRec" optional="false">
            <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>
          </record>
        </header>
        <body name="Body">
          <record name="AbortSpoolResponseRec" optional="false">
            <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>
            <fixed_field name="Response" optional="false" interpretation="Indicates whether the spool execution was successfully aborted" field_type="unsigned byte" field_units="one">
```

```
                <value_set offset_to_lower_limit="false">
                  <value_enum enum_index="0" enum_const="SpoolAborted"/>
                  <value_enum enum_index="1" enum_const="SpoolNotAborted"/>
                  <value_enum enum_index="2" enum_const="SpoolNotExecuting"/>
                  <value_enum enum_index="3" enum_const="SpoolDoesNotExist"/>
                </value_set>
              </fixed_field>
            </record>
          </body>
          <footer name="Footer"/>
        </message_def>
        <message_def name="ReportStoredSpools" message_id="4E04" is_command="false">
          <description xml:space="preserve">This message is used to provide the receiver with a list of spools stored in the Mission Spooler.</description>
          <header name="JTS_DefaultHeader">
            <record name="DefaultHeaderRec" optional="false">
              <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>
            </record>
          </header>
          <body name="Body">
            <list name="SpoolList" optional="false">
              <count_field min_count="0" max_count="255" field_type_unsigned="unsigned byte"/>
              <sequence name="SpoolSeq" optional="false">
                <presence_vector field_type_unsigned="unsigned byte"/>
                <record name="SpoolInfoRec" optional="false">
                  <fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>
                  <variable_length_string name="SpoolName" optional="false" interpretation="Human readable name for this spool">
                    <count_field min_count="0" max_count="255" field_type_unsigned="unsigned byte"/>
                  </variable_length_string>
```

```
            </record>
            <list name="TaskList" optional="true" interpretation="List of Tasks for this spool">
               <count_field min_count="0" max_count="65535" field_type_unsigned="unsigned short integer"/>
               <record name="TaskRec" optional="false">
                  <presence_vector field_type_unsigned="unsigned byte"/>
                  <bit_field name="DestinationID" optional="false" interpretation="Identifier of destination to send the message" field_type_unsigned="unsigned integer">
                     <sub_field name="ComponentID" interpretation="Component ID where a value of 255 represents all components.">
                        <bit_range from_index="0" to_index="7"/>
                        <value_set offset_to_lower_limit="false">
                           <value_range lower_limit="1" lower_limit_type="inclusive" upper_limit="255" upper_limit_type="inclusive"/>
                        </value_set>
                     </sub_field>
                     <sub_field name="NodeID" interpretation="Node ID where a value of 255 represents all nodes.">
                        <bit_range from_index="8" to_index="15"/>
                        <value_set offset_to_lower_limit="false">
                           <value_range lower_limit="1" lower_limit_type="inclusive" upper_limit="255" upper_limit_type="inclusive"/>
                        </value_set>
                     </sub_field>
                     <sub_field name="SubsystemID" interpretation="Subsystem ID, where a value of 65535 represents all subsystems">
                        <bit_range from_index="16" to_index="31"/>
                        <value_set offset_to_lower_limit="false">
                           <value_range lower_limit="1" lower_limit_type="inclusive" upper_limit="65535" upper_limit_type="inclusive"/>
                        </value_set>
                     </sub_field>
                  </bit_field>
                  <variable_length_field name="message" optional="false" interpretation="The JAUS message to be sent by the spooler." field_format="JAUS MESSAGE">
```

```
            <count_field min_count="2" max_count="2147483647" field_type_unsigned="unsigned integer"/>

        </variable_length_field>

        <bit_field name="TaskParameters" optional="false" interpretation="Parameters dealing with the execution of this message including blocking and Access Control and Management information" field_type_unsigned="unsigned byte">

            <sub_field name="Blocking" interpretation="Whether or not the message is blocking.">

                <bit_range from_index="0" to_index="0"/>

                <value_set offset_to_lower_limit="false">

                    <value_enum enum_index="0" enum_const="False"/>

                    <value_enum enum_index="1" enum_const="True"/>

                </value_set>

            </sub_field>

            <sub_field name="ControlledNeeded" interpretation="Whether or not the message requires the target component to be in Access Control Controlled state.">

                <bit_range from_index="1" to_index="1"/>

                <value_set offset_to_lower_limit="false">

                    <value_enum enum_index="0" enum_const="False"/>

                    <value_enum enum_index="1" enum_const="True"/>

                </value_set>

            </sub_field>

            <sub_field name="ReadyNeeded" interpretation="Whether or not the message requires the target component to be in Management Ready state.">

                <bit_range from_index="2" to_index="2"/>

                <value_set offset_to_lower_limit="false">

                    <value_enum enum_index="0" enum_const="False"/>

                    <value_enum enum_index="1" enum_const="True"/>

                </value_set>

            </sub_field>

            <sub_field name="MaintainControlAfterTaskCompletes" interpretation="Whether or not control of the target component should be maintained after the message is acted on.">

                <bit_range from_index="3" to_index="3"/>

                <value_set offset_to_lower_limit="false">
```

```
                    <value_enum enum_index="0" enum_const="False"/>
                    <value_enum enum_index="1" enum_const="True"/>
                  </value_set>
                </sub_field>
              </bit_field>
              <fixed_field name="Duration" optional="true" interpretation="Maximum allowed duration of a blocking message, assumed to be the maximum if not specified. When this time is exceeded, the blocking message times out and the spool will be aborted." field_type="unsigned integer" field_units="second"/>
              <fixed_field name="Wait" optional="true" interpretation="Wait time between the completion of this task and the beginning of the next in ms." field_type="float" field_units="second"/>
            </record>
          </list>
        </sequence>
      </list>
    </body>
    <footer name="Footer"/>
  </message_def>
  <message_def name="ReportSpoolStatus" message_id="4E05" is_command="false">
    <description xml:space="preserve">This message is used to provide the receiver a list containing the status of each queried spool and well as the status of its current task. The available spool statuses are: "DoesNotExist" for an ID that does not have a spool associated with it; "Created" for a spool that has been created but not run; "Running" for a spool being actively executed; "Finished" for a spool that has run to completion; and "Aborted" for a spool that has aborted its execution before finishing. The available task statuses are: "NotCreated" for when a spool with the given ID has not been created; "Created" for a task that has been created but not run; "Aborted" for a task that was aborted or failed before finishing; "ControlPending" for a task attempting to gain access control; "ReadyPending" for a task attempting to set the Management::Ready state; ‘Blocking’ for a task waiting for a blocking command event to finish execution; ReleaseControlPending for a task attempting to release access control; "Waiting" for a task that is currently waiting the specified period before finishing; and "Finished" for a task that has successfully run to completion.</description>
    <header name="JTS_DefaultHeader">
      <record name="DefaultHeaderRec" optional="false">
        <fixed_field name="MessageID" optional="false" field_type="unsigned short integer" field_units="one"/>
      </record>
    </header>
    <body name="Body">
      <list name="SpoolStatusList" optional="false">
        <count_field min_count="0" max_count="255" field_type_unsigned="unsigned byte"/>
```

```
<record name="SpoolStatusRec" optional="false">

<fixed_field name="SpoolID" optional="false" interpretation="ID of the saved spool" field_type="unsigned byte" field_units="one"/>

<fixed_field name="CurrentSpoolIndex" optional="false" interpretation=" The index of the task in the task list that corresponds to either the currently executing task when the spool is running, the task on which the execution was aborted, or the first task for both finished and not-started executions." field_type="unsigned byte" field_units="one"/>

<fixed_field name="SpoolStatus" optional="false" field_type="unsigned byte" field_units="one">

<value_set offset_to_lower_limit="false">

<value_enum enum_index="0" enum_const="DoesNotExist"/>

<value_enum enum_index="1" enum_const="Created"/>

<value_enum enum_index="2" enum_const="Running"/>

<value_enum enum_index="3" enum_const="Finished"/>

<value_enum enum_index="4" enum_const="Aborted"/>

</value_set>

</fixed_field>

<fixed_field name="TaskStatus" optional="false" field_type="unsigned byte" field_units="one">

<value_set offset_to_lower_limit="false">

<value_enum enum_index="0" enum_const="NotCreated"/>

<value_enum enum_index="1" enum_const="Created"/>

<value_enum enum_index="2" enum_const="Aborted"/>

<value_enum enum_index="3" enum_const="ControlPending"/>

<value_enum enum_index="4" enum_const="ReadyPending"/>

<value_enum enum_index="5" enum_const="Blocking"/>

<value_enum enum_index="6" enum_const="ReleaseControlPending"/>

<value_enum enum_index="7" enum_const="Waiting"/>

<value_enum enum_index="8" enum_const="Finished"/>

</value_set>

</fixed_field>

</record>

        </list>

</body>
```

```
            <footer name="Footer"/>
          </message_def>
        </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>
                  <!-- A -->
                  <transition name="accessControl.events.transport.Receive">
                    <parameter type="QueryStoredSpools" value="msg" interpretation="enveloped Query Stored Spools
message"/>
                    <parameter type="Receive.Body.ReceiveRec" value="transportData" interpretation="transportData"/>
                    <simple/>
                    <action name="sendReportStoredSpools" interpretation="Send a ReportStoredSpools message to
requesting client.">
                      <argument value="msg"/>
                      <argument value="transportData"/>
                    </action>
                  </transition>
                  <transition name="accessControl.events.transport.Receive">
                    <parameter type="Receive.Body.ReceiveRec" value="transportData" interpretation="transportData"/>
                    <parameter type="QuerySpoolStatus.Body.SpoolIDRec" value="spoolID" interpretation="ID of the
spool from the Query Spool Status message"/>
                    <simple/>
                    <action name="sendReportSpoolStatus" interpretation="Send a ReportSpoolStatus message to
requesting client.">
```