



PHARMACEUTICAL SCIENCES
**PFIZER GLOBAL RESEARCH &
DEVELOPMENT**
SANDWICH

Interim
ASTM LECIS Implementers' Guide
- for comment -

<u>Issue</u>	<u>Date</u>	<u>Author(s)</u>	<u>Reviewer</u>
3.0	08/07/2002	Dorothe Steidinger	Sam Sydenham (Project Manager, STR) Mike Duncan (Head of Technology Development) Graham Robinson (technical review) Tony Parsons

We wish to thank Valery Tkachenko (ACD) and Daniel Derry (ACD) for their contributions to this document.

Document Revision History

Issue	Date	Author(s)	Comments
1.0	14 th June 1999	Colin Read & Jon Wallis	Initial release covering: TSC and SLM power up sequences; control command and data event exchange; example message flow for balance.
2.0	19 th January 2000	Colin Read & Jon Wallis	Added: core requirements; TCP/IP communications; DCD; XML DTD for LECIS DCD; example XML document for LECIS balance DCD.
2.01	3 rd February 2000	Colin Read & Jon Wallis	Minor content and formatting changes after document review.
2.02	15 th March 2000	Colin Read & Jon Wallis	Added: state changes; TCP packet example; message dialogue flow diagrams. Modified: document format; XML DTD for LECIS DCD description.
2.03	29 th March 2000	Colin Read & Jon Wallis	Added: message recovery. Modified: TCP packet example; session/interaction identifiers. Minor content and formatting changes after document review.
2.04	4 th April 2000	Colin Read & Jon Wallis	Minor content and formatting changes after document review.
2.05	12 th January 2001	Colin Read	More detail added to session/interaction identifier generation, message communications recovery and DCD values.
2.06.3	18/02/2002	Dorothe Steidinger	Generally graphs and tables updated, clarifications in descriptions, e.g. use of session identifiers, and additional examples added DCD + example updated, corresponding description in main document added/updated. ITEM_AVAILABLE notification is optional. Additional descriptions, e.g. for NACK, X_DENIED added NEXTEVENT handling changed Synchronous alarm state introduced. Error handling graphs simplified. CLEAR, LOCAL_CTRL_REQ abort interactions 'inventory' removed from STATUS_REQ, replaced with references to DCD sections. 'item list' removed from RUN_OP. Syntax of every message specified. Behaviour for moving to local control specified.

			Standard error codes listed/changed Synchronisation after loss of connection specified. Syntax: special characters, use of quotes clarified.
2.06.4	21/02/2002	Dorothe Steidinger	Changes after review by ACD: Figure 1: Control Flow States And Messages Initiating State Transitions STATE_CHANGED (abort, terminated) removed. Table 9. Synchronous Alarm Message Exchange TERMINATED (1) moved from row 4 to row 6 TERMINATED (2) moved from row 25 to 26 rows 29 to 32 removed; interaction index updated accordingly Table 10. Command Failure Message Exchange - asynchronous alarm TERMINATED (2) moved from row 19 to 20 rows 27 to 30 removed 2.3.3.1 2.3.3.1 Command syntax: bullet 3: ascii reference added for '(' and ')' Table 11. Syntax of TSC to SLM Commands MAINTENANCE, SYSTEM_VARIABLE, RESOURCE added to STATUS_REQ syntax
2.06.5	28/03/2002	Dorothe Steidinger	Syntax of RUN_OP changed: no brackets for commands that do not take arguments.
2.06.6	18/04/2002	Dorothe Steidinger	Comments removed from Table13, last row. Description of TSC's port_id (Table 28) corrected. Recognition of VT and DD added. <i>Changes after review:</i> Captions and headings converted to Title Case (SS). Borders formatted uniformly (SS) Various omissions/punctuation errors corrected (SS) Title changed to reference ASTM version of the standard (SS).
2.06.7	22/05/2002	Dorothe Steidinger	Various omissions/punctuation/formatting errors corrected (MD). Title changed from 'ASTM LECIS Implementers' Guide' to clarify purpose of document (GCR).
3.0	08/07/2002	D. Steidinger	For publication on www.lecis.org

Table Of Contents

1	<i>Introduction</i>	6
1.1	Referenced Documents:	6
2	<i>LECIS Message And State Flow</i>	7
2.1	Core Message Requirements	7
2.2	State Changes	9
2.2.1	Commands	13
2.2.1.1	The NEXTEVENT Command	13
2.2.1.2	Lock/Unlock	14
2.2.1.3	Processing Completion	15
2.2.1.4	The Status Enquiry	15
2.2.1.5	Local Control	16
2.2.2	Events	16
2.2.2.1	Alarms	16
2.2.3	Message Acknowledgement, Message Rejection	21
2.2.3.1	ACK Messages	21
2.2.3.2	NACK Messages	21
2.2.3.3	Message Denied	21
2.3	Communications	21
2.3.1	Synchronisation After Loss Of Connection	22
2.3.2	Time Synchronisation	22
2.3.3	Packet Data	22
2.3.3.1	Command Syntax	22
2.3.3.2	Null Values	28
2.3.4	Session And Interaction Identifiers	29
2.4	Initialisation	33
2.5	Communications Recovery	36
2.5.1	Message Recovery	38
2.5.1.1	Communications Check	38
2.5.1.2	Message Communications Check	39
2.6	Error Conditions	41
2.6.1	Emergency Stop	41
2.6.2	Non-Emergency Conditions	42
3	<i>Device Capability Dataset</i>	44
3.1	Core DCD Requirements	44
3.2	DCD Constraints	45
3.2.1	Definitions	45
3.2.2	Data Type Values	45
3.2.3	Default Values	45
3.2.4	lower_limit And upper_limit Values	45
3.2.4.1	measure_unit Attribute Values	46
3.2.5	TCP/IP Port Definitions	46
3.2.6	Recovery Commands	47
3.2.7	Commands	47
3.2.8	Other Requirements	47
4	<i>Appendix 1: DTD</i>	48
5	<i>Appendix 2: Example DCD-XML file</i>	54

Table Of Tables

Table 1. TSC To SLM Commands.....	7
Table 2. TSC Responses To SLM Event	7
Table 3. Unsolicited SLM To TSC Events	9
Table 4. SLM Responses To TSC Commands.....	9
Table 5. State Changes Concurrent With Command Acknowledgement.....	10
Table 6. State Changes That Must Be Announced With STATE_CHANGED.....	10
Table 7. Control Message Exchange	14
Table 8: Standard Error Codes	16
Table 9. Synchronous Alarm Message Exchange	17
Table 10. Command Failure Message Exchange – Asynchronous Alarm.....	20
Table 11. Syntax Of TSC To SLM Commands.....	23
Table 12. Syntax Of TSC Responses To SLM Event	24
Table 13. Syntax Of Acknowledgement Messages	25
Table 14. Syntax Of SLM To TSC Events.....	26
Table 15. Example TCP Packets	28
Table 16. Parent Session Identifiers.....	30
Table 17. Primary Session Interaction Identifiers.....	31
Table 18. TSC Secondary Session Interaction Identifiers.....	31
Table 19. Initialisation Message Exchange	33
Table 20. SLM Verifies Connection #1	37
Table 21. SLM Verifies Connection #2.....	37
Table 22. TSC Emergency Stop	41
Table 23. SLM Emergency Stop.....	41
Table 24. SLM Non-Emergency Conditions	42
Table 25. DCD Sections.....	44
Table 26. data_type_value Attribute Values In XML DTD	45
Table 27. SLM TCP/IP Port Values In XML DCD	46
Table 28. TSC TCP/IP port Values in XML DCD.....	46

Table Of Figures

Figure 1: Control Flow States And Messages Initiating State Transitions	11
Figure 2: Local/Remote Control States And Messages Initiating State Transitions	13
Figure 3: Multiple Synchronous Alarms	19
Figure 4. Example TSC TCP Packet Syntax	23
Figure 5. Example TSC TCP Packets With Data.....	23
Figure 6. Example Acknowledgement Tcp Packet Syntax.....	23
Figure 7. Example SLM TCP Packet Syntax.....	23
Figure 8. Example SLM TCP Packets With Data.....	23
Figure 9. Example SLM TCP Packets Omitting Data	23
Figure 10. Example TSC TCP Packet With Optional Characters.....	29
Figure 11. Example Date And Time Stamp With Sequence Number	29
Figure 12. State Diagram With Session/Interaction Identifiers.....	32
Figure 13. Initialisation Process Flow	35

Figure 14 Message Acknowledgement Process Flow	39
Figure 15. Message Process Flow	40

1 Introduction

The intended audience for this document is experienced systems integrators who are incorporating LECIS compliance into new or existing systems.

This document should be read in conjunction with the current LECIS¹ (Laboratory Equipment Control Interface Specification) documentation. It interprets and distils some of the key information contained within the LECIS document. It is intended to be used as a 'jump start' rather than wading through the documents in their entirety with no previous exposure to the specification. Wherever there is an ambiguity or discrepancy with the LECIS document, notably with the following points, the information contained herein will take precedence:

- TCP/IP as protocol of choice, definition of package
- Next Event as interaction that does not require unique interaction id. Allowed interaction in Local Control
- synchronous alarms
- error handling
- Abort interaction is allowed when SLM is paused.
- Item Available Notification is optional and must be defined in the DCD if implemented.
- An SLM may have active interactions with any number of TSCs at the same time.
- State changes to TERMINATED in secondary interactions that are aborted (implicitly or explicitly) do not need to be announced.

The following sections expand upon areas that deal with: -

- Core requirements
- TCP/IP communications, including message syntax
- Task Sequence Controller (TSC) and Standard Laboratory Module (SLM) power up sequences
- Control command and data event exchange
- Off-normal conditions
- Device Capability Dataset (DCD)
- Example implementation

Within the scope of this document the 'states' shown in the tables are held internally by the SLM and the TSC and should be synchronised.

1.1 Referenced Documents:

1. ASTM Standard (E 1989-98) on Laboratory Equipment Control Interface Specification, American Society for Testing and Materials, West Conshohocken, PA, 1999
2. Initial CAALS Device Capability Dataset, Version 1.0.7
National Institute of Standards and Technology. Chemical Science and Technology Laboratory, Analytical Chemistry Division, Gaithersburg, Maryland 20899, USA

2 LECIS Message And State Flow

2.1 Core Message Requirements

For a TSC and SLM to be LECIS compliant they must support the REQUIRED primary and secondary state models and the REQUIRED commands and events (except for the interaction Item Available Notification, which should be implemented as custom event if it is required for the operation of an SLM). All commands and events exchanged between a TSC and an SLM must be case insensitive. That is, upper case, lower case and mixed case are interpreted in the same way. A full explanation of the syntax of these commands and events can be found in the LECIS document.

The LECIS commands that are initiated by the TSC and sent to an SLM are listed in Table 1.

Table 1. TSC To SLM Commands

Command	Purpose
REMOTE_CTRL_REQ	Request remote control of SLM
LOCAL_CTRL_REQ	Request SLM to take local control
INIT	Initialises remote SLM
SETUP	Configure remote SLM for normal operation
CLEAR	Instructs a remote SLM to return to IDLE state
LOCK_REQ	Changes remote SLM port state to locked
UNLOCK_REQ	Unlocks a port at the remote SLM
RUN_OP(<i>command</i>)	Remote SLM runs the command " <i>command</i> "
STATUS_REQ	Remote SLM responds with the current status report
ESTOP	Instructs a remote SLM to perform an emergency stop
PAUSE	Instructs a remote SLM to suspend operations after the currently executing command has finished
RESUME	Instructs a remote SLM to resume normal operations and execute the next command (if any)
ABORT_REQ	Instructs a remote SLM to stop processing the current command
NEXTEVENT	Clearance given to the SLM to talk to the TSC

The responses by the TSC to LECIS commands that are initiated by an SLM are listed in Table 2.

Table 2. TSC Responses To SLM Event

Event Response	Purpose
REMOTE_CTRL_DENIED	Remote control request from SLM rejected
REMOTE_CTRL_GRANTED	Remote control request from SLM accepted
LOCAL_CTRL_DENIED	Local control request from SLM refused
LOCAL_CTRL_GRANTED	Local control request from SLM granted

The events that can be sent asynchronously during normal operation by an SLM to the TSC are listed in Table 3.

Table 3. Unsolicited SLM To TSC Events

Events	Purpose
LOCAL_CTRL_REQ	Request local control from SLM
REMOTE_CTRL_REQ	Request TSC to take remote control
STATE_CHANGED	Informs the TSC of a change of state that the TSC may or may not have initiated
ALARM_ON	Informs the TSC that an asynchronous abnormal event has occurred
ALARM_OFF	Informs the TSC that an asynchronous abnormal event has been resolved

The messages that are sent by an SLM in response to commands initiated by the TSC are listed in Table 4.

Table 4. SLM Responses To TSC Commands

Command Responses	Purpose
REMOTE_CTRL_DENIED	Remote control request rejected by SLM
REMOTE_CTRL_ACCEPTED	Remote control granted to TSC
LOCAL_CTRL_DENIED	Local control request from TSC rejected
LOCAL_CTRL_ACCEPTED	Local control accepted by SLM
OP_DENIED	Sent when a RUN_OP request has been rejected
OP_STARTED	Sent when a RUN_OP request has started to be processed
ALARM_ON	Informs the TSC that a synchronous abnormal event has occurred
ALARM_OFF	Informs the TSC that a synchronous abnormal event has been resolved
OP_RESULT(<i>result</i>)	Returns any processing “ <i>result</i> ” from a RUN_OP to the TSC
OP_COMPLETED	Sent when a RUN_OP has finished
LOCK_DENIED	Informs the TSC that a LOCK request has been rejected
LOCK_ACCEPTED	Informs the TSC that a LOCK request has been granted
LOCKED	Informs the TSC that a LOCK request has been successful
UNLOCKED	Informs the TSC that an UNLOCK request has been successful
STATUS	Returns status information to the TSC
NO_STATUS	Informs the TSC that there is no status information available
ABORT_ACCEPTED	Sent when an ABORT request is granted
ABORT_DENIED	Sent when an ABORT request is rejected
ABORT_COMPLETED	Sent when an ABORT request is successful
STATE_CHANGED	Informs the TSC of a change of state that the TSC may or may not have initiated

2.2 State Changes

The TSC and SLM maintain internal models of both primary and secondary states; the primary states are persistent throughout secondary state operations.

State changes in the SLM that do not have a predefined event report are reported to the TSC with a STATE_CHANGED message. State changes as direct response to a command are implied by the acknowledgement of the command that initiates a state transition and are not reported to the TSC.

The state changes that are concurrent with command acknowledgement are listed in Table 5, those that must be announced with a STATE_CHANGED message by an SLM are listed in Table 6.

Table 5. State Changes Concurrent With Command Acknowledgement

TSC Command	Old SLM State	New SLM State
REMOTE_CTRL_REQ	LOCAL	REMOTE CTRL REQUESTED
LOCAL_CTRL_REQ	REMOTE	LOCAL CTRL REQUESTED
INIT	POWERED UP	INITING
SETUP	IDLE	CONFIGURING
PAUSE	CONTROL FLOW	PAUSING
CLEAR	NORMAL OPERATION	CLEARING
LOCK*		LOCK REQUESTED
UNLOCK_REQ	LOCKED	UNLOCKING
RUN_OP*		PROCESSING REQUESTED
ABORT_REQ*		ABORT REQUESTED
STATUS*		STATUS REQUESTED
NEXTEVENT*		NEXT EVENT REQUESTED
(event report to TSC)	NEXTEVENT REQUESTED	TERMINATED
ESTOP	any substate of any active secondary interaction	TERMINATED
ESTOP	any substate of Local/Remote Control	LOCAL

* The command initiates a new interaction and therefore has no “old SLM state”.

Table 6. State Changes That Must Be Announced With STATE_CHANGED

Old SLM State	New SLM State	Comment
None after complete reset or first ever connection of an SLM, or ESTOPPED.	POWERED UP	Conceptually the SLM does not undergo a state transition after having been power cycled. If an SLM connects to the TSC but does not send any event reports the TSC should query the status of the Control Flow interaction and update its state model accordingly.
INITING	IDLE	
CONFIGURING	NORMAL OPERATION	
any state	ESTOPPED	if estop was initiated by SLM
any state	PAUSING	if pause was initiated by SLM

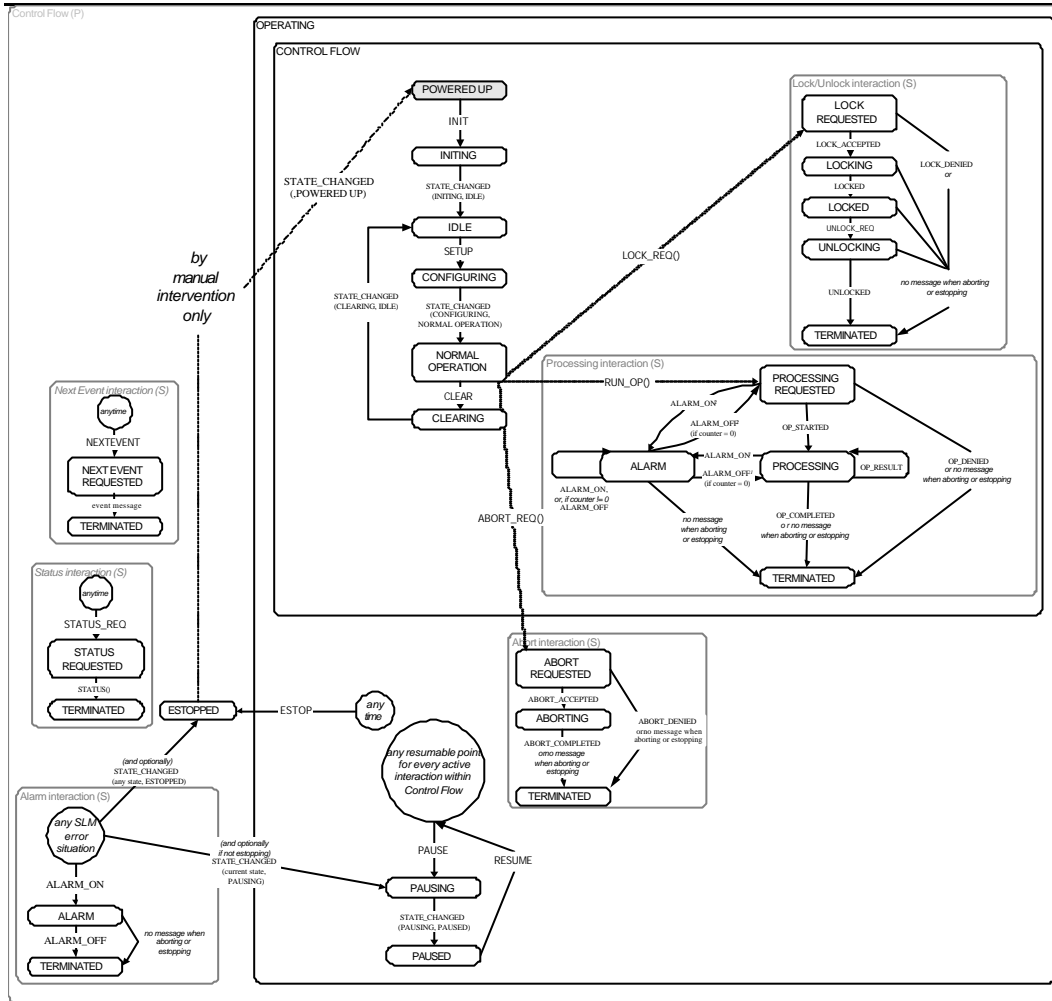
PAUSING	PAUSED	
---------	--------	--

Interim states, such as LOCK_REQUESTED, PROCESSING_REQUESTED, ABORT_REQUESTED, LOCAL_CTRL_REQUESTED and REMOTE_CTRL_REQUESTED, allow an SLM to complete currently active processes where appropriate.

For example, if the SLM is in LOCAL processing a command sent from its user interface and the user has not 'locked' the SLM (in which case it would send REMOTE_CTRL_DENIED ("user override") when it receives a REMOTE_CTRL_REQ command from the TSC) the SLM remains in REMOTE_CTRL_REQUESTED until the current process is completed, then sends REMOTE_CTRL_GRANTED announcing its transit to REMOTE.

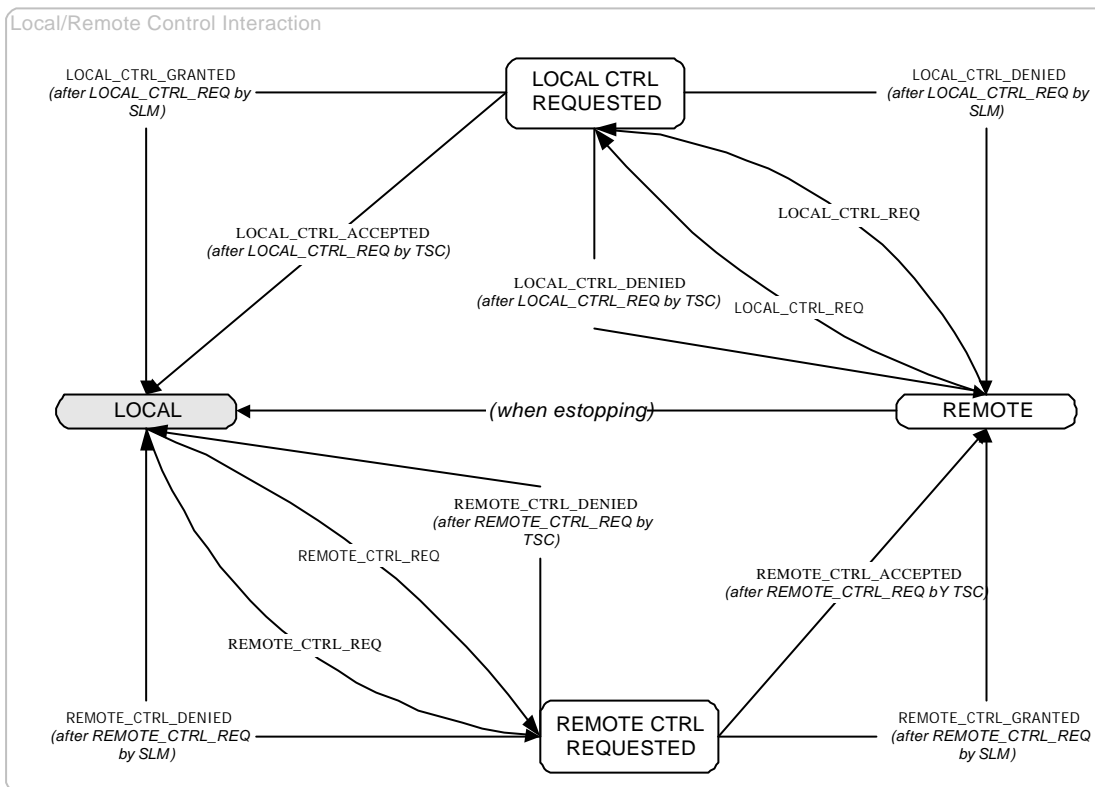
Figure 1 shows the discrete interactions and states within Control Flow with their related messages. The placement of secondary interactions, such as Abort or Status, indicates the availability of these interactions. E.g, a command can be aborted when the SLM is paused, and the SLM can notify the TSC about abnormal conditions with an alarm notification if it has been estopped. The positive acknowledgement required for each state transition is not shown.

Figure 1: Control Flow States And Messages Initiating State Transitions



The Control Flow and the Local/Remote Control interactions exist next to each other, although restrictions apply. E.g. when the SLM is in the state LOCAL it may only accept the commands NEXTEVENT, STATUS_REQ, ESTOP, and – but only if it is not in ESTOPPED at the same time - REMOTE_CTRL_REQ. For all other commands it must be in REMOTE.

Figure 2 shows the Local/Remote Control interaction with its associated messages.

Figure 2: Local/Remote Control States And Messages Initiating State Transitions

2.2.1 Commands

During a remote control session the TSC is able to send up to *command_buffer_size+1* number of commands (RUN_OP, CLEAR, LOCK, or UNLOCK) to the SLM for processing. That is, these commands are sent sequentially from the TSC to an SLM if the SLM's *command_buffer_size* parameter is set to zero and the TSC will always wait for an OP_COMPLETED, STATE_CHANGED, LOCKED or UNLOCKED event report before it will send the next command to the SLM. In any case, both TSC and SLM must wait for positive or negative acknowledgement of a message before the next message can be sent. Before an SLM can send any command response to the TSC it must have received a NEXTEVENT message from the TSC. Other commands are not affected by *command_buffer_size*.

All return values of processing requests are returned via the OP_RESULT message.

When clearing, the SLM terminates all active secondary interactions, i.e. CLEAR implicitly aborts all active processing interactions.

2.2.1.1 The NEXTEVENT Command

The NEXTEVENT message acts as a 'clear to send' signal and is always issued by the TSC to an SLM before the SLM can send it any event or command response. If the TSC can queue incoming messages and therefore does not require NEXTEVENT for message flow control, it should ensure that multiple instances of NEXTEVENT commands are available to the SLM so that SLM-initiated event reports, such as ALARM_ON or STATE_CHANGED (ESTOPPED), are not unnecessarily delayed. As a minimum, after each message sent to the TSC from an SLM the TSC must send a NEXTEVENT message

back to the SLM to give it permission to send another message when one becomes available. As with any other interaction, NEXTEVENT is not terminated if an event report from the SLM is NACKed by the TSC and the SLM does not have to wait for another NEXTEVENT before it can re-send the report. It is however optional for the TSC to create a separate interaction id for each NEXTEVENT interaction and the session identifier can be used instead.

2.2.1.2 Lock/Unlock

The LOCK_REQ command provides a mechanism for allowing the TSC exclusive access to SLM system resources that have been identified in the DCD as data (i.e. communications) or material (i.e. needs or produces something physical) *ports*. While a port is locked the SLM that owns that port is not allowed to access it. For example, the TSC may lock an HPLC autosampler tray port before instructing another SLM to access it to avoid damage to the hardware.

If the SLM transits to local control following a LOCAL_CTRL_REQUEST the lock status of a port is not affected. The '*working*' command and event message sequence is shown in Table 7.

Table 7. Control Message Exchange

	Message Initiator	SLM Event	Recipient	TSC Command	Response	Primary SLM State	New Secondary SLM State
	TSC		SLM1	NEXTEVENT			
	SLM1		TSC		ACK	NORMAL OPERATION (REMOTE)	
	TSC		SLM2	NEXTEVENT			
	SLM2		TSC		ACK	NORMAL OPERATION (REMOTE)	
	TSC		SLM1	LOCK_REQ			
	SLM1		TSC		ACK		LOCK REQUESTED
	SLM1	LOCK_ACCEPTED	TSC				
	TSC		SLM1		ACK		LOCKING PORT
	TSC		SLM1	NEXTEVENT			
	SLM1		TSC		ACK		
	SLM1	LOCKED	TSC				
	TSC		SLM1		ACK		LOCKED
	TSC		SLM1	NEXTEVENT			
	SLM1		TSC		ACK		
	TSC		SLM2	RUN_OP			
	SLM2		TSC		ACK		PROCESSING REQUESTED

	SLM2	OP_STARTED	TSC				
	TSC		SLM2		ACK		PROCESSING
	TSC		SLM2	NEXTEVENT			
	SLM2		TSC		ACK		
	SLM2	OP_RESULT	TSC				
	TSC		SLM2		ACK		PROCESSING
	TSC		SLM2	NEXTEVENT			
	SLM2		TSC		ACK		
	SLM2	OP_COMPLETED	TSC				
	TSC		SLM2		ACK		TERMINATED
	TSC		SLM2	NEXTEVENT			
	SLM2		TSC		ACK		
	TSC		SLM1	UNLOCK_REQ			
	SLM1		TSC		ACK		UNLOCKING
	SLM1	UNLOCKED	TSC				
	TSC		SLM1		ACK		TERMINATED
	TSC		SLM1	NEXTEVENT			
31	SLM1		TSC		ACK		

2.2.1.3 Processing Completion

The receipt of an OP_COMPLETED message by a TSC indicates successful completion of a processing request by an SLM. In the case of data being returned by an SLM, the receipt of OP_RESULT is not a sufficient indication of success until an OP_COMPLETED message is received by the TSC.

If a command fails on an SLM after it has sent an OP_STARTED message to the TSC the SLM will send an ALARM_ON message to the TSC. This message sequence is illustrated in Table 9 and Table 10.

2.2.1.4 The Status Enquiry

The Status interaction allows the TSC any time to request information from an SLM.

If the command is sent with only the object type (i.e. interaction, maintenance, system_variable, port, alarm, etc.) as parameter, the SLM should respond with status information about any active/available item. E.g. having received the command STATUS_REQ (interaction) the SLM would send status information about all active interactions, including primary interactions and Alarm interactions. Since the acknowledgement of the SLM's status event report marks the transit of that Status interaction to TERMINATED, the Status interaction itself is not listed in the SLM's message.

Unlike the TSC command/SLM command response sequence SLM events can happen asynchronously. As with command responses, the SLM can not send an event to the TSC without having received a NEXTEVENT message from the TSC.

The *EventDateTime* is a date and time stamp string, (yyyymmddhhmmssss), identifying when the corresponding event was generated on the SLM.

2.2.1.5 Local Control

Since an operator may invoke a transit to local control for error recovery, it is left to the operator's judgement whether an SLM must be power-cycled prior to returning it to remote control. Requesting local control implicitly aborts any processing interactions and there is no requirement for the SLM to track any state changes while in local control. To be able to continue an interrupted run, the operator has to ensure that the system is in the state the TSC expected it to be in if the aborted interactions had been completed successfully. If this is not possible – e.g. because return values of a command cannot be communicated to the TSC – the SLM has to be reset to re-enter the Control Flow interaction in POWERED UP.

2.2.2 Events

2.2.2.1 Alarms

Alarms are used by an SLM to notify the TSC of an error during command execution or of a safety-critical event.

Table 8 lists the error codes that are available for SLMs to report alarms, but also as arguments in NACK or OP_DENIED messages. The corresponding messages are defined in an SLM's DCD and the list may be extended by SLM vendors as required.

Table 8: Standard Error Codes

Error code	Error Text	reference to NACK message	Example/Comment
-00001	invalid state	INVALID_STATE	e.g.: SETUP is sent when SLM is in POWERED UP
-00002	command not supported	CMD_NOT_SUPPORTED	Command is not listed in DCD.
-00010	general operation failed	GENERAL_OP_FAILED	General error
-00020	invalid configuration	INVALID_CONFIG	e.g. the commands as specified in <required_configurations> have not been sent
-00030	invalid command format	INVALID_CMD_FORMAT	The format of the message does not follow specified syntax
-00032	invalid command argument	INVALID_ARG	

-00033	command argument out of range	ARG_OUT_OF_RANGE	Value of argument is not between lower and upper limits specified in DCD
-00035	invalid number of command arguments	INVALID_NO_OF_ARGS	Number of arguments does not match that specified in DCD.
-00037	invalid data	INVALID_DATA	
-00038	invalid data type	INVALID_DATA_TYPE	Data type of an argument does not match that specified in DCD.

The following codes have been omitted:

- 00034 (missing command argument/MISSING_ARG) – this error is more appropriately addressed by –00035 (invalid number of command arguments)
- 00031 (missing command/MISSING_CMD) – this error is more appropriately addressed by –00020 (invalid configuration) or –00002 (command not supported)
- 00036 (invalid argument separator) – this error is more appropriately addressed by –00030 (invalid command format)

There are two types of alarms: synchronous and asynchronous alarms.

2.2.2.1.1 Synchronous Alarms

Synchronous alarms indicate an error during the execution of a command, which makes the goal of a command unachievable or affects the integrity of its result. An example for a synchronous alarm would be the failure of a robotic arm to move to a position following a MoveToPositionX type command. The syntax of the event reports ALARM_ON and ALARM_OFF is not changed by this specification, however, ALARM becomes a substate of the originating interaction, i.e. no new interaction identifier is created; the event report is associated with the interaction identifier of the originating interaction. As such, ALARM_OFF must be sent before OP_COMPLETED, and the alarm is terminated with the command interaction if the command is aborted.

Table 9 shows the message exchange when the SLM detects an error that prevents it from completing the processing of an active command. Here, the TSC verifies the alarm status, aborts the command, then verifies the alarm status again.

Table 9. Synchronous Alarm Message Exchange

	from	SLM Event	TSC Command	Response	New Secondary SLM State (interaction index)	Comment
1.	TSC		NEXTEVENT			
2.	SLM			ACK	NEXTEVENT REQUESTED (1)	
3.	TSC		RUN_OP			
4.	SLM			ACK	PROCESSING	

					REQUESTED (2),	
5.	SLM	OP_STARTED				
6.	TSC			ACK	PROCESSING (2) TERMINATED (1)	
7.	TSC		NEXTEVENT			
8.	SLM			ACK	NEXTEVENT REQUESTED (3)	
9.	SLM	ALARM_ON				The SLM fails to achieve its goal.
10.	TSC			ACK	ALARM(2), TERMINATED (3)	
11.	TSC		NEXTEVENT			
12.	SLM			ACK	NEXTEVENT REQUESTED (4)	
13.	TSC		STATUS_REQ (INTERACTION) or STATUS_REQ (ALARM)			The TSC can query the status of alarms or of the original interaction
14.	SLM			ACK	STATUS REQUESTED (5)	
15.	SLM	STATUS (FAILED) or STATUS (interaction id)				The SLM responds according to the TSC's message.
16.	TSC			ACK	TERMINATED (5), TERMINATED (4)	
17.	TSC		NEXTEVENT			
18.	SLM			ACK	NEXTEVENT REQUESTED (6)	
19.	TSC		ABORT_REQ			
20.	SLM			ACK	ABORT REQUESTED (7)	
21.	SLM	ABORT_ACCEPTED				
22.	TSC			ACK	ABORTING (7), TERMINATED (6)	
23.	TSC		NEXTEVENT			
24.	SLM			ACK	NEXTEVENT REQUESTED (8)	
25.	SLM	ABORT_COMPLETED				
26.	TSC			ACK	TERMINATED (2), TERMINATED (7), TERMINATED (8)	
27.	TSC		NEXTEVENT			
28.	SLM			ACK	NEXTEVENT REQUESTED (9)	
29.	TSC		STATUS_REQ (INTERACTION) or STATUS_REQ (ALARM)			
30.	SLM			ACK	STATUS REQUESTED (10)	
31.	SLM	STATUS (ABORTED) or NO_STATUS				The alarm was terminated with the command.

32.	TSC			ACK	TERMINATED (9), TERMINATED (10)	
-----	-----	--	--	-----	------------------------------------	--

The following example shows a situation where two synchronous alarms occur during the processing of a command:

Figure 3: Multiple Synchronous Alarms

```
// SLM in Normal Operation
<- 2001082909221803<CR>NEXTEVENT<CR><LF>
--> 2001082909221803<CR>ACK<CR><LF>
<- 2001082909224761_3<CR>RUN_OP("ToBalance1"<CR><CR>)<CR><LF>
--> 2001082909224761_3<CR>ACK<CR><LF>
// Processing Requested for 2001082909224761_3
--> 2001082909224761_3<CR>2001082909224899<CR>OP_STARTED<CR><LF>
<- 2001082909224761_3<CR>ACK<CR><LF>
// Processing for 2001082909224761_3
<- 2001082909224761<CR>NEXTEVENT<CR><LF>
--> 2001082909224761<CR>ACK<CR><LF>
// SLM detects that it can't move to that position
--> 2001082909224761_3<CR>2001082909224900<CR>ALARM_ON (-10001<CR>"x ACHSIS OUT OF
RANGE"<CR><CR>)<CR><LF>
<- 2001082909224761_3<CR>ACK<CR><LF>
// Alarm for 2001082909224761_3, counter = 1
// SLM detects that it lost communication with encoder
<- 2001082909224761<CR>NEXTEVENT<CR><LF>
--> 2001082909224761<CR>ACK<CR><LF>
--> 2001082909224761_3<CR>2001082909224901<CR>ALARM_ON (-10002<CR>"Encoder communication
problem"<CR><CR>)<CR><LF>
<- 2001082909224761_3<CR>ACK<CR><LF>
// Alarm for 2001082909224761_3, counter = 2
<- 2001082909224761<CR>NEXTEVENT<CR><LF>
--> 2001082909224761<CR>ACK<CR><LF>
// Communication with encoder re-established
--> 2001082909224761_3<CR>2001082909224902<CR>ALARM_OFF (-10002<CR><CR>)<CR><LF>
<- 2001082909224761_3<CR>ACK<CR><LF>
// Alarm for 2001082909224761_3, counter = 1
// SLM managed to move to position
<- 2001082909224761<CR>NEXTEVENT<CR><LF>
--> 2001082909224761<CR>ACK<CR><LF>
--> 2001082909224761_3<CR>2001082909224903<CR>ALARM_OFF (-10001<CR><CR>)<CR><LF>
<- 2001082909224761_3<CR>ACK<CR><LF>
// Processing for 2001082909224761_3, Alarm counter = 0
<- 2001082909224761<CR>NEXTEVENT<CR><LF>
--> 2001082909224761<CR>ACK<CR><LF>
--> 2001082909224761_3<CR>2001082909224905<CR>OP_COMPLETED<CR><LF>
<- 2001082909224761_3<CR>ACK<CR><LF>
```

2.2.2.1.2 Asynchronous Alarms

Asynchronous alarms indicate error situations that are detected independently of active command interactions; an asynchronous alarm would be raised e.g. to notify the TSC that a user has pressed an emergency button to stop the system. Asynchronous alarms are independent interactions with unique interaction ids.

Table 10. Command Failure Message Exchange – Asynchronous Alarm

	Message Initiator	SLM Event	TSC Command	Response	Primary SLM State	New Secondary SLM State (interaction index)
1.	TSC		NEXTEVENT		NORMAL OPERATION (REMOTE)	
2.	SLM			ACK		NEXTEVENT REQUESTED (1)
3.	TSC		RUN_OP			
4.	SLM			ACK		PROCESSING REQUESTED(2)
5.	SLM	OP_STARTED				
6.	TSC			ACK		PROCESSING (2), TERMINATED (1)
7.	TSC		NEXTEVENT			
8.	SLM			ACK		NEXTEVENT REQUESTED (3)
9.	SLM	ALARM_ON				
10.	TSC			ACK		ALARM(4), TERMINATED (3)
11.	TSC		NEXTEVENT			
12.	SLM			ACK		NEXTEVENT REQUESTED (5)
13.	TSC		ABORT_REQ(2)			
14.	SLM			ACK		ABORT REQUESTED (6)
15.	SLM	ABORT_ACCEPTED				
16.	TSC			ACK		ABORTING (6), TERMINATED (5)
17.	TSC		NEXTEVENT			
18.	SLM			ACK		NEXTEVENT REQUESTED (7)
19.	SLM	ABORT_COMPLETED				
20.	TSC			ACK		TERMINATED (2) TERMINATED (6), TERMINATED (7)
21.	TSC		NEXTEVENT			
22.	SLM			ACK		NEXTEVENT REQUESTED (8)
23.	SLM	ALARM_OFF				
24.	TSC			ACK		TERMINATED (4), TERMINATED (8)
25.	TSC		NEXTEVENT			
26.	SLM			ACK		NEXTEVENT REQUESTED (9)

2.2.3 Message Acknowledgement, Message Rejection

Each message, whether sent by the TSC or the SLM, is acknowledged by its recipient. The sender must not send another message until it has received an acknowledgement for the previous message. However, here it does not matter whether the acknowledgement was positive (ACK) or negative (NACK).

2.2.3.1 ACK Messages

ACK messages (sent by TSC or SLM) indicate that the message received was legal, i.e. it has been recognised, was syntactically correct, and allowed for the current SLM state. It also coincides with a state change in the SLM.

2.2.3.2 NACK Messages

NACK messages indicate that the message received has either not been recognised, is syntactically incorrect, or is not allowed in the current SLM state. It also implies that the SLM has not changed the state of an interaction (note that this includes NEXTEVENT). The reason for rejection of a message is included in the negative acknowledgement.

2.2.3.3 Message Denied

If an SLM cannot fulfil a request (RUN_OP, LOCK_REQ, LOCAL or REMOTE_CTRL_REQ, ABORT_REQ), it denies the request indicating the reason for the denial, and – except for the Local/Remote control interaction - changes to TERMINATED.

Possible reasons for denying a request include:

- LOCK_REQ: The specified port is in use or it already is locked.
- RUN_OP: a port required for the operation is locked.
- REMOTE_CTRL_REQ: the Laboratory Module is currently in use in local control
- LOCAL_CTRL_REQ: a processing interaction is currently active.
- ABORT_REQ: a process that cannot be aborted has already been started, or the interaction is already terminated.

2.3 Communications

This section outlines the steps required to establish a communications link between an SLM and the TSC.

All SLMs will be available over a computer network link using TCP/IP between the SLM and the TSC. When establishing low-level communications for the first time the TSC will listen on a known port for a TCP/IP connection from an SLM. When a connection is received it will resolve the IP address and check it against a known SLM list. The TSC will then issue a NEXTEVENT command to that SLM as an acknowledgement to start a LECIS dialogue. The TCP/IP port by which the SLM connects to the TSC should be user configurable on the SLM.

The SLM will hold the TCP/IP link to the TSC open all the time that it is available. Although both the SLM and TSC are required to perform communication checks periodically, it is the responsibility of the SLM to maintain the integrity of the TCP/IP link with the TSC. Accuracy of message transmission is adequately provided by TCP/IP, however both TSC and SLM

must periodically check that the communication link itself is not interrupted. This check can be achieved by means of low-level checks, e.g. *ping*, or by querying the operating system for the TCP/IP socket status. High level checks are not considered adequate means for checking connection integrity as the failure of either party to respond is not necessarily due to a lost connection. It is also the responsibility of the SLM to re-establish communications with the TSC in the event of a problem, (see 2.5).

2.3.1 Synchronisation After Loss Of Connection

To allow synchronisation of states after connection loss, both TSC and SLM should store the last message they sent (other than acknowledgement) until it has been acknowledged, and the last message (other than acknowledgement) received until they receive the next one.

This will allow them to cope with following situations:

1. a message (other than acknowledgement) was sent after the connection was interrupted
2. a message was received, but could not be acknowledged since the connection was interrupted.
3. An acknowledgement was sent after the connection was interrupted.

Once reconnected all buffered messages are sent, starting – if applicable – with the last message for which no acknowledgement was received. Messages received in duplicate are ignored as they can be recognised by their interaction id.

Finally it is recommended that the TSC query the status of the remaining interactions to ensure the state models are fully synchronised.

2.3.2 Time Synchronisation

Time synchronisation between the TSC and all SLMs can be achieved by using a standard time service such as NTP (Network Time Protocol).

2.3.3 Packet Data

The LECIS command and event text within each packet will be Unicode standard compliant. Numbers are represented in character form and in the case of binary data, it will be converted to Unicode characters, (e.g. MIME64), before it is sent to the TSC or SLM. It is then the responsibility of the application software to do the necessary conversion from text to the data type specified in the SLM's DCD.

For consistency and compliance with the original LECIS documentation the parenthesis encapsulating argument lists have been retained.

2.3.3.1 Command Syntax

The following rules apply to TCP packets: -

- Carriage return character codes separate the arguments in each TCP packet.
- TCP packets are terminated with a carriage return and line feed pair.
- Text arguments (arguments having *data_type string* in DCD) are encapsulated in quotes if they contain special characters (i.e. ASCII symbols 13 (carriage return), 10 (line feed), 34 (quote), 40 (opening bracket), 41 (closing bracket), otherwise quotes are optional. Quotes that are part of the data must be preceded by a back-slash to distinguish from

those used to indicate the boundaries of an argument. See Figures 5 and 8 for valid examples.

- Argument lists for commands are enclosed in parenthesis.
- All arguments for a packet must be supplied.
- If an argument has a null value then its place must still be indicated. If a message may take a single argument but does not in a particular instance, the opening and closing brackets must still be supplied.
- No white space is permitted in the packet; except as part of string data item where all characters are treated as data.
- All other data types are not encapsulated in quotes.

A TSC TCP packet is illustrated in Figure 4 and an SLM TCP packet is illustrated in Figure 7.

Figure 4. Example TSC TCP Packet Syntax

DateTime_n<CR>RUN_OP(“DoSomething”[<CR>([“]Argument1[”]){<CR>[“]Argument2[”]}D])<CR>[StartDateTime]<CR><LF>

Figure 5. Example TSC TCP Packets With Data

DateTime_n<CR>RUN_OP(“DoSomething”<CR>(“Textargument1”<CR>NumericArgument2)<CR>StartDateTime)<CR><LF>

DateTime_n<CR>RUN_OP(DoSomething<CR>(“Text<CR>argument1”<CR>TextArgument2<CR>”Text: \”argument\””)<CR>StartDateTime)<CR><LF>

Figure 6. Example Acknowledgement Tcp Packet Syntax

DateTime_n<CR>NACK(“ReasonText”<CR>([“]Argument1[”]){<CR>[“]Argument2[”]}<CR><LF>

Figure 7. Example SLM TCP Packet Syntax

DateTime_n<CR>ResponseDateTime<CR>OP_RESULT([“]Data1[”][<CR>[“]Data2[”]])<CR><LF>

Figure 8. Example SLM TCP Packets With Data

DateTime_n<CR>ResponseDateTime<CR>OP_RESULT(“Data1”<CR>Data2)<CR><LF>

DateTime_n<CR>ResponseDateTime<CR>OP_RESULT(“Data<CR>1:.”text\””<CR>Data2)<CR><LF>

Figure 9. Example SLM TCP Packets Omitting Data

1. Two of two arguments omitted:

DateTime_n<CR>ResponseDateTime<CR>OP_RESULT(<CR>)<CR><LF>

2. First of two arguments omitted:

DateTime_n<CR>ResponseDateTime<CR>OP_RESULT(<CR>Data2)<CR><LF>

3. One of one argument omitted: *DateTime_n*<CR>ResponseDateTime<CR>OP_RESULT()<CR><LF>

Table 11. Syntax Of TSC To SLM Commands

Command	Syntax/Example
REMOTE_CTRL_REQ	<i>DateTime</i> <CR>REMOTE_CTRL_REQ<CR><LF>
	2001102213352870<CR>REMOTE_CTRL_REQ<CR><LF>

LOCAL_CTRL_REQ	<i>DateTime</i> <CR>LOCAL_CTRL_REQ<CR><LF>
	2001102213352870<CR>LOCAL_CTRL_REQ<CR><LF>
INIT	<i>DateTime_x</i> <CR>INIT<CR><LF>
	2001102213352870_1<CR>INIT<CR><LF>
SETUP	<i>DateTime_x</i> <CR>SETUP([<i>Argument</i>])<CR><LF>
	2001102213352870_1<CR>SETUP()<CR><LF>
CLEAR	<i>DateTime_x</i> <CR>CLEAR([<i>Argument</i>])<CR><LF>
	2001102213352870_1<CR>CLEAR()<CR><LF>
PAUSE	<i>DateTime_x</i> <CR>PAUSE<CR><LF>
	2001102213352870_1<CR>PAUSE<CR><LF>
RESUME	<i>DateTime_x</i> <CR>RESUME<CR><LF>
	2001102213352870_1<CR>RESUME<CR><LF>
LOCK_REQ	<i>DateTime_n</i> <CR>LOCK_REQ(<i>PortID</i> {<CR> <i>PortIndexReference</i> }){<CR>(<i>PortID</i> {<CR> <i>PortIndexReference</i> })}<CR><LF>
	2001102213352870_3<CR>LOCK_REQ(("Rack1"<CR>1<CR>2)<CR>("Rack2"))<CR><LF>
UNLOCK_REQ	<i>DateTime_n</i> <CR>UNLOCK_REQ<CR><LF>
	2001102213352870_3<CR>UNLOCK_REQ<CR><LF>
RUN_OP(<i>command</i>)	<i>DateTime_n</i> <CR>RUN_OP(<i>CommandID</i> [<CR>(<i>Argument</i>)]{<CR> <i>Argument2</i> })<CR>[<i>StartDateTime</i>]<CR><LF>
	2001102213352870_3<CR>RUN_OP("DoSomething"<CR>(1<CR>2)<CR>2001102213371868)<CR><LF>
STATUS_REQ	<i>DateTime_n</i> <CR>STATUS_REQ(<i>INTERACTION</i> / <i>MAINTENANCE</i> / <i>PORT/ALARM</i> / <i>SYSTEM_VARIABLE</i> <i>RESOURCE</i> [(<i>Argument</i>)]<CR><LF>
	2001102213352870_3<CR>STATUS_REQ("INTERACTION"<CR>"2001102213352870_1")<CR><LF> 2001102213352870_3<CR>STATUS_REQ("SYSTEM_VARIABLE"<CR>"iLastSolventTank")<CR><LF> 2001102213352870_3<CR>STATUS_REQ("MAINTENANCE"<CR>"")<CR><LF>
ESTOP	<i>DateTime</i> <CR>ESTOP<CR><LF>
	2001102213352870<CR>ESTOP<CR><LF>
ABORT_REQ	<i>DateTime_n</i> <CR>ABORT_REQ(<i>Argument</i>)<CR><LF>
	2001102213352870_5<CR>ABORT_REQ("2001102213352870_3")<CR><LF>
NEXTEVENT	<i>DateTime_n</i> <CR>NEXTEVENT<CR><LF>
	2001102213352870<CR>NEXTEVENT<CR><LF>

Table 12. Syntax Of TSC Responses To SLM Event

Event Response	Purpose
REMOTE_CTRL_DENIED	<i>DateTime</i> <CR>REMOTE_CTRL_DENIED(<i>ReasonCode</i> [<CR> <i>ReasonText</i>])<CR><LF>
	2001102213352870<CR>REMOTE_CTRL_DENIED(-1022<CR>"OperatorOverwrite")<CR><LF>
REMOTE_CTRL_GRANTED	<i>DateTime</i> <CR>REMOTE_CTRL_GRANTED<CR><LF>

	2001102213352870<CR>REMOTE_CTRL_GRANTED<CR><LF>
LOCAL_CTRL_DENIED	<i>DateTime</i> <CR>LOCAL_CTRL_DENIED(<i>ReasonCode</i> [<CR> <i>ReasonText</i>])<CR><LF>
	2001102213352870<CR>LOCAL_CTRL_DENIED(-1023<CR>"Processing")<CR><LF>
LOCAL_CTRL_GRANTED	<i>DateTime</i> <CR>REMOTE_CTRL_GRANTED<CR><LF>
	2001102213352870<CR>REMOTE_CTRL_GRANTED<CR><LF>

Table 13. Syntax Of Acknowledgement Messages

Message	Purpose
ACK	<i>DateTime_n</i> <CR>ACK<CR><LF>
	2001102213352870<CR>ACK<CR><LF>
NACK	<i>DateTime_n</i> <CR>NACK("ReasonText"[<CR>(Argument1 {<CR>Argument})])<CR><LF>
	2001102213352870_5<CR>NACK("INVALID DATA TYPE"<CR>(4<CR>"double"))<CR><LF>

Table 14. Syntax Of SLM To TSC Events

Events	Purpose
LOCAL_CTRL_REQ	<i>DateTime</i> <CR> <i>EventDateTime</i> <CR>REMOTE_CTRL_REQ<CR><LF>
	2001102213352870<CR>2001102213373431<CR>REMOTE_CTRL_REQ<CR><LF>
REMOTE_CTRL_REQ	<i>DateTime</i> <CR> <i>EventDateTime</i> <CR>LOCAL_CTRL_REQ<CR><LF>
	2001102213352870<CR>2001102213373431<CR>LOCAL_CTRL_REQ<CR><LF>
STATE_CHANGED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>STATE_CHANGED(<i>PreviousState</i> <CR> <i>NewState</i>)<CR><LF>
	2001061309080435_2<CR>2001061309074823<CR>STATE_CHANGED(""<CR>"POWERED UP")<CR><LF>
ALARM_ON	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>ALARM_ON(<i>AlarmID</i> [<CR> <i>AlarmText</i>])<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>ALARM_ON("-00001"<CR>"invalid state")<CR><LF>
ALARM_OFF	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>ALARM_OFF(<i>AlarmID</i>)<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>ALARM_OFF("-00001")<CR><LF>
REMOTE_CTRL_DENIED	<i>DateTime</i> <CR> <i>EventDateTime</i> <CR>REMOTE_CTRL_DENIED(<i>ReasonCode</i> [<CR> <i>ReasonText</i>])<CR><LF>
	2001102215564578<CR>2001102215565603<CR>REMOTE_CTRL_DENIED("-1022"<CR>"OperatorOverwrite")<CR><LF>
REMOTE_CTRL_ACCEPTED	<i>DateTime</i> <CR> <i>EventDateTime</i> <CR>REMOTE_CTRL_ACCEPTED<CR><LF>
	2001102215564578<CR>2001102215565603<CR>REMOTE_CTRL_ACCEPTED<CR><LF>
LOCAL_CTRL_DENIED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>LOCAL_CTRL_DENIED(<i>ReasonCode</i> [<CR> <i>ReasonText</i>])<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>LOCAL_CTRL_DENIED("-1023"<CR>"Processing")<CR><LF>
LOCAL_CTRL_ACCEPTED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>REMOTE_CTRL_ACCEPTED<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>REMOTE_CTRL_ACCEPTED<CR><LF>
OP_DENIED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>OP_DENIED(<i>ReasonCode</i> [<CR> <i>ReasonText</i>])<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>OP_DENIED("-1023"<CR>"Processing")<CR><LF>
OP_STARTED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>OP_STARTED<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>OP_STARTED<CR><LF>
OP_RESULT	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>OP_RESULT(<i>Response</i>){<CR> <i>Response</i> }<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>OP_RESULT

	ULT(1.234<CR>0.1)<CR><LF>
OP_COMPLETED	<i>DateTime_n<CR>EventDateTime<CR>OP_COMPLETED<CR><LF></i>
	2001102215564578_3<CR>2001102215565603<CR>OP_COMPLETED<CR><LF>
LOCK_DENIED	<i>DateTime_n<CR>EventDateTime<CR>LOCK_DENIED(ReasonCode[<CR>ReasonText])<CR><LF></i>
	2001102215564578_3<CR>2001102215565603<CR>LOCK_DENIED("1024"<CR>"Port in use")<CR><LF>
LOCK_ACCEPTED	<i>DateTime_n<CR>EventDateTime<CR>LOCK_ACCEPTED<CR><LF></i>
	2001102215564578_3<CR>2001102215565603<CR>LOCK_ACCEPTED<CR><LF>
LOCKED	<i>DateTime_n<CR>EventDateTime<CR>LOCKED<CR><LF></i>
	2001102215564578_3<CR>2001102215565603<CR>LOCKED<CR><LF>
UNLOCKED	<i>DateTime_n<CR>EventDateTime<CR>UNLOCKED<CR><LF></i>
	2001102215564578_3<CR>2001102215565603<CR>UNLOCKED<CR><LF>
STATUS	<p><u>Interaction:</u> <i>DateTime_n<CR>EventDateTime<CR>STATUS((InteractionID<CR>InteractionType<CR>InteractionState){<CR>(InteractionID<CR>InteractionType<CR>InteractionState))}<CR><LF></i></p> <p><u>Port:</u> <i>DateTime_n<CR>EventDateTime<CR>STATUS((PortID[<CR>PortIndexID]<CR>PortLockState[<CR>PortCondition][<CR>ResourceID<CR>ResourceQuantity]){<CR>(PortID[<CR>PortIndexID]<CR>PortLockState[<CR>PortCondition][<CR>ResourceID<CR>ResourceQuantityValue[<CR>Unit])}<CR><LF></i></p> <p><u>system variable:</u> <i>DateTime_n<CR>EventDateTime<CR>STATUS(("system_variable"<CR>SystemVariableID<CR>CurrentValue){<CR>("system_variable"<CR>SystemVariableID<CR>CurrentValue)})<CR><LF></i></p> <p><u>maintenance:</u> <i>DateTime_n<CR>EventDateTime<CR>STATUS(("maintenance"<CR>MaintenanceID<CR>NextDueDateTime){<CR>("maintenance"<CR>MaintenanceID<CR>NextDueDateTime)})<CR><LF></i></p> <p><u>resource:</u> <i>DateTime_n<CR>EventDateTime<CR>STATUS(("resource"<CR>ResourceID<CR>CurrentValue[<CR>Unit]){<CR>("resource"<CR>ResourceID<CR>CurrentValue[<CR>Unit])}<CR><LF></i></p> <p><u>Alarm:</u> <i>DateTime_n<CR>EventDateTime<CR>STATUS(AlarmID{<CR>AlarmID}<CR><LF></i></p>
	<p>2001102215564578_3<CR>2001102215565603<CR>STATUS(2001102215564578_1<CR>"Control Flow"<CR>"Normal Operation")<CR><LF></p> <p>2001102215564578_3<CR>2001102215565603<CR>STATUS(("maintenance"<CR>"BalanceCalibration"<CR>20011025165735</p>

	88))<CR><LF> 2001102215564578_3<CR>2001102215565603<CR>STATUS(-10024<CR>-10025)<CR><LF>
NO_STATUS	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>NO_STATUS<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>NO_STATUS<CR><LF>
ABORT_ACCEPTED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>ABORT_ACCEPTED<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>ABORT_ACCEPTED<CR><LF>
ABORT_DENIED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>ABORT_DENIED(<i>ReasonCode</i> [<CR> <i>ReasonText</i>])<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>ABORT_DENIED("1025"<CR>"Processing started")<CR><LF>
ABORT_COMPLETED	<i>DateTime_n</i> <CR> <i>EventDateTime</i> <CR>ABORT_COMPLETED<CR><LF>
	2001102215564578_3<CR>2001102215565603<CR>ABORT_COMPLETED<CR><LF>

An example TCP message sequence is illustrated in Table 15.

Table 15. Example TCP Packets

	Message Initiator	Message
1.	TSC	2000030117413312<CR>NEXTEVENT<CR><LF>
2.	SLM	2000030117413312<CR>ACK<CR><LF>
3.	SLM	2000030117413312_2<CR>200003011742261288<CR>STATE_CHANGED(""<CR>"POWERED UP")<CR><LF>
4.	TSC	2000030117413312_2<CR>ACK<CR><LF>
5.	TSC	2000030117413312<CR>NEXTEVENT<CR><LF>
6.	SLM	2000030117413312<CR>ACK<CR><LF>
7.	TSC	2000030117413312_1<CR>REMOTE_CTRL_REQ<CR><LF>
8.	SLM	2000030117413312_1<CR>ACK<CR><LF>
9.	SLM	2000030117413312_1<CR>200003011743052848<CR>REMOTE_CTRL_ACCEPTED<CR><LF>
10.	TSC	2000030117413312_1<CR>ACK<CR><LF>
11.	TSC	2000030117413312<CR>NEXTEVENT<CR><LF>
12.	SLM	2000030117413312<CR>ACK<CR><LF>
13.	TSC	2000030117413312_1<CR>INIT<CR><LF>
14.	SLM	2000030117413312_1<CR>ACK<CR><LF>
15.	SLM	2000030117413312_1<CR>200003011743564107<CR>STATE_CHANGED("INITING"<CR>"IDLE")<CR><LF>
16.	TSC	2000030117413312_1<CR>ACK<CR><LF>
17.	TSC	2000030117413312<CR>NEXTEVENT<CR><LF>
18.	SLM	2000030117413312<CR>ACK<CR><LF>

2.3.3.2 Null Values

Null values for text arguments are indicated by empty quotes, other values are omitted. The example in Figure 5 is shown again in Figure 10 with a null value for the first text argument and the *StartDateTime* and *ItemList* arguments omitted.

Figure 10. Example TSC TCP Packet With Optional Characters

DateTime_n<CR>RUN_OP("DoSomething"<CR>(""*NumericArgument2*)<CR><CR>)<CR><LF>

2.3.4 Session And Interaction Identifiers

When an SLM and TSC establish communications the TSC creates a unique session identifier for that link. The session identifier takes the format of date and time string that uniquely identifies the connection between the SLM and the TSC.

To uniquely identify interactions within a session an incrementing sequence number is appended to the date and time string separated by an underscore. The interaction number is used to identify command or event and response groups so that acknowledgements (ACKs) and command or event responses can be paired off. The TSC will generate odd sequence numbers and the SLM will generate even sequence numbers for unsolicited events.

Any message exchange that pertains to the primary interactions, such as the Local/Remote Control interaction, does not have a sequential interaction identifier appended to it. This can be considered as the parent Interaction as all others logically exist within it.

The Next Event interaction does not require a sequence number appended as it is used for flow control as part of the session.

Primary LECIS commands (INIT, SETUP, CLEAR, PAUSE, RESUME, ESTOP) exist within the same interaction (Control Flow) and use the same interaction identifier.

All SLM generated interaction identifiers for SLM initiated state changed (except STATE_CHANGED (ESTOPPED)), asynchronous alarm and event messages will have their own unique sequential identifier for each interaction. Alarms that are defined as synchronous in the DCD, are labelled with the same interaction identifier as the command during the execution of which the error was detected.

An example date and time string is shown in Figure 11 and a message sequence illustrating the incremental values is shown in Figure 3.

Figure 11. Example Date And Time Stamp With Sequence Number

DateTime_n = *yyyymmddhhmmssss_n*

Table

16,

Table 17 and Table 18 group the TSC to SLM commands that exist within the parent, primary and secondary session interactions.

Table 16. Parent Session Identifiers

From	Message	Session/Interaction Identifier Format
TSC or SLM	REMOTE_CTRL_REQ	<i>DateTime</i>
TSC or SLM	LOCAL_CTRL_REQ	<i>DateTime</i>
TSC	ESTOP	<i>DateTime</i>
TSC	NEXTEVENT*	<i>DateTime</i>
SLM	STATE_CHANGED (previous state, ESTOPPED)	<i>DateTime</i>

* *optional, if solution suggested by Table 18 is not adopted*

Table 17. Primary Session Interaction Identifiers

From	Message	Session/Interaction Identifier Format
SLM	STATE_CHANGED (, POWERED_UP)	<i>DateTime_X</i>
TSC	INIT	<i>DateTime_X</i>
TSC	SETUP	<i>DateTime_X</i>
TSC	PAUSE	<i>DateTime_X</i>
TSC	RESUME	<i>DateTime_X</i>
TSC	CLEAR	<i>DateTime_X</i>

For this table, x = 2 if SLM initiated the Control Flow interaction with the STATE_CHANGED(POWERED UP) report, otherwise x = 1.

Table 18. TSC Secondary Session Interaction Identifiers

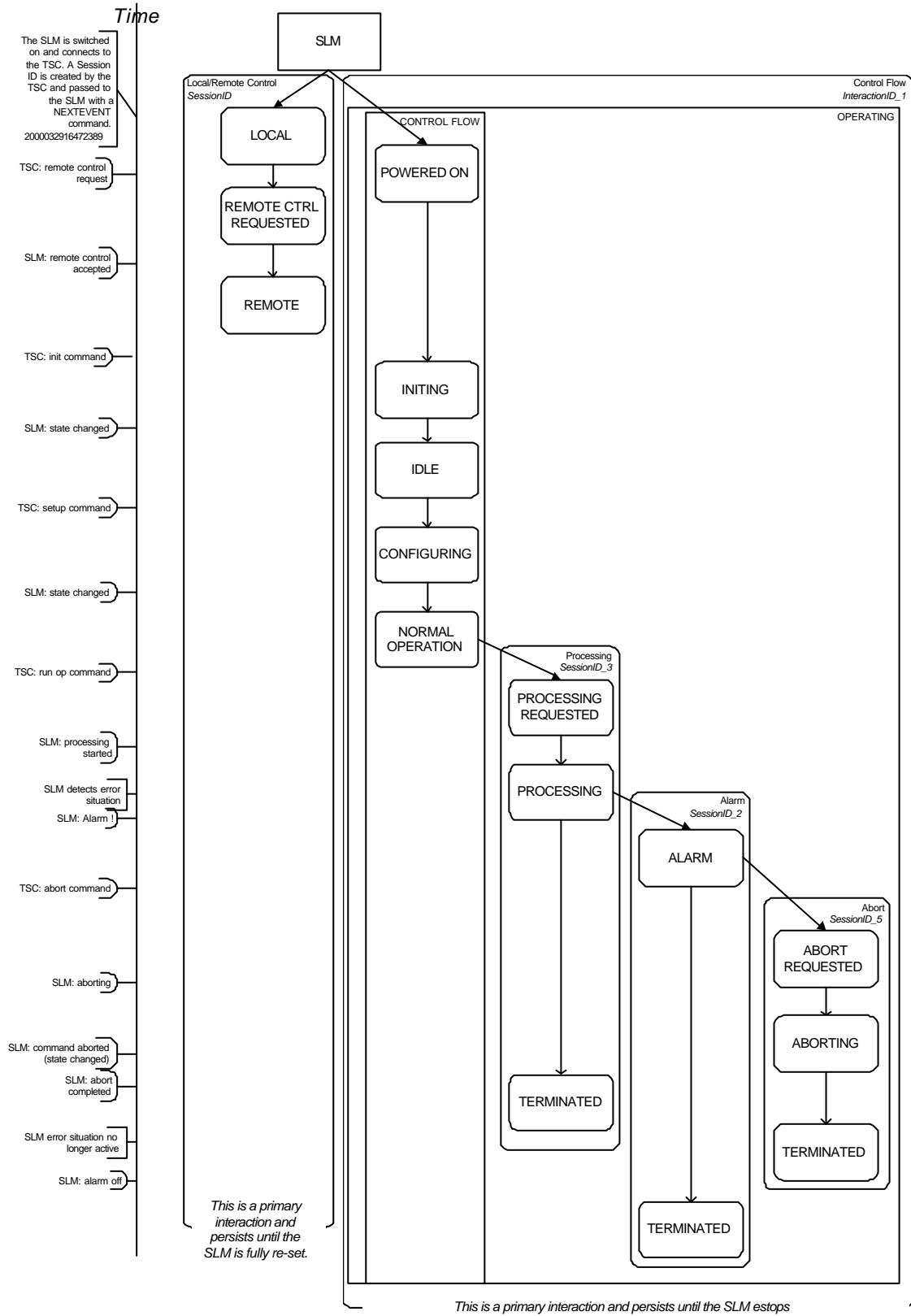
Command	Session/Interaction Identifier Format
LOCK_REQ	<i>DateTime_a</i>
UNLOCK_REQ	<i>DateTime_a</i>
RUN_OP(<i>command</i>)	<i>DateTime_b</i>
STATUS_REQ	<i>DateTime_c</i>
ABORT_REQ	<i>DateTime_d</i>
NEXTEVENT*	<i>DateTime_e</i>

* *optional, if solution suggested by Table 16 is not adopted.*

a to e are odd numbers > 1.

Figure 12 shows an example of a pseudo interaction diagram with the Session/Interaction Identifier creation points marked. Although the diagram implies a hierarchical relationship between the interactions at various points the interactions exist in parallel. The diagram has been laid out this way to illustrate the sequential nature of the message exchange.

Figure 12. State Diagram With Session/Interaction Identifiers



2.4 Initialisation

This section outlines the steps required to set-up the SLM so that it is in a useable state.

Table 19 starts with both the TSC and the SLM being powered on and the TSC starting with a ready state and the SLM starting with a powered up state. This implies that they have both successfully completed any self-diagnostic checks that they may perform. Once an SLM has powered up and completed POST (Power-On Self Test) checks it must first initiate communications with the TSC. This will be done by the SLM performing a TCP connect to a pre-defined port on the TSC. The TSC will then resolve the IP address of the SLM that has just made the connection and check it against a list of available DCDs. Contained within the DCD for each SLM is the physical address for its communications port or channel. SLM initialisation and set-up takes place after it transfers to *remote control* mode and is then controlled by the TSC. If the SLM is to be used in *local control* mode it must be initialised and set-up manually. In this case it then has to be reset before it is made available to the TSC again to ensure that its internal state corresponds to POWERED UP. The first command that the SLM will receive is a NEXTEVENT, this clears it to communicate the TSC at a 'LECIS' level.

Table 19. Initialisation Message Exchange

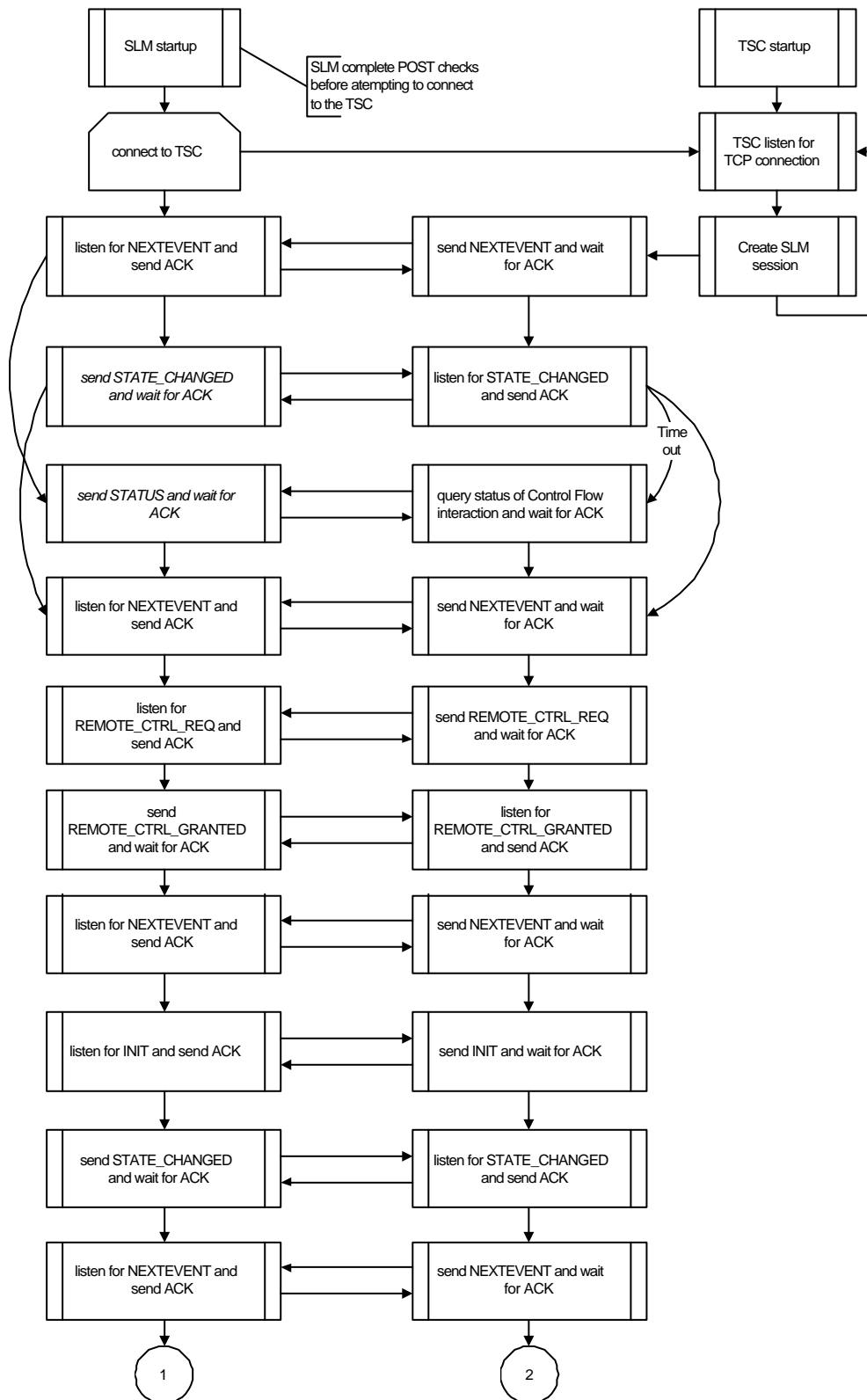
	Message Initiator	SLM Event	TSC Command	Response	Primary SLM State
1.	TSC		NEXTEVENT		LOCAL (POWERED UP)
2.	SLM			ACK	
3.	SLM	STATE_CHANGED (,POWERED_UP) ^{a8}			
4.	TSC			ACK	
5.	TSC		NEXTEVENT		
6.	SLM			ACK	
7.	TSC		REMOTE_CTRL_REQ		
8.	SLM			ACK	REMOTE CONTROL REQUESTED
9.	SLM	REMOTE_CTRL_ACCEPTED			
10.	TSC			ACK	REMOTE
11.	TSC		NEXTEVENT		
12.	SLM			ACK	
13.	TSC		INIT		
14.	SLM			ACK	INITING (REMOTE)
15.	SLM	STATE_CHANGED			
16.	TSC			ACK	IDLE (REMOTE)
17.	TSC		NEXTEVENT		
18.	SLM			ACK	
19.	TSC		SETUP		
20.	SLM			ACK	CONFIGURING (REMOTE)
21.	SLM	STATE_CHANGED			
22.	TSC			ACK	NORMAL OPERATION (REMOTE)

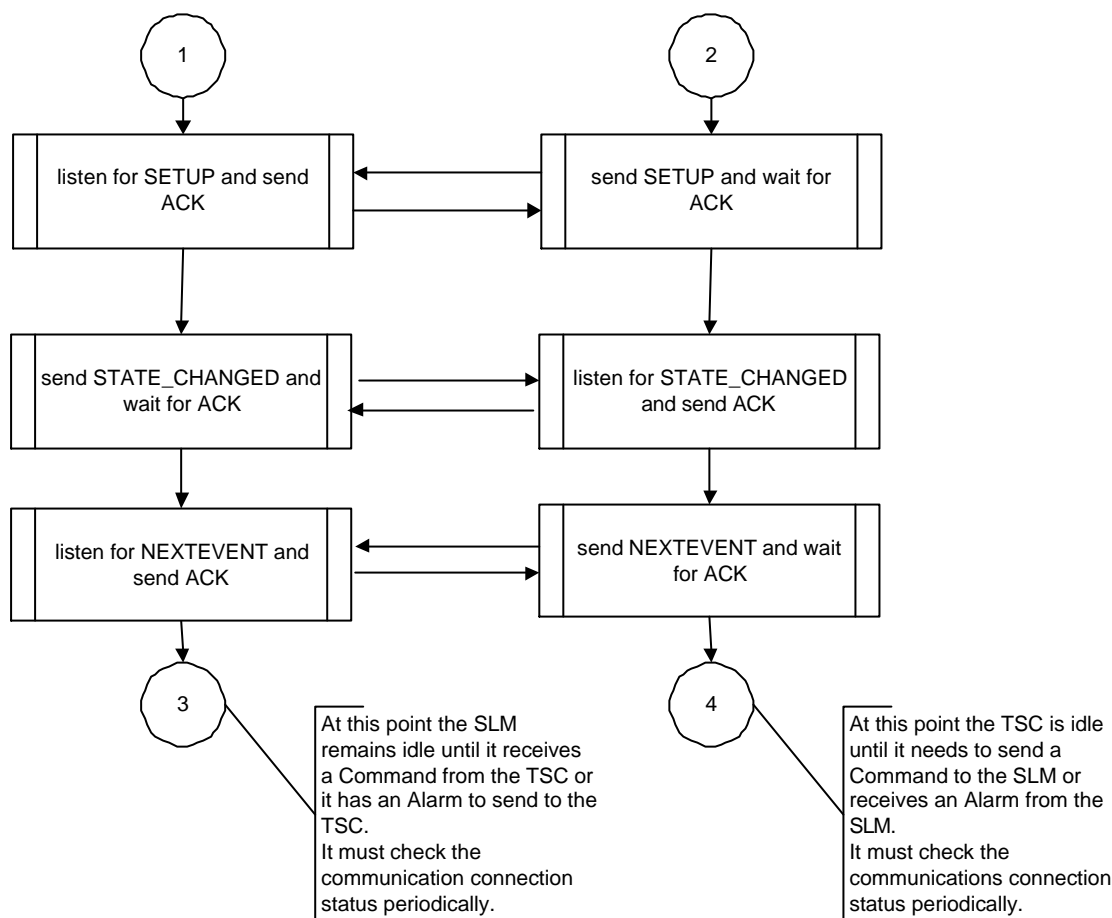
23	TSC		NEXTEVENT		
24	SLM			ACK	

**optional*

The initialisation message sequence in Table 19 is represented in Figure 13.

Figure 13. Initialisation Process Flow





At the end of Figure 13 the SLM is now ready to receive commands from the TSC and process them.

2.5 Communications Recovery

As described previously it is the responsibility of the SLM to reconnect the communications channel. However, during idle time the TSC should check that each SLM is available periodically (e.g. once every five to ten minutes). If an error is detected by the TSC it should raise an appropriate error message to the application program requesting operator intervention, or, for more sophisticated implementations of a TSC, trigger an error handler routine for user defined action sequences.

The following scenarios assume that it takes a negligible amount of time to correct line conditions to allow the SLM to reconnect to the TSC. The scenarios in the following tables illustrate recovery from communications failures: -

- Example 1, Table 20, shows the SLM discovering a line error during a processing operation where loss of communications will not affect the results.
- Example 2, Table 21, shows the SLM discovering a line error during a processing operation where loss of communications will have an adverse effect on the results.

Table 20. SLM Verifies Connection #1

	Message Initiator	Command/Event /Response	Primary SLM State	Secondary SLM State	Comments
1			NORMAL OPERATION	PROCESSING	SLM does line check or wants to send a message to the TSC .
2					SLM detects line error.
3					SLM continues processing buffering any message (and data) until command buffer is empty. SLM indicates line error status on local console.
4					User fixes communications line.
5					SLM connects to TSC.
6	TSC	NEXTEVENT			TSC detects connection from an SLM that already has an active session.
7	SLM	ACK			
8	SLM	event report			The SLM sends all buffered messages in the order of occurrence, starting – if applicable- with the first message that was not acknowledged.
11	TSC	ACK			
12	TSC	NEXTEVENT			
13	SLM	ACK			

The process is the same whether the SLM establishes communication after a reset or recovers from a break during a run:

1. SLM connects to TSC.
2. SLM waits for NEXTEVENT.
3. SLM sends buffered message (which could be STATE_CHANGED (POWERED UP) or any other event report depending on situation).

Steps 2/3 are repeated until all buffered messages have been sent.

The buffered event reports reference with their interaction identifiers the previous session. For the NEXTEVENT messages and any new interactions, however, the new session identifier is used to build the interaction ids.

Table 21. SLM Verifies Connection #2

	Message Initiator	Command/Event /Response	Primary SLM State	Secondary SLM State	Comments
1			NORMAL OPERATION	PROCESSING	SLM does line check or wants to send a message to the TSC.
2					SLM detects line error.
3					If the SLM can not finish processing it should raise an alarm.
4	SLM		(PAUSED)	ALARM	SLM buffers any message (and data). SLM indicates line status on local console.
5					User fixes line.
6					SLM connects to TSC.
7	TSC	NEXTEVENT			TSC detects connection from an SLM

					that already has an active session.
8	SLM	ACK			
11	SLM	event report			The SLM sends all buffered reports in the order of occurrence, including the STATE_CHANGED (NORMAL OPERATION, PAUSING), STATE_CHANGED (PAUSING, PAUSED) and ALARM_ON.
12	TSC	ACK			
13	TSC	NEXTEVENT			
14	SLM	ACK			
20					At this point the TSC can act upon the ALARM message sent by the SLM.

2.5.1 Message Recovery

The following scenarios outline the process stages involved in checking for correct LECIS messages. The diagrams illustrate recovery from communications failures or resynchronising message flows.

2.5.1.1 Communications Check

Any syntactical checks on receipt of a message should be carried out and any message should be sent without undue delay, such that timeout errors indicate a problem.

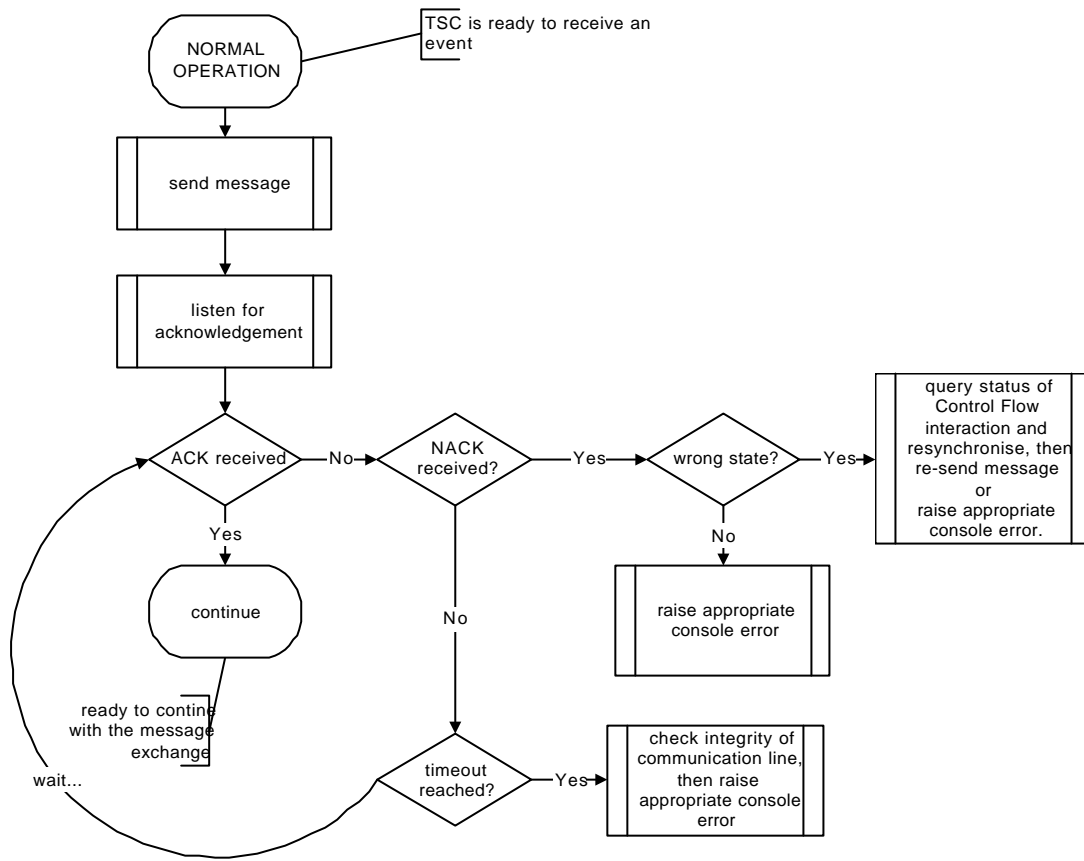
Figure 14, shows the dialogue between the TSC and SLM when the TSC checks for an acknowledgement for a given session and interaction. The TSC also checks for negative acknowledgements, timeouts and line errors.

All negative acknowledgements – whether sent by the TSC or an SLM - are sent in the following format:

<interaction id><,>NACK(><NACK message><)>

An ESTOP command from the TSC or STATE_CHANGED (ESTOPPED) message from an SLM must not be rejected with a NACK message.

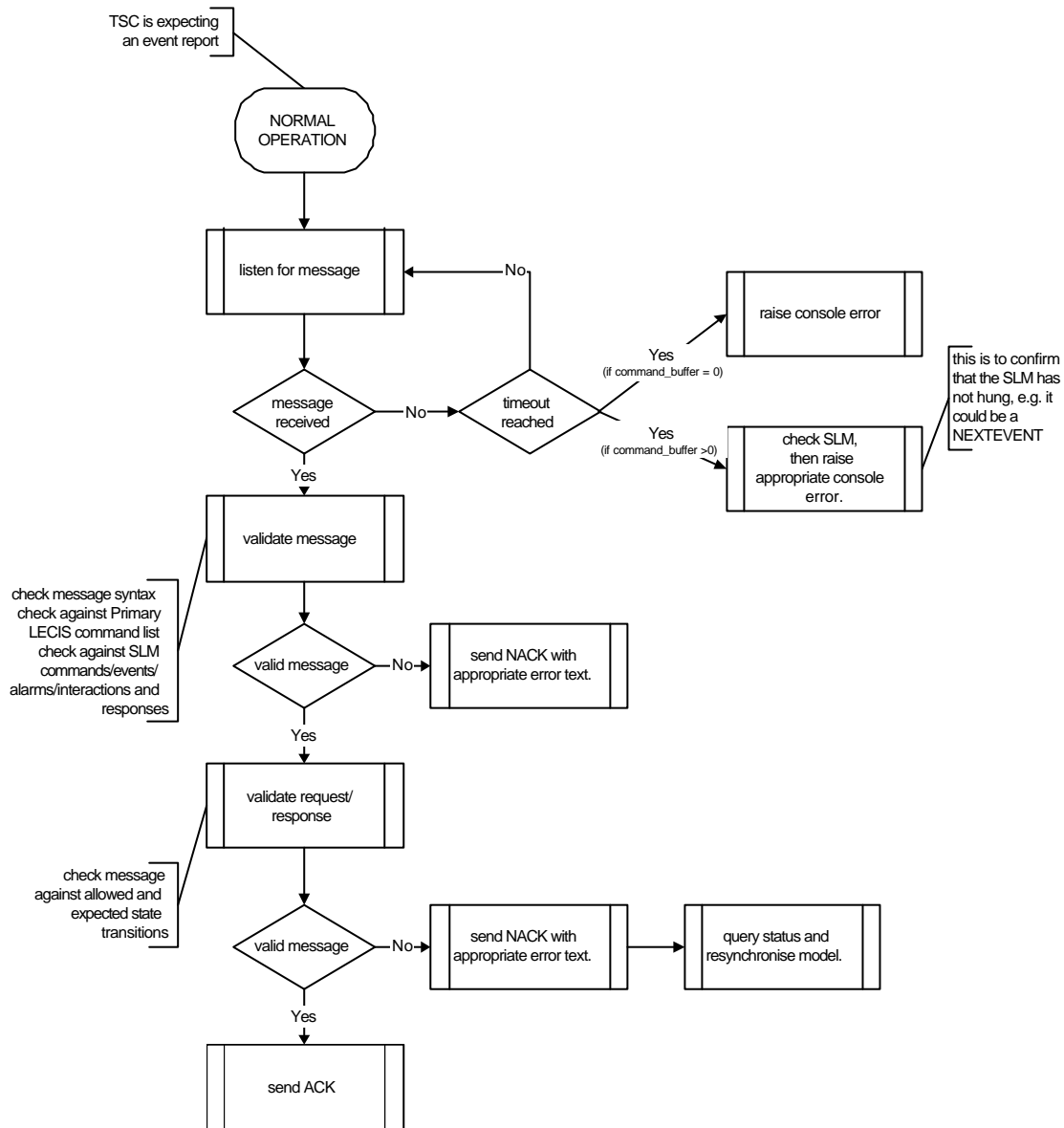
Figure 14 Message Acknowledgement Process Flow



2.5.1.2 Message Communications Check

Example 2, Figure 15, shows a message process flow when receiving messages, i.e. the TSC listening for SLM events, alarms and command responses.

Figure 15. Message Process Flow



2.6 Error Conditions

Error conditions can be divided into either emergency (ESTOP) or non-emergency (ALARM) conditions.

2.6.1 Emergency Stop

An emergency stop (ESTOP) can be initiated by the TSC or the SLM. The SLM will abort all secondary interactions immediately, including Lock/Unlock interactions, and transit to an ESTOPPED state and await manual intervention in local control. If the SLM initiates an emergency stop it sends the TSC an ALARM message indicating the problem as well as a STATE_CHANGED message.

Table 22. TSC Emergency Stop

	Message Initiator	Command/Event/Response	New Primary SLM State	New Secondary SLM State	Comments
1			NORMAL OPERATION	<i>AnyState</i>	
2	TSC	ESTOP			
3	SLM	ACK			SLM aborts all current processing (if any) and stops immediately.
					At this point the SLM will transit to local control /ESTOPPED awaiting manual intervention.
					A state changed message indicating the transition to ESTOPPED must not be NACKed. However, LECIS communication does not end in ESTOPPED, so the acknowledgement should be sent.

The dialogue between the SLM and TSC for an SLM-initiated estop is shown in Table 23.

Table 23. SLM Emergency Stop

	Message Initiator	Command/Event/Response	New Primary SLM State	New Secondary SLM State	Comments
1			NORMAL OPERATION	<i>AnyState</i>	
					The SLM detects an off-normal situation that requires an Estop. SLM aborts all current processing (if any) and stops immediately.
2	SLM	ALARM_ON("Emergency		ALARM	SLM notifies the TSC

		Alarm")			why it had to abort all current processing (if any).
3	TSC	ACK			
4	TSC	NEXTEVENT			
5	SLM	ACK			
6	SLM	STATE_CHANGED ("Anything" "ESTOPPED")	ESTOPPED		
7	TSC	ACK			The SLM will transit to local control awaiting manual intervention.
					A state changed message indicating the transition to ESTOPPED must not be NACKed. However, LECIS communication does not end in ESTOPPED, so the acknowledgement should be sent.

2.6.2 Non-Emergency Conditions

Error conditions are indicated by an Alarm event that is sent from the SLM to the TSC. Each alarm condition that exists for an SLM has a detailed description of why the alarm occurs, how it is cleared and how it affects the TSC and SLM operation. The documentation should include any commands that are affected by the alarm event and recovery commands required from the TSC to clear the condition. The recovery commands for an Alarm are listed in sequence in the Alarm section of the SLM's DCD and are sent with the default values specified for each command in the command section. Once the alarm has been turned off, the commands specified in *configuration_commands* are sent in the same manner. More sophisticated implementations of the TSC may trigger a handler routine that allows the user to specify context-sensitive recovery routines and by-pass the routines specified in the DCD.

In the example below a sensor detects a door being opened. The *Door Open* Alarm requires a PAUSE command to be sent and then manual intervention to close the door. On the door being closed a RESUME command is required to continue. The message exchange between the SLM and TSC is illustrated in Table 24.

Table 24. SLM Non-Emergency Conditions

	Message Initiator	Command/Event/Response	New Primary SLM State	New Secondary SLM State	Comments
1			NORMAL OPERATION	AnyState	
2	SLM	ALARM_ON(-1, "Door Open")			SLM first notifies the TSC what has happened.
3	TSC	ACK		ALARM	
4	TSC	NEXTEVENT			

5	SLM	ACK			
6	TSC	PAUSE			SLM needs this because of the type of alarm. This is specified in the DCD.
7	SLM	ACK	PAUSING		
8	TSC	NEXTEVENT			
9	SLM	ACK			
	SLM	STATE_CHANGED("NORMAL_OPERATION" "PAUSING")			
	TSC	ACK	PAUSING		
	TSC	NEXTEVENT			
	SLM	ACK			
10	SLM	STATE_CHANGED("PAUSING" "PAUSED")			SLM waits for someone to closed the door.
11	TSC	ACK	PAUSED		
12	TSC	NEXTEVENT			
13	SLM	ACK			
14	SLM	ALARM_OFF(-I)			The door is closed and the alarm condition is cleared. SLM then notifies the TSC.
15	TSC	ACK		TERMINATE D	
16	TSC	NEXTEVENT			
17	SLM	ACK			
18	TSC	RESUME			This has been specified as configuration_command
19	TSC	ACK			
20	TSC	NEXTEVENT			
21	SLM	ACK			
22	SLM	STATE_CHANGED("PAUSED" "NORMAL_OPERATION")			
23	TSC	ACK	NORMAL OPERATIO N		
24	TSC	NEXTEVENT			
25	SLM	ACK			

3 Device Capability Dataset

3.1 Core DCD Requirements

The LECIS¹ document refers to the CAALS (Consortium on Automated Analytical Laboratory Systems) Device Capability Dataset² (DCD) document as a means of storing and maintaining information about SLMs. However, the structure of the proposed rule set has been deemed inadequate to express the requirements of the LECIS, and this document contains a revised version of it. The information in the DCD document has been interpreted into an XML (Extensible Markup Language) DTD (Document Type Definition) for a LECIS DCD. The XML DTD is a set of rules against which an XML document can be validated. The DTD rule set is common for all SLMs.

A vendor will provide an XML Device Capability Dataset document with each SLM. This XML document describes the capabilities of their SLM and lists the instrument specific alarm, command and event message set that are available to the TSC. Since datasets are likely to be maintained in a database system by the TSC, an SLM should not rely on having access to its DCD.

For an XML DCD to be valid it must contain an Administrative section as a minimum, but a DCD that contained an Administrative section only would be of little use! All of the sections that make up a DCD are listed in Table 25.

Table 25. DCD Sections

Section	Occurrence	Description
Administrative	Once	Contains information about and uniquely identifies the instrument
Alarm	Zero or many	Lists abnormal conditions that can occur
Command	Zero or many	List of commands and their arguments and returned information that the instrument can process. The commands specified here are sent to the SLM as arguments to RUN_OP.
Message	Zero or many	List of messages that the SLM and/or the TSC can generate. This section has been introduced to allow the specification of LECIS required and optional primary messages, as well as custom NACK messages.
Interaction	Zero or many	This section is available to provide for interactions beyond the scope of LECIS
Maintenance	Zero or many	Maintenance schedules and status
Physical Characteristic	Zero or many	This section has been 'glossed over' in the XML DTD because of its complexity and

		limitations in XML
Port	Zero or many	List of data and/or material access points
Resource	Zero or many	Identifies resource requirements and outputs
State	Zero or many	This section is available to provide for states beyond the scope of LECIS
System Variable	Zero or many	Caters for semi-static storage of SLM variables that the TSC will need access to

3.2 DCD Constraints

3.2.1 Definitions

Commands, messages, states, interactions, ports, etc. can only be specified in the main section of the DCD; any references used in sub-definitions (e.g. <command_reference> in <command_or_message>) must point to a command id that has already been declared in the main section.

3.2.2 Data Type Values

In addition to the CAALS DCD, the values for the *data_type* element *data_type_value* attribute are restricted the IEEE data type representations listed in Table 26.

Table 26. data_type_value Attribute Values In XML DTD

Value	Representation
boolean	8 bit value: where zero = FALSE and non-zero = TRUE
byte integer	8 bit value: whole number, signed or unsigned
word integer	16 bit value: whole number, signed or unsigned
long integer	32 bit value: whole number, signed or unsigned
single	32 bit value: signed floating point number
double	64 bit value: signed floating point number
string	one-dimensional alphanumeric character array

3.2.3 Default Values

The (*argument*) *default* element will be converted to the data type of the (*argument*) *data_type* element *data_type_value* attribute and have the same unit attribute as the (*argument, value_range, range, lower_limit, measure*) *measure_unit* element *measure_prefix_value* and *measure_unit_value* attributes.

3.2.4 lower_limit And upper_limit Values

The following rules are applied to value *X* by the *lower_limit* and *upper_limit* element values: -

1. If *lower_limit* does not equal NULL and *upper_limit* equals NULL then *X* must be greater than or equal to *lower_limit*.

2. If *lower_limit* does not equal NULL and *upper_limit* does not equal NULL and *lower_limit* equals *upper_limit* then *X* must equal *lower_limit*.
3. If *lower_limit* does not equal NULL and *upper_limit* does not equal NULL and *lower_limit* does not equal *upper_limit* then *lower_limit* must be less than *upper_limit* and *X* must be greater than or equal to *lower_limit* and less than or equal to *upper_limit*.

3.2.4.1 *measure_unit* Attribute Values

It is the responsibility of the SLM vendor to ensure that there is a logical relationship between the *measure_unit* attributes when used in conjunction with the *measure_value* element within the context of *lower_limit* and *upper_limit* elements. The TSC application provider must also perform a logical check on the *measure_unit* attributes when used in the same context.

3.2.5 TCP/IP Port Definitions

To enable the TSC to work with any SLM, the SLMs DCD should contain the same representation for the TCP/IP port. This includes two entries, the first is used to identify the address and port of the SLM that the TSC should expect connections from and the second is used to identify the address and port of the TSC that the SLM should connect to. Examples of these are shown in Table 27 and Table 28.

Table 27. SLM TCP/IP Port Values In XML DCD

Name	Value	Description
port_id	tcpip_in	SLM address
category label	communications	
port_type port_type_value	data	
port_access_type port_access_type_value	in	
port_location location	<i>IPAddress:PortNumber</i>	TCP/IP address and port in the format: 127.0.0.1:50

Table 28. TSC TCP/IP port Values in XML DCD

Name	Value	Description
port_id	tcpip_out	TSC address
category label	communications	
port_type port_type_value	data	

port_access_type	out	
port_access_type_value		
port_location location	<i>IPAddress:PortNumber</i>	TCP/IP address and port in the format: 127.0.0.1:50

3.2.6 Recovery Commands

The commands listed in the *alarm/recovery_commands/command_reference* section are executed sequentially with the default values specified for each command argument.

3.2.7 Commands

The commands defined in the *command* section are sent as arguments to RUN_OP to initiate a Processing interaction. LECIS required messages (including NACK), as well as custom defined messages, are defined within the *message* section.

3.2.8 Other Requirements

interaction_states lists all possible states for an interaction; the sequence of occurrence in a particular instance of the interaction depends on the messages sent.

entry and *exit* are lists of possible ways to enter or exit a state; for an instance of a state only one of each is applicable as required in this particular situation.

For all other lists all items must be used in the defined order.

priority for alarms indicates the SLMs behaviour after an abnormal situation has been detected:

1. The SLM initiates an estop.
2. The SLM initiates transitions to pausing, then paused.
3. The SLM suspends activity until the alarm is turned off.
4. The SLM continues processing.

4 Appendix 1: DTD

<!-- start of xml dtd for lecis device capability dataset -->

<!-- ?xml version="1.0" encoding="ISO-8859-1" ? -->

<!-- DCD version 1.1 -->

<!-- slm device capability dataset, main body -->

<!ELEMENT Device_Capability_Dataset (administrative, alarm*, command*, message*, interaction*, maintenance*, physical_characteristics*, port*, resource*, state*, system_variable*)>

<!-- main sections -->

<!ELEMENT administrative (manufacturer, model_number, serial_number, software_version_number?, dcd_version_number, update_address?, support_address?, information_address?, command_buffer_size)>

<!ELEMENT alarm (alarm_id, description, priority, category?, devices?, recovery_commands?, configuration_commands?)>

<!ELEMENT command (command_id, description, category?, formal_arguments?, duration?, required_configurations?, exclusion_list?, response_data?, synchronous_alarms?)>

<!ELEMENT message (message_id, description, category?, originator, message_type?, priority, message_data_syntax?)>

<!ELEMENT interaction (interaction_id, description, interaction_type, interaction_states+)>

<!ELEMENT maintenance (maintenance_id, description, maintenance_type?, performance_date_time, performance_repetition, maintenance_tasks)>

<!ELEMENT physical_characteristics (slm_location?, geometry?, slm_devices?)>

<!ELEMENT port (port_id, description, port_type)>

<!ELEMENT resource (resource_id, description, category)>

<!ELEMENT state (state_id, description, required_parent_state, entry+, exit*)>

What does the required_parent_state mean?<!ELEMENT system_variable (variable_id, description, data_type, variable_access_type, category, device_reference?, initial_value?, value_range?, current_value)>

<!-- main definitions -->

<!-- administrative definitions -->

<!ELEMENT manufacturer (#PCDATA)>

<!ELEMENT model_number (#PCDATA)>

<!ELEMENT serial_number (#PCDATA)>

<!ELEMENT software_version_number (#PCDATA)>

<!ELEMENT dcd_version_number (#PCDATA)>

<!ELEMENT update_address (#PCDATA)>

<!ELEMENT support_address (#PCDATA)>

<!ELEMENT information_address (#PCDATA)>

<!ELEMENT command_buffer_size (#PCDATA)>

<!-- alarm definitions -->

<!ELEMENT alarm_id (identifier)>

<!ELEMENT recovery_commands (command_or_message+)>

```
<!ELEMENT configuration_commands (command_or_message+)>

<!-- command definitions -->
<!ELEMENT command_id (identifier)>
<!ELEMENT formal_arguments (argument+)>
<!ELEMENT duration (time)>
<!ELEMENT required_configurations (command_or_message*, variable_value*,
required_parent_state*)>
<!ELEMENT exclusion_list (devices?, resources?, ports? )>
<!ELEMENT response_data (data+)>
<!ELEMENT synchronous_alarms (alarm_reference+)>
<!ELEMENT devices (device_id+)>
<!ELEMENT resources (resource_id*)>
<!ELEMENT ports (port_id+)>
<!ELEMENT variable_value (variable_id, (measure_value | range))>

<!-- message definitions -->
<!ELEMENT message_id (identifier)>
<!ELEMENT message_data_syntax (argument | complex_argument)+>
<!ELEMENT message_type EMPTY>
<!ATTLIST message_type message_type_value (required | custom | acknowledgement) #REQUIRED >

<!-- interaction definitions -->
<!ELEMENT interaction_id (identifier)>
<!ELEMENT interaction_type EMPTY>
<!ATTLIST interaction_type interaction_type_value (required_primary |
required_secondary |
optional_primary |
optional_secondary ) #REQUIRED >
<!ELEMENT interaction_states (state_reference+ )>

<!-- maintenance definitions -->
<!ELEMENT maintenance_id (identifier)>
<!ELEMENT maintenance_type EMPTY>
<!ATTLIST maintenance_type maintenance_type_value (cleaning | calibration | verification) #IMPLIED
>
<!ELEMENT performance_date_time (date_and_time)>
<!ELEMENT performance_repetition (regular | custom)>
<!ELEMENT regular EMPTY>
<!ATTLIST regular regular_value (once | hourly | daily | weekly | monthly | annual) #IMPLIED>
<!ELEMENT custom (year?, month?, week?, day?, hour?, minute?)>
<!ELEMENT maintenance_tasks (command_reference+)>

<!-- physical_characteristics definitions -->
<!ELEMENT slm_location (location)>
<!ELEMENT slm_devices (device+)>
```

```

<!-- port definitions -->
<!ELEMENT port_id (identifier )>
<!ELEMENT port_type (data_port | material_port)>
<!ELEMENT port_access EMPTY>
<!ATTLIST port_access port_access_type (read | write | readwrite) #REQUIRED >
<!ELEMENT data_port (port_access, port_location)>
<!ELEMENT material_port (port_category, port_location?, capacity, current_contents)>
<!ATTLIST material_port availability (available | unavailable) #REQUIRED>
<!ELEMENT port_category EMPTY>
<!ATTLIST port_category port_category_value (fixed_port | portable_port) #REQUIRED>
<!ELEMENT material_category EMPTY>
<!ATTLIST material_category material_category_value (port | matter) #IMPLIED>
<!ELEMENT capacity (material_category, measure)>
<!ELEMENT current_contents (content, measure?)>
<!ELEMENT content (port_reference | resources)>
<!ELEMENT port_location (location)>

<!-- resource definitions -->
<!ELEMENT resource_id (identifier )>

<!-- state definitions -->
<!ELEMENT state_id (identifier )>
<!ELEMENT required_parent_state (state_reference?)>
<!ELEMENT entry (entry_event?, previous_state?)>
<!ELEMENT exit (exit_event?, next_state?)>
<!ELEMENT entry_event (message_reference)>
<!ELEMENT exit_event (message_reference)>
<!ELEMENT previous_state (state_reference)>
<!ELEMENT next_state (state_reference)>
<!ELEMENT message_reference (message_id)>

<!-- system_variable definitions -->
<!ELEMENT variable_id (identifier )>
<!ELEMENT data_type EMPTY>
<!ATTLIST data_type data_type_value (boolean |
    byte_integer |
    word_integer |
    long_integer |
    single |
    double |
    string |
    complex ) #IMPLIED>
<!ELEMENT variable_access_type EMPTY>
<!ATTLIST variable_access_type variable_access_type_value (read_only |
    write_only |

```

```

    read_write ) #REQUIRED >
<!ELEMENT device_id (identifier)>
<!ELEMENT device_reference (device_id)>
<!ELEMENT initial_value (#PCDATA )>
<!ELEMENT value_range (range+)>
<!ELEMENT current_value (#PCDATA )>

<!-- common definitions -->
<!ELEMENT alarm_reference (alarm_id)>
<!ELEMENT argument ( argument_id, data_type?, default?, value_range?)>
<!ATTLIST argument repetition_type (one |
    zero_or_one |
    one_or_more |
    zero_or_more ) #IMPLIED >
<!ELEMENT argument_id (identifier)>
<!ELEMENT buffer_size (#PCDATA )>
<!ELEMENT category (#PCDATA )>
<!ELEMENT command_or_message (command_reference | message_reference )>
<!ELEMENT command_reference (command_id )>
<!ELEMENT complex_argument ( argument_id, complex )>
<!ELEMENT complex EMPTY>
<!ATTLIST complex complex_type (command | message | state | alarm | port | maintenance | interaction |
resource | system_variable) #REQUIRED
    complex_level (full | reference) #REQUIRED
    required_category CDATA #IMPLIED>
<!ELEMENT configuration_reference (command_reference )>
<!ELEMENT data (data_id, description?, data_type, data_unit? )>
<!ELEMENT data_id (identifier)>
<!ELEMENT data_unit (#PCDATA )>
<!ELEMENT date (day, month, year )>
<!ELEMENT date_and_time (date, time )>
<!ELEMENT day (#PCDATA )>
<!ELEMENT default (#PCDATA )>
<!ELEMENT description (#PCDATA )>
<!ELEMENT device (device_id, description )>
<!ELEMENT event_reference (event_id )>
<!ELEMENT floor_coordinate (x, y)>
<!ELEMENT footprint (floor_coordinate+ )>
<!ELEMENT geometry (footprint, height)>
<!ELEMENT height (#PCDATA )>
<!ELEMENT hour (#PCDATA )>
<!ELEMENT identifier (#PCDATA )>
<!ELEMENT location (#PCDATA )>
<!ELEMENT lower_limit (measure )>
<!ELEMENT material (description, category, quantity? )>
<!ELEMENT measure (measure_value, measure_unit? )>

```

```
<!ELEMENT measure_value (#PCDATA)>
<!ELEMENT measure_unit EMPTY>
<!ATTLIST measure_unit measure_prefix_value      (exa |
    peta |
    tera |
    giga |
    mega |
    kilo |
    hecto |
    deca |
    deci |
    centi |
    milli |
    micro |
    nano |
    pico |
    femto |
    atto) #IMPLIED
    measure_unit_value      (meter |
    litre |
    bit |
    gram |
    second |
    ampere |
    kelvin |
    mole |
    candela |
    radian |
    steradian |
    hertz |
    newton |
    pascal |
    joule |
    watt |
    coulomb |
    volt |
    farad |
    ohm |
    siemens |
    weber |
    tesla |
    henry |
    degree_Celsius |
    lumen |
    lux |
    becquerel |
```

```
gray |
sievert ) #REQUIRED >
<!ELEMENT minute (#PCDATA )>
<!ELEMENT millisecond (#PCDATA )>
<!ELEMENT month (#PCDATA )>
<!ELEMENT originator EMPTY>
<!ATTLIST originator originator_value (tsc | slm | tsc_or_slm) #REQUIRED >
<!ELEMENT port_reference (port_id )>
<!ELEMENT priority (#PCDATA )>
<!ELEMENT quantity (measure )>
<!ELEMENT range (upper_limit?, lower_limit? )>
<!ELEMENT second (#PCDATA )>
<!ELEMENT shedule (start_date, duration)>
<!ELEMENT start_date (date_and_time)>
<!ELEMENT state_reference (state_id )>
<!ELEMENT time (hour?, minute?, second?, millisecond? )>
<!ELEMENT upper_limit (measure )>
<!ELEMENT week (#PCDATA )>
<!ELEMENT x (#PCDATA )>
<!ELEMENT y (#PCDATA )>
<!ELEMENT year (#PCDATA )>
```

5 Appendix 2: Example DCD-XML file

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Device_Capability_Dataset SYSTEM "DCD1_1.DTD">

<Device_Capability_Dataset>
  <administrative>
    <manufacturer>SomeCompany Ltd</manufacturer>
    <model_number>GoodRobot1</model_number>
    <serial_number>SN12345-1</serial_number>
    <software_version_number>1</software_version_number>
    <dcd_version_number>1</dcd_version_number>
    <update_address>www.somecompany.com/updates</update_address>
    <support_address>www.somecompany.com/suport</support_address>
    <information_address>www.somecompany.com/contact</information_address>
    <command_buffer_size>0</command_buffer_size>
  </administrative>
  <alarm>
    <alarm_id>
      <identifier>5000</identifier>
    </alarm_id>
    <description>This robot doesn't work</description>
    <priority>3</priority>
    <category>fatal failure</category>
    <devices>
      <device_id>
        <identifier>Arm</identifier>
      </device_id>
    </devices>
    <recovery_commands>
      <command_or_message>
        <message_reference>
          <message_id>
            <identifier>CLEAR</identifier>
          </message_id>
        </message_reference>
      </command_or_message>
    </recovery_commands>
    <configuration_commands>
      <command_or_message>
        <message_reference>
          <message_id>
            <identifier>SETUP</identifier>
          </message_id>
        </message_reference>
      </command_or_message>
    </configuration_commands>
  </alarm>
</alarm>
```

```
<alarm_id>
  <identifier>5001</identifier>
</alarm_id>
<description>arm axis position unknown</description>
<priority>3</priority>
<category/>
<devices>
  <device_id>
    <identifier>Arm</identifier>
  </device_id>
</devices>
<recovery_commands>
  <command_or_message>
    <command_reference>
      <command_id>
        <identifier>HomeArm</identifier>
      </command_id>
    </command_reference>
  </command_or_message>
</recovery_commands>
<!--no configuration_commands-->
</alarm>
<command>
  <command_id>
    <identifier>HomeArm</identifier>
  </command_id>
  <description>moves arm to home position</description>
  <category>arm</category>
  <formal_arguments>
    <argument>
      <argument_id>
        <identifier>iAxis</identifier>
      </argument_id>
      <data_type data_type_value="long_integer"/>
      <default>1</default>
      <value_range>
        <range>
          <upper_limit>
            <measure>
              <measure_value>1</measure_value>
              <!--no measure unit-->
            </measure>
          </upper_limit>
          <lower_limit>
            <measure>
              <measure_value>3</measure_value>
              <!--no measure unit-->
            </measure>
          </lower_limit>
        </range>
      </value_range>
```

```
</argument>
</formal_arguments>
<duration>
  <time>
    <hour>0</hour>
    <minute>1</minute>
    <second>3</second>
    <millisecond>0</millisecond>
  </time>
</duration>
<required_configurations>
  <variable_value>
    <variable_id>
      <identifier>ArmInitialised</identifier>
    </variable_id>
    <measure_value>true</measure_value>
  </variable_value>
</required_configurations>
<exclusion_list>
  <devices>
    <device_id>
      <identifier>Arm</identifier>
    </device_id>
  </devices>
</exclusion_list>
<response_data>
  <data>
    <data_id>
      <identifier>iActualPosition</identifier>
    </data_id>
    <description>position where the arm has stopped</description>
    <data_type data_type_value="long_integer"><!--no data_unit--></data_type>
    <data_unit/>
  </data>
</response_data>
<synchronous_alarms>
  <alarm_reference>
    <alarm_id>
      <identifier>5001</identifier>
    </alarm_id>
  </alarm_reference>
</synchronous_alarms>
</command>
<command>
  <command_id>
    <identifier>MoveArm</identifier>
  </command_id>
  <description>moves arm to specified position</description>
  <category>arm</category>
</formal_arguments>
<argument>
```

```
<argument_id>
  <identifier>iAxis</identifier>
</argument_id>
<data_type data_type_value="long_integer"/>
<default>1</default>
<value_range>
  <range>
    <upper_limit>
      <measure>
        <measure_value>1</measure_value>
        <!--no measure unit-->
      </measure>
    </upper_limit>
    <lower_limit>
      <measure>
        <measure_value>3</measure_value>
        <!--no measure unit-->
      </measure>
    </lower_limit>
  </range>
</value_range>
</argument>
<argument>
  <argument_id>
    <identifier>iPosition</identifier>
  </argument_id>
  <data_type data_type_value="long_integer"/>
  <default>0</default>
  <value_range>
    <range>
      <upper_limit>
        <measure>
          <measure_value>0</measure_value>
          <!--no measure unit-->
        </measure>
      </upper_limit>
      <lower_limit>
        <measure>
          <measure_value>3000</measure_value>
          <!--no measure unit-->
        </measure>
      </lower_limit>
    </range>
  </value_range>
</argument>
</formal_arguments>
<duration>
  <time>
    <hour>0</hour>
    <minute>1</minute>
    <second>3</second>
```

```
<millisecond>0</millisecond>
</time>
</duration>
<required_configurations>
  <variable_value>
    <variable_id>
      <identifier>ArmlInitialised</identifier>
    </variable_id>
    <measure_value>1</measure_value>
  </variable_value>
</required_configurations>
<exclusion_list>
  <devices>
    <device_id>
      <identifier>Arm</identifier>
    </device_id>
  </devices>
</exclusion_list>
<response_data>
  <data>
    <data_id>
      <identifier>iActualPosition</identifier>
    </data_id>
    <description>position where the arm has stopped</description>
    <data_type data_type_value="long_integer"><!--no data_unit--></data_type>
    <data_unit/>
  </data>
</response_data>
<synchronous_alarms>
  <alarm_reference>
    <alarm_id>
      <identifier>5001</identifier>
    </alarm_id>
  </alarm_reference>
</synchronous_alarms>
</command>

<message>
  <message_id>
    <identifier>MISSING_ARG</identifier>
  </message_id>
  <description>argument missing from message</description>
  <category>error</category>
  <originator originator_value="tsc_or_slm"/>
  <message_type message_type_value="acknowledgement"/>
  <priority>2</priority>
  <message_data_syntax>
    <argument repetition_type="zero_or_more">
      <argument_id>
        <identifier>EXPECTED_ARG_NUMBER</identifier>
      </argument_id>
    </argument_id>
  </argument_id>
</message_data_syntax>
</message_data_syntax>
```

```
<data_type data_type_value="long_integer"/>
  <default/>
</argument>
</message_data_syntax>
</message>

<message>
  <message_id>
    <identifier>INIT</identifier>
  </message_id>
  <description>instructs SLM to initialise</description>
  <category>Control Flow</category>
  <originator originator_value="tsc"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <!--no message_data_syntax-->
</message>
<message>
  <message_id>
    <identifier>SETUP</identifier>
  </message_id>
  <description>instructs the SLM to configure</description>
  <category>Control Flow</category>
  <originator originator_value="tsc"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <argument repetition_type="zero_or_one">
      <argument_id>
        <identifier>sConfigFile</identifier>
      </argument_id>
      <data_type data_type_value="string"/>
      <!--no default or value range-->
    </argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>CLEAR</identifier>
  </message_id>
  <description>instructs the SLM to clear and return to IDLE</description>
  <category>Control Flow</category>
  <originator originator_value="tsc"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <argument repetition_type="zero_or_one">
      <argument_id>
        <identifier>sClearType</identifier>
      </argument_id>
```

```
<data_type data_type_value="string"/>
  <!--no default or value range-->
</argument>
</message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>RUN_OP</identifier>
  </message_id>
  <description>instructs the SLM to perform an SLM specific processing request</description>
  <category>Control Flow 2</category>
  <originator originator_value="tsc"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
        <identifier>ProcessingRequest</identifier>
      </argument_id>
      <complex complex_type="command" complex_level="full"/>
    </complex_argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>NACK</identifier>
  </message_id>
  <description>negative acknowledgement of a message</description>
  <category>Flow Control</category>
  <originator originator_value="tsc_or_slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
        <identifier>NACKMessage</identifier>
      </argument_id>
      <complex complex_type="message" complex_level="full" required_category="error message"/>
    </complex_argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>ABORT_REQ</identifier>
  </message_id>
  <description>instructs the SLM to abort a secondary interaction</description>
  <category>Control Flow 2</category>
  <originator originator_value="tsc"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
```

```
<argument repetition_type="one">
  <argument_id>
    <identifier>slInteractionID</identifier>
  </argument_id>
  <data_type data_type_value="string"/>
  <!--no default or value range-->
</argument>
</message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>REMOTE_CTRL_REQ</identifier>
  </message_id>
  <description>requestst remote control from the SLM</description>
  <category>Local/Remote Control</category>
  <originator originator_value="tsc"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <!--no message_data_syntax-->
</message>
<message>
  <message_id>
    <identifier>STATE_CHANGED</identifier>
  </message_id>
  <description>SLM notifies TSC about a state change that does not come in response to a
command.</description>
  <category>Contrl Flow</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority/>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
        <identifier>PreviousState</identifier>
      </argument_id>
      <complex complex_type="state" complex_level="reference" />
    </complex_argument>
    <complex_argument>
      <argument_id>
        <identifier>NewState</identifier>
      </argument_id>
      <complex complex_type="state" complex_level="reference"/>
    </complex_argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>LOCAL_CTRL_REQ</identifier>
  </message_id>
  <description>requestst remote control from the SLM</description>
  <category>Local/Remote Control</category>
```

```
<originator originator_value="tsc"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>PAUSE</identifier>
</message_id>
<description>requests the SLM to pause</description>
<category>Control Flow</category>
<originator originator_value="tsc"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>RESUME</identifier>
</message_id>
<description>requests a paused SLM to resume</description>
<category>Control Flow</category>
<originator originator_value="tsc"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>ESTOP</identifier>
</message_id>
<description>instructs the SLM to perform an emergency stop</description>
<category>Control Flow</category>
<originator originator_value="tsc"/>
<message_type message_type_value="required"/>
<priority>0</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>STATUS_REQ</identifier>
</message_id>
<description>requests status information from the SLM</description>
<category>Control Flow 2</category>
<originator originator_value="tsc"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<message_data_syntax>
<argument repetition_type="zero_or_one">
<argument_id>
<identifier>sMainType</identifier>
```

```
</argument_id>
  <data_type data_type_value="string"/>
  <!--no default or value_range-->
</argument>
<argument repetition_type="zero_or_more">
  <argument_id>
    <identifier>sSubType</identifier>
  </argument_id>
  <data_type data_type_value="string"/>
  <!--no default or value_range-->
</argument>
</message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>ALARM_ON</identifier>
  </message_id>
  <description>SLM notifies TSC of a problem</description>
  <category>Control Flow 2</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <argument repetition_type="one">
      <argument_id>
        <identifier>sAlarmID</identifier>
      </argument_id>
      <data_type data_type_value="string"/>
      <!--no default or value_range-->
    </argument>
    <argument repetition_type="zero_or_one">
      <argument_id>
        <identifier>sAlarmText</identifier>
      </argument_id>
      <data_type data_type_value="string"/>
      <!--no default or value_range-->
    </argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>REMOTE_CTRL_DENIED</identifier>
  </message_id>
  <description>SLM does not accept remote control.</description>
  <category>Local/Remote Control</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
```

```
<identifier>DenialMessage</identifier>
  </argument_id>
  <complex complex_type="message" complex_level="full" required_category="error message"/>
</complex_argument>
</message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>REMOTE_CTRL_ACCEPTED</identifier>
  </message_id>
  <description>SLM accepts remote control</description>
  <category>Local/Remote Control</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <!--no message_data_type-->
</message>
<message>
  <message_id>
    <identifier>LOCAL_CTRL_DENIED</identifier>
  </message_id>
  <description>SLM does not accept local control.</description>
  <category>Local/Remote Control</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
        <identifier>DenialMessage</identifier>
      </argument_id>
      <complex complex_type="message" complex_level="full" required_category="error message"/>
    </complex_argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>LOCAL_CTRL_ACCEPTED</identifier>
  </message_id>
  <description>SLM accepts local control</description>
  <category>Local/Remote Control</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <!--no message_data_type-->
</message>
<message>
  <message_id>
    <identifier>OP_STARTED</identifier>
  </message_id>
  <description>SLM notifies TSC that it has started processing</description>
```

```
<category>required</category>
<originator originator_value="slm"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>OP_COMPLETED</identifier>
</message_id>
<description>SLM has finished processing</description>
<category>Control Flow 2</category>
<originator originator_value="slm"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>ABORT_ACCEPTED</identifier>
</message_id>
<description>SLM accepts abort request</description>
<category>Control Flow 2</category>
<originator originator_value="slm"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>ABORT_COMPLETED</identifier>
</message_id>
<description>SLM reports that it has completed aborting interaction</description>
<category>Control Flow 2</category>
<originator originator_value="slm"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
<message_id>
<identifier>NO_STATUS</identifier>
</message_id>
<description>SLM reports that there is no status information available</description>
<category>Control Flow</category>
<originator originator_value="slm"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<!--no message_data_syntax-->
</message>
<message>
```

```
<message_id>
  <identifier>ALARM_OFF</identifier>
</message_id>
<description>SLM notifies TSC that alarm situation is over</description>
<category>Control Flow 2</category>
<originator originator_value="slm"/>
<message_type message_type_value="required"/>
<priority>1</priority>
<message_data_syntax>
  <argument repetition_type="one">
    <argument_id>
      <identifier>sAlarmID</identifier>
    </argument_id>
    <data_type data_type_value="string"/>
    <!--no default or value_range-->
  </argument>
</message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>STATUS</identifier>
  </message_id>
  <description>SLM reports staus information</description>
  <category>Control Flow 2</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <argument repetition_type="one_or_more">
      <argument_id>
        <identifier>sReturnValue</identifier>
      </argument_id>
      <data_type data_type_value="string"/>
      <!--no default or value_range-->
    </argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>ABORT_DENIED</identifier>
  </message_id>
  <description>SLM does not accept abort request</description>
  <category>Control Flow 2</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
        <identifier>DenialMessage</identifier>
      </argument_id>
```

```
<complex complex_type="message" complex_level="full" required_category="error message"/>
  </complex_argument>
</message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>OP_RESULT</identifier>
  </message_id>
  <description>SLM reports results</description>
  <category>Control Flow 2</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <argument repetition_type="one_or_more">
      <argument_id>
        <identifier>ReturnValue</identifier>
      </argument_id>
      <!--no data_type_value -->
      <!--no default or value_range-->
    </argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>OP_DENIED</identifier>
  </message_id>
  <description>The SLM does not accept a processing request</description>
  <category>Control Flow 2</category>
  <originator originator_value="slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <message_data_syntax>
    <complex_argument>
      <argument_id>
        <identifier>DenialMessage</identifier>
      </argument_id>
      <complex complex_type="message" complex_level="full" required_category="error message"/>
    </complex_argument>
  </message_data_syntax>
</message>
<message>
  <message_id>
    <identifier>ACK</identifier>
  </message_id>
  <description>positive acknowledgement of message</description>
  <category>Flow Control</category>
  <originator originator_value="tsc_or_slm"/>
  <message_type message_type_value="required"/>
  <priority>1</priority>
  <!--no message_data_syntax-->
```

```
</message>
<interaction>
  <interaction_id>
    <identifier>Local/Remote Control</identifier>
  </interaction_id>
  <description>negotiation of local and remote control of slm</description>
  <interaction_type interaction_type_value="required_primary"/>
  <interaction_states>
    <!-- 1 or more state references-->
    <state_reference>
      <state_id>
        <identifier>LOCAL</identifier>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier>REMOTE</identifier>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier>LOCAL CTRL REQUESTED</identifier>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier>REMOTE CTRL REQUESTED</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
</interaction>
<interaction>
  <interaction_id>
    <identifier>Control Flow</identifier>
  </interaction_id>
  <description>use of slm</description>
  <interaction_type interaction_type_value="required_primary"/>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>POWERED UP</identifier>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier/>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier>INITING</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
</interaction>
```

```
</state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>DLE</identifier>
  </state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>CONFIGURING</identifier>
  </state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>NORMAL OPERATION</identifier>
  </state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>CLEARING</identifier>
  </state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>PAUSING</identifier>
  </state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>PAUSED</identifier>
  </state_id>
</state_reference>
</interaction_states>
</interaction>
<interaction>
  <interaction_id>
    <identifier>Processing</identifier>
  </interaction_id>
  <description>slm specific processing</description>
  <interaction_type interaction_type_value="required_secondary"/>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>PROCESSING REQUESTED</identifier>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier>PROCESSING</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
</interaction>
```

```
</state_reference>
<state_reference>
  <state_id>
    <identifier>ALARM</identifier>
  </state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>TERMINATED</identifier>
  </state_id>
</state_reference>
</interaction_states>
</interaction>
<interaction>
  <interaction_id>
    <identifier>Abort</identifier>
  </interaction_id>
  <description>Early termination of secondary interaction</description>
  <interaction_type interaction_type_value="required_secondary"/>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>ABORT REQUESTED</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>ABORTING</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>TERMINATED</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
</interaction>
<interaction>
  <interaction_id>
    <identifier>Status</identifier>
  </interaction_id>
  <description>status enquiry</description>
  <interaction_type interaction_type_value="required_secondary"/>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>STATUS REQUESTED</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
</interaction>
```

```
</state_id>
</state_reference>
<state_reference>
  <state_id>
    <identifier>TERMINATED</identifier>
  </state_id>
</state_reference>
</interaction_states>
</interaction>
<interaction>
  <interaction_id>
    <identifier>Alarm</identifier>
  </interaction_id>
  <description>asynchronous problem</description>
  <interaction_type interaction_type_value="required_secondary"/>
  <interaction_states>
    <state_reference>
      <state_id>
        <identifier>ALARM</identifier>
      </state_id>
    </state_reference>
    <state_reference>
      <state_id>
        <identifier>TERMINATED</identifier>
      </state_id>
    </state_reference>
  </interaction_states>
</interaction>
<maintenance>
  <maintenance_id>
    <identifier>ExerciseArm</identifier>
  </maintenance_id>
  <description>move robot arm around</description>
  <maintenance_type maintenance_type_value="verification"/>
  <performance_date_time>
    <date_and_time>
      <date>
        <day>29</day>
        <month>2</month>
        <year>2002</year>
      </date>
      <time>
        <hour>13</hour>
        <minute>48</minute>
        <second>52</second>
        <millisecond>0</millisecond>
      </time>
    </date_and_time>
  </performance_date_time>
  <performance_repetition>
    <regular regular_value="weekly"/>
  </performance_repetition>
</maintenance>
</performance_date_time>
</performance_repetition>
```

```
</performance_repetition>
<maintenance_tasks>
  <command_reference>
    <command_id>
      <identifier>MoveArm</identifier>
    </command_id>
  </command_reference>
  <command_reference>
    <command_id>
      <identifier>HomeArm</identifier>
    </command_id>
  </command_reference>
</maintenance_tasks>
</maintenance>
<physical_characteristics>
  <slm_location>
    <location>Somewhere here, room x</location>
  </slm_location>
  <geometry>
    <footprint>
      <floor_coordinate>
        <x>0</x>
        <y>0</y>
      </floor_coordinate>
      <floor_coordinate>
        <x>2500</x>
        <y>2500</y>
      </floor_coordinate>
      <floor_coordinate>
        <x>1760</x>
        <y>1500</y>
      </floor_coordinate>
      <floor_coordinate>
        <x>1000</x>
        <y>500</y>
      </floor_coordinate>
    </footprint>
    <height>1756</height>
  </geometry>
  <slm_devices>
    <device>
      <device_id>
        <identifier>Arm</identifier>
      </device_id>
      <description>robotic arm used to transfer consumables</description>
    </device>
    <device>
      <device_id>
        <identifier>Encoder</identifier>
      </device_id>
      <description>encoder locating position of robotic arm</description>
    </device>
  </slm_devices>
</physical_characteristics>
```

```
</device>
</slm_devices>
</physical_characteristics>
<port>
  <port_id>
    <identifier>TCP_REMOTE</identifier>
  </port_id>
  <description>port on TSC SLM connects to</description>
  <port_type>
    <data_port>
      <port_access port_access_type="readwrite"/>
      <port_location>
        <location>ss-strdev-02.ws.cr.sandwich.pfizer.com:1114</location>
      </port_location>
    </data_port>
  </port_type>
</port>
<port>
  <port_id>
    <identifier>TCP_LOCAL</identifier>
  </port_id>
  <description>port on SLM SLM connects from</description>
  <port_type>
    <data_port>
      <port_access port_access_type="readwrite"/>
      <port_location>
        <location>ss-strdev-01.ws.cr.sandwich.pfizer.com:4001</location>
      </port_location>
    </data_port>
  </port_type>
</port>
<!--material ports-->
<port>
  <port_id>
    <identifier>Gripper</identifier>
  </port_id>
  <description>robotic gripper for moving test tubes</description>
  <port_type>
    <material_port availability="available">
      <port_category port_category_value="fixed_port"/>
      <port_location>
        <location>on robot</location>
      </port_location>
      <capacity>
        <material_category material_category_value="port"/>
        <measure>
          <measure_value>1</measure_value>
          <!--no measure unit-->
        </measure>
      </capacity>
      <current_contents>
```

```
<content>
  <port_reference>
    <port_id>
      <identifier/>
    </port_id>
  </port_reference>
  <!--no measure-->
</content>
</current_contents>
</material_port>
</port_type>
</port>
<!--resources are constant-->
<!--only need to be declared once-->
<resource>
  <resource_id>
    <identifier>Solvent</identifier>
  </resource_id>
  <description>solvent for dilutions</description>
  <category>reagent</category>
</resource>
<resource>
  <resource_id>
    <identifier>Sample</identifier>
  </resource_id>
  <description>sample to be treated</description>
  <category>sample</category>
</resource>
<state>
  <state_id>
    <identifier>OPERATING</identifier>
  </state_id>
  <description>top level state of Control Flow</description>
  <required_parent_state/>
  <entry/>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>ESTOP</identifier>
        </message_id>
      </message_reference>
    </exit_event>
    <next_state>
      <state_reference>
        <state_id>
          <identifier>ESTOPPED</identifier>
        </state_id>
      </state_reference>
    </next_state>
  </exit>
```

```
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>STATE_CHANGED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
</next_state>
  <state_reference>
    <state_id>
      <identifier>ESTOPPED</identifier>
    </state_id>
  </state_reference>
</next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>ESTOPPED</identifier>
  </state_id>
  <description>top level state of Control Flow</description>
  <required_parent_state/>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>ESTOP</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>OPERATING</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>STATE_CHANGED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>OPERATING</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
```

```
</state_id>
  </state_reference>
</previous_state>
</entry>
<!--no exit (manual intervention required)-->
</state>
<state>
  <state_id>
    <identifier>POWERED UP</identifier>
  </state_id>
  <description>default state of Control Flow</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>OPERATING</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry/>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>INIT</identifier>
        </message_id>
      </message_reference>
    </exit_event>
    <next_state>
      <state_reference>
        <state_id>
          <identifier>INITING</identifier>
        </state_id>
      </state_reference>
    </next_state>
  </exit>
</state>
<state>
  <state_id>
    <identifier>INITING</identifier>
  </state_id>
  <description>slm initialises itself</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>OPERATING</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
```

```
<message_id>
  <identifier>INIT</identifier>
</message_id>
</message_reference>
</entry_event>
<previous_state>
  <state_reference>
    <state_id>
      <identifier>POWERED UP</identifier>
    </state_id>
  </state_reference>
</previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>STATE_CHANGED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>IDLE</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>IDLE</identifier>
  </state_id>
  <description>SLM waits to configure</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>OPERATING</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>STATE_CHANGED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
```

```
<state_reference>
  <state_id>
    <identifier>INITING</identifier>
  </state_id>
</state_reference>
</previous_state>
</entry>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>STATE_CHANGED</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>CLEARING</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>SETUP</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>CONFIGURING</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>CONFIGURING</identifier>
  </state_id>
  <description>SLM configures itself</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>OPERATING</identifier>
      </state_id>
    </state_reference>
```

```
</required_parent_state>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>SETUP</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>IDLE</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>STATE_CHANGED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>NORMAL OPERATION</identifier>
  </state_id>
  <description>SLM is ready for secondary interactions</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>OPERATING</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>STATE_CHANGED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
  </entry>
</state>
```

```
</message_id>
</message_reference>
</entry_event>
<previous_state>
  <state_reference>
    <state_id>
      <identifier>CONFIGURING</identifier>
    </state_id>
  </state_reference>
</previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>CLEAR</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>CLEARING</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>CLEARING</identifier>
  </state_id>
  <description>SLM is clearing its buffers etc.</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>OPERATING</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>CLEARING</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
```

```
<identifier>NORMAL OPERATION</identifier>
  </state_id>
</state_reference>
</previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>STATE_CHANGED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>IDLE</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>PROCESSING REQUESTED</identifier>
  </state_id>
  <description>slm has accepted processing request</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>RUN_OP</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <!--no previous state within interaction-->
  </entry>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>OP_STARTED</identifier>
        </message_id>
      </message_reference>
```

```
</exit_event>
<next_state>
  <state_reference>
    <state_id>
      <identifier>PROCESSING</identifier>
    </state_id>
  </state_reference>
</next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>PROCESSING</identifier>
  </state_id>
  <description>slm is processing a processing request</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>OP_STARTED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>PROCESSING REQUESTED</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>OP_RESULT</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>PROCESSING</identifier>
        </state_id>
      </state_id>
    </previous_state>
  </entry>
```

```
</state_reference>
</previous_state>
</entry>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>ALARM_OFF</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>ALARM</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>OP_COMPLETED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>TERMINATED</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>ALARM_ON</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>ALARM</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
```

```
</state>
<state>
  <state_id>
    <identifier>ALARM</identifier>
  </state_id>
  <description>slm has notified tsc of problem</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>ALARM_ON</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>PROCESSING</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>ALARM_ON</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <!--no previous state specified for asynchronous alarms -->
  </entry>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>ALARM_OFF</identifier>
        </message_id>
      </message_reference>
    </exit_event>
    <!--no next state specified for asynchronous alarms -->
  </exit>
</state>
<state>
```

```
<state_id>
  <identifier>ABORT REQUESTED</identifier>
</state_id>
<description>slm has accepted abort request</description>
<required_parent_state>
  <state_reference>
    <state_id>
      <identifier>NORMAL OPERATION</identifier>
    </state_id>
  </state_reference>
</required_parent_state>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>ABORT_REQ</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <!--no previous state specified-->
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>ABORT_ACCEPTED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>ABORTING</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>ABORT_DENIED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>TERMINATED</identifier>
      </state_id>
    </state_reference>
```

```
</next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>ABORTING</identifier>
  </state_id>
  <description>slm is processing abort request</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>ABORT_ACCEPTED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>ABORT REQUESTED</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>ABORT_COMPLETED</identifier>
        </message_id>
      </message_reference>
    </exit_event>
    <next_state>
      <state_reference>
        <state_id>
          <identifier>TERMINATED</identifier>
        </state_id>
      </state_reference>
    </next_state>
  </exit>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
```

```
<identifier>ABORT_DENIED</identifier>
  </message_id>
</message_reference>
</exit_event>
<next_state>
  <state_reference>
    <state_id>
      <identifier>TERMINATED</identifier>
    </state_id>
  </state_reference>
</next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>PAUSING</identifier>
  </state_id>
  <description>slm is halting active secondary interactions</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>PAUSE</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <!--no previous state specified-->
  </entry>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>STATE_CHANGED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <!--no previous state specified-->
  </entry>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>RESUME</identifier>
        </message_id>
```

```
</message_reference>
</exit_event>
<!--returns to unspecified state-->
</exit>
</state>
<state>
  <state_id>
    <identifier>PAUSED</identifier>
  </state_id>
  <description>the slm has paused all secondary interactions</description>
  <required_parent_state>
    <state_reference>
      <state_id>
        <identifier>NORMAL OPERATION</identifier>
      </state_id>
    </state_reference>
  </required_parent_state>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>STATE_CHANGED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>PAUSING</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
</state>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>RESUME</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <!--returns to unspecified state-->
</exit>
</state>
<state>
  <state_id>
    <identifier>STATUS REQUESTED</identifier>
  </state_id>
  <description>the slm has accepted a status request</description>
  <required_parent_state/>
  <entry>
```

```
<entry_event>
  <message_reference>
    <message_id>
      <identifier>STATUS_REQ</identifier>
    </message_id>
  </message_reference>
</entry_event>
<!--no specified previous state in interaction-->
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>STATUS</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>TERMINATED</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>NO_STATUS</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>TERMINATED</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>LOCAL</identifier>
  </state_id>
  <description>slm is in local control (default state)</description>
  <required_parent_state/>
  <entry>
    <entry_event>
      <message_reference>
```

```
<message_id>
  <identifier>ESTOP</identifier>
</message_id>
</message_reference>
</entry_event>
<previous_state>
  <state_reference>
    <state_id>
      <identifier>REMOTE</identifier>
    </state_id>
  </state_reference>
</previous_state>
</entry>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>LOCAL_CTRL_ACCEPTED</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>LOCAL CTRL REQUESTED</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>REMOTE_CTRL_REQ</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>REMOTE CTRL REQUESTED</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>LOCAL CTRL REQUESTED</identifier>
  </state_id>
  <description>slm has accepted local control request</description>
```

```
<required_parent_state/>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>LOCAL_CTRL_REQ</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>REMOTE</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>LOCAL_CTRL_DENIED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>REMOTE</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>LOCAL_CTRL_ACCEPTED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>LOCAL</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
</state>
<state>
```

```
<state_id>
  <identifier>REMOTE CTRL REQUESTED</identifier>
</state_id>
<description>slm has accepted remote control request</description>
<required_parent_state/>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>REMOTE_CTRL_REQ</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>LOCAL</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>REMOTE_CTRL_DENIED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>LOCAL</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>REMOTE_CTRL_ACCEPTED</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>REMOTE</identifier>
      </state_id>
    </state_reference>
```

```
</next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>REMOTE</identifier>
  </state_id>
  <description>slm is under remote control</description>
  <required_parent_state/>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>REMOTE_CTRL_ACCEPTED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>REMOTE_CTRL_REQUESTED</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <exit>
    <exit_event>
      <message_reference>
        <message_id>
          <identifier>LOCAL_CTRL_REQ</identifier>
        </message_id>
      </message_reference>
    </exit_event>
  <next_state>
    <state_reference>
      <state_id>
        <identifier>LOCAL_CTRL_REQUESTED</identifier>
      </state_id>
    </state_reference>
  </next_state>
</exit>
<exit>
  <exit_event>
    <message_reference>
      <message_id>
        <identifier>ESTOP</identifier>
      </message_id>
    </message_reference>
  </exit_event>
  <next_state>
    <state_reference>
```

```
<state_id>
  <identifier>LOCAL</identifier>
</state_id>
</state_reference>
</next_state>
</exit>
</state>
<state>
  <state_id>
    <identifier>TERMINATED</identifier>
  </state_id>
  <description>final state of secondary interactions</description>
  <required_parent_state/>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>ALARM OFF</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>ALARM</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>OP_COMPLETED</identifier>
        </message_id>
      </message_reference>
    </entry_event>
    <previous_state>
      <state_reference>
        <state_id>
          <identifier>PROCESSING</identifier>
        </state_id>
      </state_reference>
    </previous_state>
  </entry>
  <entry>
    <entry_event>
      <message_reference>
        <message_id>
          <identifier>ABORT_COMPLETED</identifier>
        </message_id>
```

```
</message_reference>
</entry_event>
<previous_state>
  <state_reference>
    <state_id>
      <identifier>ABORTING</identifier>
    </state_id>
  </state_reference>
</previous_state>
</entry>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>STATUS</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>STATUS REQUESTED</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
<entry>
  <entry_event>
    <message_reference>
      <message_id>
        <identifier>NO STATUS</identifier>
      </message_id>
    </message_reference>
  </entry_event>
  <previous_state>
    <state_reference>
      <state_id>
        <identifier>STATUS REQUESTED</identifier>
      </state_id>
    </state_reference>
  </previous_state>
</entry>
</state>

<system_variable>
  <variable_id>
    <identifier>ArmInitialised</identifier>
  </variable_id>
  <description>>false if the arm isn't initialised, true if it is</description>
  <data_type data_type_value="boolean"/>
  <variable_access_type variable_access_type_value="read_write"/>
```

```
<category>status</category>
<device_reference>
  <device_id>
    <identifier>Arm</identifier>
  </device_id>
</device_reference>
<initial_value>>false</initial_value>
<value_range>
  <range>
    <upper_limit>
      <measure>
        <measure_value>>true</measure_value>
        <!--no measure_unit-->
      </measure>
    </upper_limit>
    <lower_limit>
      <measure>
        <measure_value>>false</measure_value>
        <!--no measure_unit-->
      </measure>
    </lower_limit>
  </range>
</value_range>
<current_value>>true</current_value>
</system_variable>
</Device_Capability_Dataset>
```