5.4 The HSMS-SS state machine is illustrated in the diagram below.

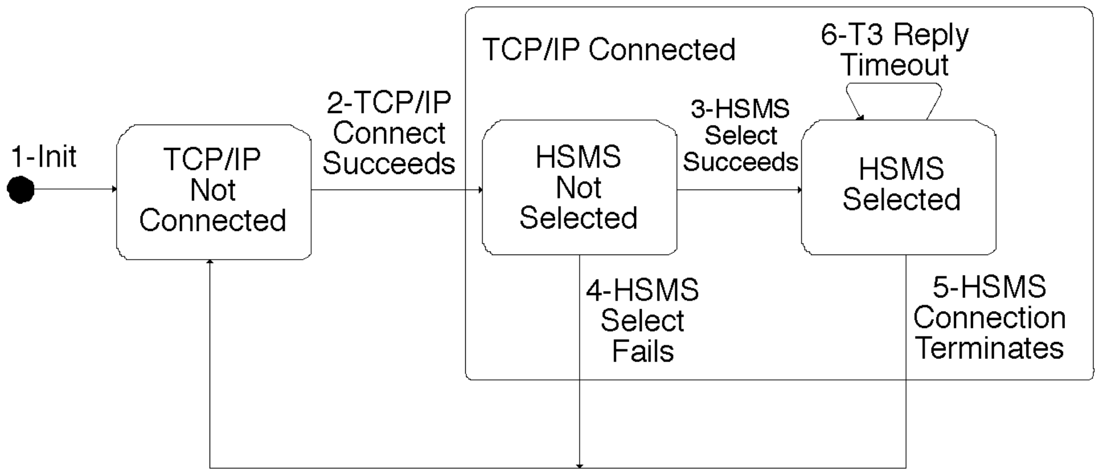

Figure 1

5.5 *State Transition Table for Passive Mode Connect*

Table 1 HSMS-SS Passive Mode Connect State Transitions

#	*Old State*	*New State*	*Trigger*	*Actions*
1	—	TCP/IP NOT CONNECTED	Initialization	
2	TCP/IP NOT CONNECTED	HSMS NOT SELECTED	TCP/IP Connect Succeeds: 1. TCP/IP "accept" succeeds.	Start T7 timeout.
3	HSMS NOT SELECTED	HSMS SELECTED	HSMS Select Succeeds: 1. Receive Select.req and decide to allow it.	1. Cancel T7 timeout; and 2. Send Select.rsp with zero SelectStatus.
4	HSMS NOT SELECTED	TCP/IP NOT CONNECTED	HSMS Select Fails: 1. T7 Timeout waiting for Select.req; or 2. Receive Select.req and decide to reject it and send Select.rsp with non-zero SelectStatus; or 3. Receive any HSMS message other than Select.req; or 4. Receive HSMS message length not equal to 10; or 5. Receive bad HSMS message header; or 6. T8 timeout waiting for TCP/IP; or 7. Other unrecoverable TCP/IP Error (entity-specific).	1. Close TCP/IP connection.

#	*Old State*	*New State*	*Trigger*	*Actions*
5	HSMS SELECTED	TCP/IP NOT CONNECTED	HSMS Connection Terminates: 1. Decide to terminate and send Separate.req; or 2. Receive Separate.req; or 3. T6 timeout waiting for Linktest.rsp; or 4. Receive HSMS message <10; or 5. Receive HSMS message length > maximum supported by entity; or 6. Receive bad HSMS message header; or 7. T8 timeout waiting for TCP/IP; or 8. Other uncorrectable TCP/IP Error (entity-specific).	1. Close TCP/IP connection.
6	HSMS SELECTED	HSMS SELECTED	T3 Timeout waiting for Data Reply Message.	1. Cancel the Data Transaction as appropriate (entity-specific) but do not terminate the TCP/IP connection; and 2. If entity is EQUIPMENT send SECS-II S9F9.

5.6 *State Transition Table for Active Mode Connect*

Table 2 HSMS-SS Active Mode Connect State Transitions

#	*Old State*	*New State*	*Trigger*	*Actions*
1	–	TCP/IP NOT CONNECTED	Initialization	
2	TCP/IP NOT CONNECTED	HSMS NOT SELECTED	TCP/IP Connect Succeeds: 1. Decide to connect.	1. TCP/IP Connect; and 2. Send Select.req; and 3. Start T6 timeout.
3	HSMS NOT SELECTED	HSMS SELECTED	HSMS Select Succeeds: 1. Receive Select.rsp with zero SelectStatus.	1. Cancel T6 timeout.
4	HSMS NOT SELECTED	TCP/IP NOT CONNECTED	HSMS Select Fails: 1. T6 Timeout waiting for Select.rsp; or 2. Receive Select.rsp with non-zero Select.Status; or 3. Receive any HSMS message other than Select.rsp; or 4. Receive HSMS message length not equal to 10; or 5. Receive bad HSMS message header; or 6. T8 timeout waiting for TCP/IP; or 7. Other unrecoverable TCP/IP Error (entity-specific).	1. Close TCP/IP connection; and 2. Start T5 Timeout.

#	*Old State*	*New State*	*Trigger*	*Actions*
5	HSMS SELECTED	TCP/IP NOT CONNECTED	HSMS Connection Terminates: 1. Decide to terminate and send Separate.req; or 2. Receive Separate.req; or 3. T6 timeout waiting for Linktest.rsp; or 4. Receive HSMS message length < 10; or 5. Receive HSMS message length > maximum supported by entity; or 6. Receive bad HSMS message header; or 7. T8 timeout waiting for TCP/IP; or 8. Other uncorrectable TCP/IP Error (entity-specific).	1. Close TCP/IP connection.
6	HSMS SELECTED	HSMS SELECTED	T3 Timeout waiting for Data Reply Message.	1. Cancel the Data Transaction as appropriate (entry-specific) but do not terminate the TCP/IP connection; and 2. If entity is EQUIPMENT, send SECS-II S9F9.

Table 3 When HSMS Transactions are Allowed

HSMS Transition	*Allowed in State(s)*	*Who Initiates Transaction?*
Select	HSMS Not Selected	Active Entity
Link Test	HSMS Selected	Either Entity
Data	HSMS Selected	Either Entity
Separate	HSMS Selected	Either Entity

6 HSMS-SS Use of TCP/IP

6.1 As defined in HSMS.

7 HSMS-SS Procedures

7.1 *Select Procedure* — The Select Procedure shall only be initiated by the entity establishing the TCP/IP connection in active mode. The Passive Mode Entity shall not initiate the Select Procedure.

7.1.1 The Select Procedure is only permitted in the NOT SELECTED state. It uses a SessionID value of 0xFFFF and implies that all device IDs are available for communication. Immediately following any Select Procedure which fails to complete successfully with a zero Select Status, each Entity must close the TCP/IP connection and transit to the NOT CONNECTED state.

7.2 *Data Procedure* — The Data Procedure is as defined in HSMS Generic Services. Note that any SessionID value that corresponds with a DeviceID supported by the Local Entity is valid as long as the Local Entity is in the SELECTED state.

7.3 *Deselect Procedure* — Deselect shall not be used in an HSMS-SS implementation. Communication is ended using the Separate Procedure.

7.4 *Linktest Procedure* — As defined by HSMS. Under HSMS-SS, the use of Linktest is strictly limited to the SELECTED state.

7.5 *Reject Procedure* — The Reject Procedure is optional in HSMS communications. Note, however, that any situation which would require the use of the Reject as described in HSMS Generic Services shall be treated as a communications failure in implementations not supporting reject. Specifically, the TCP/IP connection is immediately closed.

7.6 *Separate Procedure* — Separate shall always use SessionID 0xFFFF (binary, all ones). In HSMS-SS, the Separate.req is valid only in the TCP/IP CONNECTED state and its substates. After either initiating or receiving a Separate.req message, the entity shall immediately close the TCP/IP connection and transit to the TCP/IP NOT CONNECTED state.

7.7 *Communications Failures* — As defined by HSMS. Note that, in addition to the communications failures defined under HSMS, any violation of the restrictions defined in prior sections of this document are also to be treated as communication failures.

8 HSMS-SS Message Format

8.1 *Session ID* — In HSMS-SS Data Messages, the high-order bit of Session ID is zero, and the low-order 15 bits contain Device ID, a 15-bit unsigned integer value, which occupies the low-order 7 bits (bits 6-0) of byte 0 and all of byte 1 of the header. Device ID is a property of the equipment, and can be viewed as a logical identifier associated with a physical device or sub-entity within the equipment. The precise meaning of "device" or "sub-entity" is equipment-defined. A unit of equipment must have at least one Device ID. Equipment which contains several devices may define a unique Device ID for each device.

In HSMS-SS Control Messages, Session ID will always assume the special value 0xFFFF (all one bits).

8.2 *PType* — All HSMS-SS messages are PType 0 (SECS II encoded) as defined in HSMS.

8.3 *SType* — Only HSMS-defined STypes are permitted in HSMS-SS. User-defined SType messages are not permitted.

9 Special Considerations

9.1 *Multiblock Messages* — For each SECS-II message, the SECS-II standard defines whether that message should be transmitted in SECS-I as a single-block message or as a multiblock message. This distinction becomes unimportant with HSMS, which transmits all messages in the same fashion. However, to be compatible with older SECS-I applications, when an HSMS application sends a SECS-II message defined as single block, the HSMS Message Length should not exceed 254 bytes (10 byte header plus 244 text bytes).

10 HSMS-SS Documentation

10.1 An HSMS-SS implementation is required to document the following information in addition to the information required by HSMS.

1. The number of deviceIDs supported and their specific values.
2. Whether or not the implementation supports the normal or the restricted procedure for terminating communications.
3. The setting of the host vs. equipment parameter.

10.2 *Host vs. Equipment* — Many applications using SECS-II will need to designate one end of the communication link as "Equipment" and the other end as "Host." HSMS-SS itself does not require configuring of "Host" and "Equipment," but this parameter may be included in configuration where needed. HSMS can also be used in applications where the distinction between Host and Equipment is not used.

RELATED INFORMATION 1
APPLICATION NOTES

NOTE: This related information is not an official part of SEMI E37.1 and is not intended to modify or supercede the official standard. Publication was authorized by full letter ballot. Determination of the suitability of the material is solely the responsiblity of the user.

R1-1

R1-1.1 An entity may have more than one session. If a unit of equipment can be divided into two or more logical sub-equipments, such as process chambers, or process resources, each of these may have separated session ID. If two or more physical sub-equipments are controlled by an equipment controller or communicated to through a TCP/IP network device, each sub-equipment may have a separate session ID. Such a session can be established once HSMS is selected. Each session ID corresponds to a device ID.

R1-1.2 If a unit of equipment has more than one sub-equipment or process resource (e.g. process modules) but the sub-equipment share a common resource (e.g. transfer subsystem), it is not recommended that each sub-equipment have an independent session to communicate with host. Even if the host requests an action to the shared subsystem with a session identifier specific for a sub-equipment, the shared subsystem may not always serve for the sub-equipment. Since the host expects the shared subsystem to perform the service for the sub-equipment, an error will occur if it is not possible to perform the service for the designated sub-equipment. See following figure.

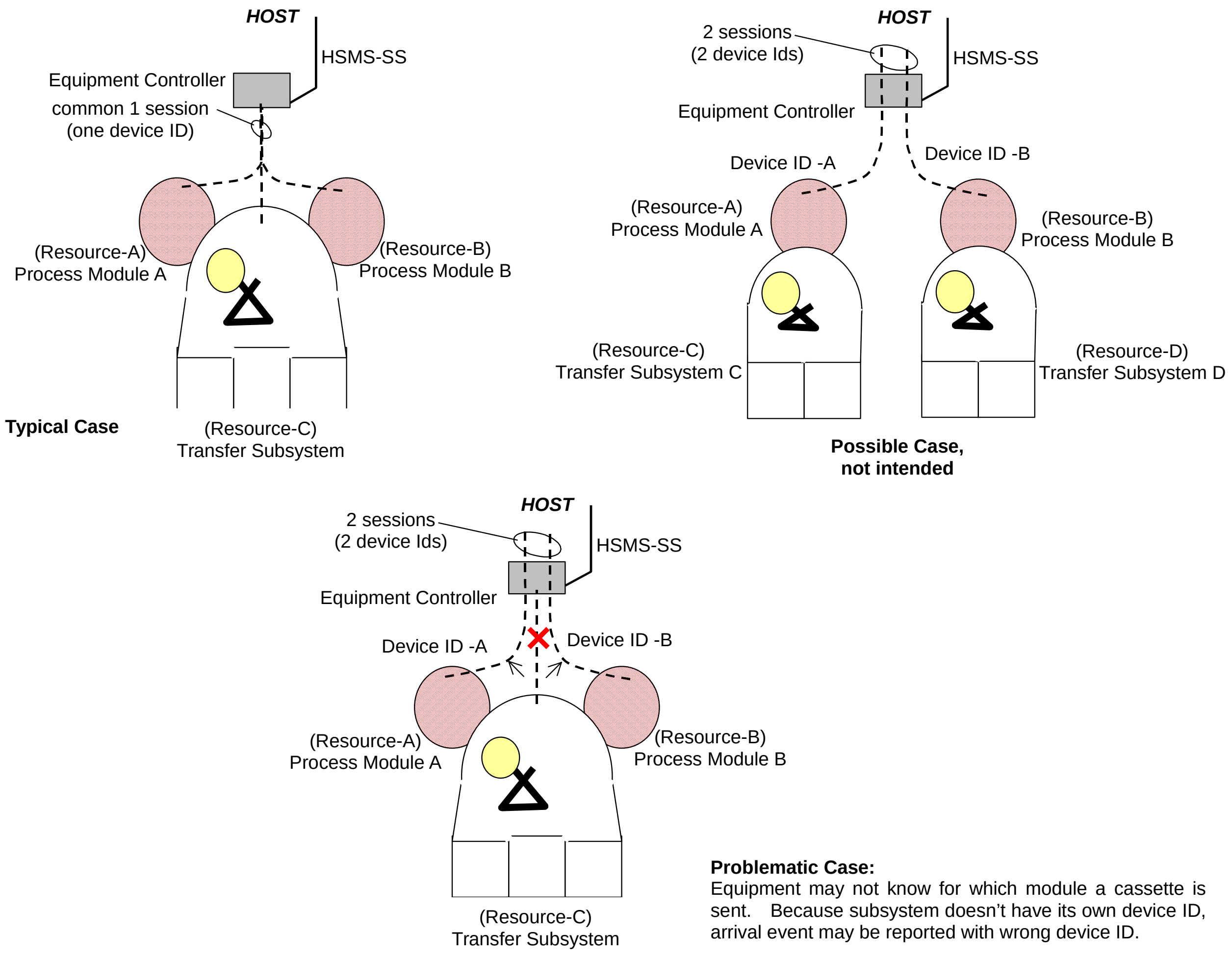

Figure R1-1

R1-2 Multiple HSMS Connections

R1-2.1 Typically, an Equipment will accept only one Host Connection.

R1-2.2 A Host may connect to several units of Equipment, so the Host may have several simultaneously active Connections (each to one Equipment).

R1-2.3 A Cell Controller (or similar entity) might have one Connection by which the Cell Controller appears as “Equipment” to the Factory Host Computer, as well as several Connections by which the Cell Controller appears as “Host” to Equipment.

R1-3 Equipment Support for Multiple Hosts

R1-3.1 HSMS requires Equipment to accept at least one active Connection, and does not require the equipment to support access by multiple concurrent Hosts. That is, if the Equipment has already accepted a Host Connection, but a Host (the same or a different Host) attempts a second Connection, the Equipment will immediately terminate that second Connection attempt.

R1-3.2 For specialized applications, an equipment could accept more than one Host Connection. Coordination of activity by multiple hosts is equipment-defined.

SEMI E37.2-95 (Reapproved 0303)
HIGH-SPEED SECS MESSAGE SERVICES GENERAL SESSION (HSMS-GS)

This standard was technically approved by the Global Information and Control Committee and is the direct responsibility of the North American Information and Control Committee. Current edition approved by the North American Regional Standards Committee on October 25, 2002. Initially available at www.semi.org December 2002; to be published March 2003. Originally published in 1995.

1 Purpose

1.1 HSMS-GS is intended to support the needs of complex systems containing multiple independently accessible subsystems such as cluster tools or track systems. Specifically, procedures are defined to permit access to any individual subsystem or set of subsystems within any complex system.

2 Scope

2.1 High-Speed SECS Message Services General Session (HSMS-GS) is a subsidiary standard to High-Speed SECS Message Services (HSMS) Generic Services.

NOTICE: This standard does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this standard to establish appropriate safety and health practices and determine the applicability of regulatory or other limitations prior to use.

3 Referenced Standards

3.1 *SEMI Standards*

SEMI E4 — SEMI Equipment Communications Standard 1 Message Transfer (SECS-I)

SEMI E5 — SEMI Equipment Communications Standard 2 Message Content (SECS-II)

SEMI E37 — High-Speed SECS Message Services (HSMS) Generic Services

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

4 Terminology

4.1 *Definitions*

4.1.1 *Selected Entity List* — a list of session entities currently selected for communication on a given TCP/IP connection.

4.1.2 *Selection Count* — the number of sessions opened by an HSMS Select procedure and not yet ended by an HSMS Deselect or Separate procedure.

4.1.3 *Session Entity* — an individually selectable entity within an HSMS-GS system.

4.1.4 *Session Entity ID* — a 16-bit identifier for a Session-Entity.

4.1.5 *SessionEntityList* — a list of all available session entities within an HSMS-GS system and associated with a particular IP address and port number.

4.2 In addition, all definitions for HSMS Generic Services apply.

4.3 Note that the terms HSMS and HSMS Generic Services both refer to the HSMS Generic Services standard definition (SEMI E37).

5 HSMS-GS Overview and State Machine

5.1 HSMS-GS provides a set of definitions which permit the individual subentities (e.g., subsystems) of complex entities (e.g., systems) to be separately accessible during HSMS procedures. HSMS-GS defines no new procedures or message types beyond HSMS Generic Services to provide these services. It does, however, require extensions to the HSMS State machine, in the form of additional state transition definitions and additional state information, which are used by the extended state machine which must be maintained by an HSMS-GS implementation to support the extended state machine. The additional information consists of the following:

1. The Session Entity List
2. The Selected Entity List
3. The Selection Count

5.2 The Session Entity List consists of the set of all Session Entities having individual accessibility within the HSMS-GS entity. The scope of this list is normally the entire HSMS-GS entity. HSMS-GS, however, does not require this scope: the supplier may provide access to HSMS-GS entity through more than one well known port and provide a different Session Entity List for each.

5.3 A Session Entity is any individually addressable subentity within the HSMS-GS entity: for example, a Session Entity may be an individual sub-device in a track system or cluster tool, or may be an individual service provider, such as a data server, within an entity.

HSMS-GS only provides the conventions for identifying Session Entities. It places no restrictions on the individual supplier as to what constitues a SessionEntity: the supplier must determine what is the most appropriate for the particular implementation.

5.4 The Selected Entity List is the list of Session Entities actually selected for access on a given TCP/IP connection. When the TCP/IP connection is established (CONNECTED state entered), an empty Selected Entity List is created. Each time, the Select Procedure is used to select a Session Entity, its Session Entity ID is added to the Selected Entity List created for that TCP/IP connection. Each time the Session Entity is deselected via the Deselect or Separate procedures, its Session Entity ID is removed from the Selected Entity List. At any given time on any given TCP/IP connection endpoint, HSMS Data Messages will only be accepted by an entity if the SessionID of the Data Message is equal to any Session Entity ID in the Selected Entity List.

5.5 The Selection Count is simply the number of Session Entity IDs in the Selected Entity List. Its value affects the behavior of the state machine: the transition to NOT SELECTED from SELECT can only take place when the SELECTION COUNT is zero.

5.6 Note that the above lists and count are defined for the purpose of explaining state machine operation only. There is no requirement that any of the above lists and count be explicitly implemented.

5.7 *HSMS-GS State Machine* — The HSMS-GS state machine is the same as the HSMS Generic Services state machine with the addition of the two new state transitions (#6 and #7) and the use of the Selection Count in the other transitions as described in the state transition table below.

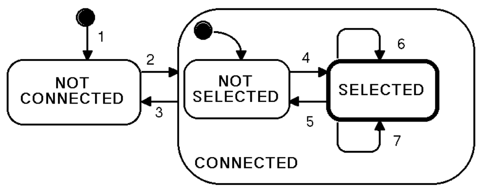

Figure 1

5.8 *State Descriptions* — The state descriptions are the same as HSMS Generic Services.

5.9 *State Transition Table* — The state transition table is almost identical with the state transition table for HSMS Generic Services. For convenience, however, the entire table is reproduced and extended here, not just those areas which differ from it.

Table 1

#	*Current State*	*Trigger*	*New State*	*Actions*	*Comment*
1	...	Local entity specific preparation for TCP/IP communication.	NOT CONNECTED	Local entity-specific.	Action depends on connection procedure to be used: active or passive.
2	NOT CONNECTED	A TCP/IP connection is established for HSMS communication.	CONNECTED - NOT SELECTED	Set Selection Count = 0 and create empty Selected Entity List.	none
3	CONNECTED	Breaking of TCP Connection.	NOT CONNECTED	Local entity-specific.	none
4	NOT SELECTED	Successful completion of HSMS Select Procedure.	SELECTED	Set Selection Count = 1 and add selected Session Entity to Selected Entity List.	none
5	SELECTED	Deselect or Separate procedure resulting in Selection Count = 0.	NOT SELECTED	Local entity-specific.	See transition 7 below.
6	SELECTED	Successful completion of HSMS Select Procedure when Selection Count > 0.	SELECTED	Increment Selection Count and add selected Session Entity to Selected Entity List.	none
7	SELECTED	Successful completion of HSMS Deselect or Separate when Selection Count > 1.	SELECTED	Decrement Selection Count and remove selected Session Entity from Selected Entity List.	If this transition results in Selection Count = 0, immediately trigger state transition 5.

6 HSMS-GS Use of TCP/IP

6.1 There is no additional HSMS-GS specification for the use of TCP/IP beyond the above mentioned creation of the empty “Selected Entity List” upon entry into the CONNECTED state, NOT SELECTED substate.

7 HSMS-GS Specific Procedures

HSMS-GS provides further specification of the following procedures.

7.1 *Select Procedure* — The select procedure is permitted in both the NOT SELECTED state and the SELECTED state. The procedure for both the initiator and the responding entity is the same as HSMS Generic Services, with the following additional conditions:

1. If the responding entity contains no Session Entity in its Session Entity List whose ID matches the SessionID in the Select.req, a Select Status of No Such Entity is used in the Select.rsp.
2. If the responding entity is unable to provide access to the selected entity because it is usable on only a single TCP/IP connection at any one time and it is already in use on a different TCP/IP connection, a response code of Entity In Use is used in the Select.rsp.
3. If the responding entity finds the SessionID already in its Selected Entity List, a response code of Entity Selected is used in the Select.rsp.

7.1.1 The Select Status values referenced above are all defined in Section 8.

7.1.2 If none of the above is true, the Select completes successfully, and a Select Status of 0 is provided in the Select.rsp. Both entities will add the SessionID from the Select.req to the Selected Entity List.

7.2 *Data Procedure* — The Data Procedure is as defined in HSMS Generic Services. Note that the SessionID of any Data Message must match a SessionID in the SelectedEntityList. If a Data Message is received with a SessionID other than one from the Selected Entity List, a Reject.req message is sent in response by the receiving entity. The reason code will be Entity Not Selected.

7.3 *Deselect Procedure* — The Deselect procedure is restricted by the following conditions:

1. The SessionID must be in the Selected Entity List.
2. The corresponding SessionEntity is in a state which permits Deselect. This decision is local entity specific and not subject to the HSMS-GS.

7.3.1 If both of the above are true, then the Deselect can proceed. Assuming that the Deselect completes successfully, the SessionID is removed from the Selected Entity List, and the Selection Count is decremented. The transition to the NOT SELECTED state transpires only if the resulting Selection Count is equal to zero (i.e., an empty Selected Entity List).

7.4 *Linktest Procedure* — As defined by HSMS.

7.5 *Reject Procedure* — As defined by HSMS. Note, in particular, the use of Reject in response to certain data messages (above).

7.6 *Separate Procedure* — The Separate procedures and state transitions are subject to the same restrictions and conditions as the Deselect.

7.7 *Communications Failures* — As defined by HSMS. Note that any abrupt termination has the effect of deselecting all entities in the Selected Entity List.

8 HSMS-GS Message Format Issues

8.1 *Session ID* — In HSMS-GS Select, Data, Deselect, Reject, and Separate messages, the SessionID will equal the Session Entity ID of the target Session Entity which must equal the Session ID of a Session Entity contained in the Session Entity List. In the Linktest, it is 0xFFFF, as in HSMS.

8.2 *PType* — HSMS-GS messages are generally PType = 0 (SECS-II-encoded) as defined in HSMS. Although other PTypes are permitted, specific application domains may restrict the use of HSMS-GS to Ptype = 0.

8.3 *SType* — Only HSMS-defined STypes are permitted in HSMS-GS.

8.4 *Select/Deselect Status* — The following additional enumeration applies to the Select/Deselect status in HSMS-GS:

Table 2 SelectStatus

Value	*Description*
4	No Such Entity — Session ID does not correspond to any Session Entity ID available at this connection.
5	Entity In Use (by another connection) — Session Entity corresponding to session ID is not sharable connections and is already selected by another connection.
6	Entity Selected (by current connection) — Session entity corresponding to Session ID is already selected on current connection.

9 Special Considerations

9.1 There are no special considerations above and beyond those described in HSMS Generic Services.

10 HSMS-GS Documentation

10.1 An HSMS-GS implementation is required to document the following information in addition to the information required by HSMS.

1. The Session Entity List — specifically the number of Session Entities available for HSMS-GS communication and the value of their particular Session Entity ID’s.

RELATED INFORMATION 1
APPLICATION NOTES

NOTICE: This related information is not an official part of SEMI E37.2 and is not intended to modify or supercede the official standard. Publication was authorized by full letter ballot procedures. Determination of the suitability of the material is solely the responsibility of the user.

R1-1 Supporting Both HSMS-GS and HSMS-SS Simultaneously

R1-1.1 In certain applications, the equipment manufacturer may be faced with a requirement of providing both HSMS-GS and HSMS-SS communications interfaces. For example, a cluster tool may use HSMS-GS as the intra-tool communications and HSMS-SS as a GEM interface to the factory. However, implementing a communications facility that is simultaneously HSMS-SS- and HSMS-GS-compliant is straightforward.

R1-1.2 Assuming that one has implemented an HSMS-GS, a simple test can be added to the first Select.req received by the equipment. If it is a Select for any particular SessionID, then operate as HSMS-GS. If its value is a -1, then copy the contents of the Session Entity List corresponding TCP/IP IP address and port number into the Selected Entity List. The Selection Counter would be set to a special value, such as -1, to indicate selection in this manner. The Separate would provide the reverse function. If the passive entity is implemented as an HSMS-SS node and the active entity is an implementation that supports both modes of operation, it must be explicitly configured to initiate the Select with a Session ID of -1 and must have a Session Entity List containing the Device IDs of all the available sub-devices within the passive mode entity.

R1-1.3 In the local API for the case where the equipment must originate the select, a configuration parameter for the equipment could indicate which mode to use for a particular remote target.

SEMI E38-1296
CLUSTER TOOL MODULE COMMUNICATIONS (CTMC)

1 Purpose

1.1 Cluster tools fulfill a need for modularity and flexibility in semiconductor manufacturing equipment. This standard addresses the communications with and among modules within a cluster tool for automated control.

1.2 The module communications services defined here will enable standard interoperability of modules from independent suppliers. Together with other cluster tool standards, this is intended to result in the emergence of standards-compliant interoperable sub-systems. They should allow applications software to be developed which can assume the existence of these services and allow software products to be developed to offer them.

1.3 The adoption of the standards described will greatly reduce the effort required to integrate cluster tool components from independent suppliers. Compliance requires support of a minimal, but specific, set of standard services.

2 Scope

2.1 Cluster tool module communications address only communications with and among modules within a cluster tool and not the communications between the cluster tool and the factory. It is the modules and their interrelations which are modeled and not the cluster tool seen as a single equipment.

2.2 This standard does not specify cluster controller functionality (e.g., scheduling, human machine interface, cluster tool recipe editing, and management) but will enable development of standards in this area. The cluster controller is included only to the extent that it represents the entity or entities responsible for supervisory control of the modules in a cluster tool.

2.3 This standard identifies the communications services necessary to achieve automated control of independent transport, process, and cassette modules in a cluster tool. It defines the essential module architecture and the concepts and models on which the communications services are based.

2.4 The scope includes primary control services for material processing in process modules, material movement within the cluster tool, and material input/output with the factory. Support services also exist to enable recipe handling, resolution of exception conditions, event reporting, and data access.

2.5 A reliable communications environment is required for distributed control and is specified in supplementary standards.

2.6 This standard specifies the application of more general service standards as required within a cluster tool. This includes the limitations imposed by the cluster tool architecture and the fundamental functionality needed for compliance. The details of the general services, protocols, and communication environment elements are defined in the corresponding standards documents referenced.

2.7 This version of the standard is provisional because a number of the general services on which it relies are not yet standardized. The sections referring to these services have been omitted until they have become standards. These sections will be balloted separately with the corresponding general services and incorporated into this provisional standard.

2.8 The communication between the cluster tool as a single equipment and the factory is beyond the scope of cluster tool module communications standardized here and should follow the applicable SEMI standards (GEM, MMMS).

2.9 These standards place no restriction on where, within the cluster tool control architecture, this specific set of definitions is implemented. However, to have complete compliance to the standards in the area of the communications interface definitions, the entire layered structure as outlined in this document must be implemented.

3 Referenced Documents

3.1 *SEMI Standards*

SEMI E21 — Cluster Tool Module Interface: Mechanical Interface and Wafer Transport Standard

SEMI E22 — Cluster Tool Module Interface: Transport Module End Effector Exclusion Volume Standard

SEMI E30 — Generic Model for Communications and Control of SEMI Equipment (GEM)

SEMI E32 — Material Movement Management Standard

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services

4 Terminology

4.1 The following terms are those that are appropriate for describing the overall structure of cluster tools. These terms, in addition to the more specific terms located in the individual service, protocol, and communications environment standards, complete the glossary of terms required.

4.1.1 *agent* — an intelligent system within a factory that provides one or more service resources and uses the services of other agents. A generalization of host, equipment, cell, cluster, cluster module, station controller, work station. Agents are associated with a physical system or a collection of physical systems, including computer platforms.

4.1.2 *alarm* — an alarm is related to any abnormal situation on the equipment that may endanger people, equipment, or material being processed. [SEMI E30]

4.1.3 *atomic transfer* — the basic unit of movement. The transfer of a single material between two partners where only one change in ownership occurs.

4.1.4 *attached module* — any component of a cluster tool which mechanically attaches to the transport module at an interface flange. Specifically, the cassette, process, and docking modules are all attached modules. The attached module can be isolated from the transport module by an isolation valve controlled by the transport module. In addition, the attached module may also possess an additional isolation valve. The point of separation between the isolation valve on the transport module and the attached module is called the interface plane.

4.1.5 *am transfer job* — a material transfer job for an attached module specifying all criteria for receiving the material from, or sending the material to, a transport module.

4.1.6 *carrier (material)* — a device for the holding of material for various processing steps in semiconductor manufacturing. [SEMI E1]

4.1.7 *cassette module* — an attached module which provides the means of exchanging material between the intertool environment and the intratool environment of a particular cluster. There is no requirement that the material interface to the intertool environment with the use of cassettes.

4.1.8 *cm input/output transfer job* — a material transfer job for a cassette module, specifying all criteria for receiving the material from, or sending the material to, the factory.

4.1.9 *cluster controller* — the entity or entities responsible for supervisory control within a cluster tool, achieved through communicating with the various modules. The form of the cluster controller is not dictated. It may be a single platform, distributed, or incorporated with one or more modules.

4.1.10 *cluster tool* — an integrated, environmentally isolated, manufacturing system consisting of process, transport, and cassette modules mechanically linked together. There is no requirement that the modules come from the same supplier. [SEMI E21]

4.1.11 *decision authority* — an entity requiring to be notified of significant exception condition changes and which decides how to proceed to resolve abnormal situations related to recoverable error conditions. The decision authority may be represented by a supervisory controller interacting with an operator who may ultimately choose the recovery action.

4.1.12 *docking module* — a component for allowing the exchange, within a cluster tool, of material between multiple transport modules. The docking module presents an attached module interface to both transport modules and is modeled as a multi-port process module for intra-cluster tool communications.

4.1.13 *end effector* — a physical location attached to a transfer resource capable of holding material during an end-to-end material transfer.

4.1.14 *error condition* — an exception condition which is not an alarm and which may support recovery actions requested by a decision authority.

4.1.15 *exception condition* — a managed condition for reporting on and providing recovery from an abnormal situation in the equipment.

4.1.16 *factory* — entities outside the control domain of the cluster tool and from which material is received for processing and returned upon process completion. Also considered to be the intertool environment.

4.1.17 *factory transport resource* — n operator or a piece of equipment specialized in the transport of material from one equipment to another.

4.1.18 *form* — type of data representing information contained in an object attribute or service message parameter. The data types are detailed in Section 4.2.

4.1.19 *fundamental requirements* — the requirement for information and behavior that must be satisfied for compliance to a standard. Fundamental requirements apply to specific areas of application, objects, or services.

4.1.20 *handoff micro move* — an operation requested by an attached module and performed by a transport module to achieve all or part of a material handoff between them.

4.1.21 *interface flange* — the boundary plane separating an attached module and a single transport module.

4.1.22 *intertool environment* — defined relative to a particular cluster tool to be everywhere outside of the intratool environment.

4.1.23 *intertool material transfer job* — a transfer job in the cluster controller to receive material from, or send the material to, the factory. In automated transfer, it may be an element of a factory transfer job.

4.1.24 *intertool port resource* — a module resource associated with a particular cassette module which operates on behalf of the said module during material transfer between the cassette module and the intertool environment.

4.1.25 *intratool environment* — the entire environmentally isolated volume contained within a cluster tool.

4.1.26 *intratool port resource* — a module resource associated with a particular interface plane of a particular attached module which operates on behalf of the said module during material transfer between the transport module and the attached module.

4.1.27 *intratool material transfer job* — a transfer job in a cluster controller for transfer of material from one attached module to another through the linking transport module.

4.1.28 *isolation valve* — a mechanical component used at an interface plane to permit environmental isolation between the transport module and an attached module.

4.1.29 *material handoff* — the process by which material moves from the sending transfer partner to the receiving transfer partner.

4.1.30 *material location* — a physical position capable of holding a single material.

4.1.31 *material slot* — a material location in a carrier capable of holding a single material with an assigned identifier.

4.1.32 *module* — an independently operable unit that is part of a tool or system. [SEMI E21]

4.1.33 *module resource* — an entity which provides specific capabilities required of the module to be accessed or controlled externally. This includes the physical capability and the associated control and access management. It may contain material in one or more material locations.

4.1.34 *namespace* — a domain within which object identifiers are unique.

4.1.35 *port* — a point on the cluster at which a change of ownership of material occurs. A port is not itself a location but must have an associated location, such as an interface plane.

4.1.36 *process job* — a material processing job for a process module, specifying and tracking the processing to be applied to the material while in the module.

4.1.37 *process module* — an attached module which provides manufacturing value to material in a cluster tool.

4.1.38 *processing resource* — a module resource within a module which is independently capable of providing manufacturing value to material.

4.1.39 *protocol (communications)* — the message encoding used when transferring a message across some communication facility.

4.1.40 *recipe* — the pre-planned and reusable portion of the set of instructions, settings, and parameters under control of a processing agent that determines the processing environment seen by the material. Recipes may be subject to change between runs or processing cycles.

4.1.41 *recipe executor* — a component of a module that stores and executes recipes.

4.1.42 *recipe namespace* — a logical management domain with the responsibility for the storage and management of recipes. It ensures the uniqueness of recipe identifiers and provides services pertaining to recipes stored within that domain.

4.1.43 *recovery action* — an operation associated with an error condition with the aim of resolving the abnormal situation detected. It may supply information to the exception agent or request the exception agent to perform some activity.

4.1.44 *service* — the set of messages and definition of the behavior of a service provider that enables remote access to a particular functionality.

4.1.45 *service-provider* — the software control entity that is the provider of any of the related services.

4.1.46 *service-user* — the software control entity that is the user of any of the related services.

4.1.47 *transport module* — a module containing one or more transfer resources and a number of interface planes. It is capable of end-to-end transfer between any of the interface planes.

4.1.48 *tm transfer job* — a material transfer job for a transport module specifying all criteria for receiving the material from the source-attached module and sending the material to the destination module.

4.1.49 *transfer resource* — a module resource within a transport module independently capable of transferring material from one attached module to another.

4.2 *Data Type*

4.2.1 *form* — type of data: positive integer, unsigned integer, integer, enumerated, boolean, text, formatted text, structure, list, ordered list.

4.2.2 *positive integer* — may take the value of any positive whole number. Messaging protocol may impose a limit on the range of possible values.

4.2.3 *unsigned integer* — may take the value of any positive integer or zero. Messaging protocol may impose a limit on the range of possible values.

4.2.4 *integer* — may take on the value of any negative or unsigned integer. Messaging protocol may impose a limit on the range of possible values.

4.2.5 *enumerated* — may take on one of a limited set of possible values. These values may be given logical names, but they may be represented by any single-item data type.

4.2.6 *boolean* — may assume one of two possible values, equating to TRUE or FALSE.

4.2.7 *text* — a text string. Messaging protocol may impose restrictions, such as length or ASCII representation.

4.2.8 *formatted text* — a text string with an imposed format. This could be by position, by use of special characters, or both.

4.2.9 *structure* — a complex structure consisting of a specific set of items, of possibly mixed data types, in a specified arrangement.

4.2.10 *list* — a set of one or more items that are all of the same form (one of the above forms).

4.2.11 *ordered list* — a list for which the order in which items appear is significant.

5 Conventions

5.1 *Harel State Model*

5.1.1 This document uses the Harel State Chart notation to describe the dynamic behavior of the objects defined. An overview of this notation is presented in an Appendix of SEMI E30. The formal definition of this notation is presented in *Science of Computer Programming 8*, "Statecharts: A Visual Formalism for Complex Systems," by D. Harel, 1987.

5.1.2 Transition tables are provided in conjunction with the state diagrams to describe explicitly the nature of each state transition. A transition contains columns for Transition #, Current State, Trigger, New State, Action(s). The "trigger" (column 3) for the transition occurs while in the "current" state. The "actions" (column 5) include a combination of (1) actions taken upon exit of the current state, (2) actions taken upon entry of the new state, and (3) actions taken which are most closely associated with the transition. No differentiation is made.

5.2 *OMT Object Information Model* — The object models are presented using the Object Modeling Technique developed by Rumbaugh, James, et al., in "Object-Oriented Modeling and Design," Prentice Hall, Englewood Cliffs, NJ, ©1991. Overviews of this notation are provided in an appendix to SEMI E39.

5.3 *Object Attribute Representation* — The object information models for standardized objects will be supported by an attribute definition table with the following column headings:

Attibute Name	*Definition*	*Access*	*Rqmt*	*Form*
The formal text name of the attribute.	Description of the information contained.	RO or RW	Y or N	(see below)

The Access column uses RO (Read Only) or RW (Read and Write) to indicate the access that service-users have to the attribute.

A 'Y' or 'N' in the requirement (Rqmt) column indicates if this attribute must be supported in order to meet fundamental compliance for the service.

The Form column is used to indicate the format of the attribute. (See Section 4.1 for definitions.)

6 Overview

The Cluster Tool Module Communications standard specifies the communications services necessary to achieve automated control within a cluster tool. It defines the essential cluster tool architecture and the concepts and models on which the communications services are based.

Cluster tools provide a means of grouping a number of independent process steps into a single equipment. The primary function of a cluster tool is material processing. Material enters the cluster tool, undergoes a number of sequential process steps which adds value to the material, and then exits.

From a mechanical viewpoint, Cluster Tool Module Interface standards-compliant equipment has a physical structure where process modules and cassette modules are attached in a standardized way to some form of transport module, a robotic material handler operating in an isolated environment. Cassette modules provide the input and output of material for the cluster tool, usually in cassettes.

This communications standard enables the distributed control of these cluster tools, and of other cluster tools and multi-resource equipment which can be decomposed into process, transport, and cassette modules, whether environmentally isolated or not. A basic assumption is made that the primary human-machine interfaces are not located at the individual modules.

Communication and control services are specified among service-users and service-providers within a cluster tool. The service-providers are the controlled processing, transfer, intratool port, and intertool port resources in the appropriate modules. The service-users are the application entities responsible for supervisory control and data acquisition within the cluster tool.

The services are fully defined in terms of the service-providers, and as such, do not dictate the architecture of the service-user(s). An example of cluster tool control architecture is shown in Figure 1, where the service-user is a single cluster controller platform providing internal scheduling, human-machine interface, and standardized communications interface to the factory.

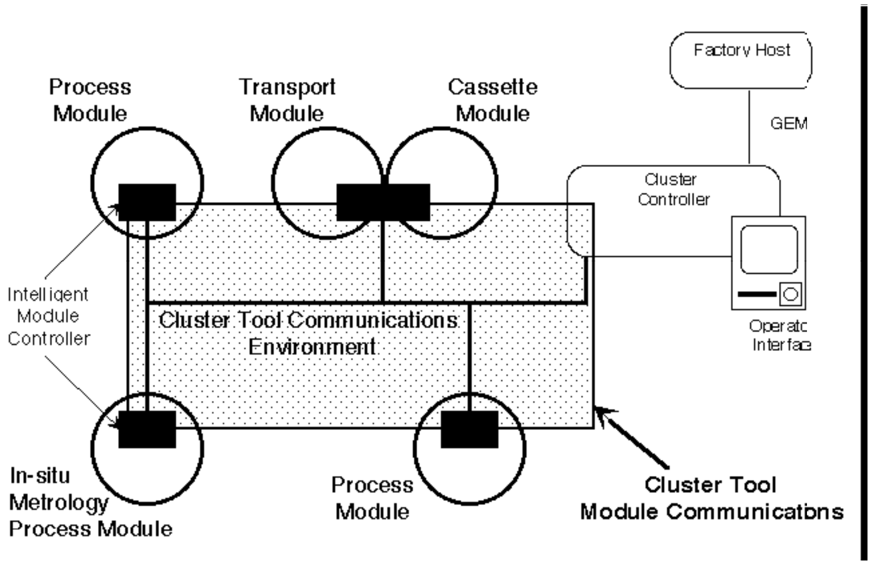
Factory Host
Process Module
Transport Module
Cassette Module
GEM
Cluster Controller
Intelligent Module Controller
Cluster Tool Communications Environment
Operator Interface
In-situ Metrology Process Module
Process Module
Cluster Tool Module Communications

Figure 1
Example Cluster Control Architecture

The communication services include primary control services for material processing in process modules (processing management) and material movement within the cluster tool and material input/output with the factory (material movement management). The cluster controller, which is responsible for supervisory control within the cluster tool, is the service-user of these primary control services. The form of the cluster controller is not dictated. It may be a single platform, distributed, or incorporated with one or more modules.

Support services are also specified to enable resolution of exception conditions, recipe handling, event reporting and data access. The decision authority, which is responsible for making error recovery decisions, is the service-user for exception management. Appropriate agents are the designated service-users for the recipe management and event reporting support services. All service-users access data using the Object Services.

The communications services are defined independently of protocol in order to allow future standardization of alternative protocols without invalidating the services. Figure 2 shows the full communications stack needed for compliance, where the services are mapped to the communications environment, which is defined in supplementary standards.

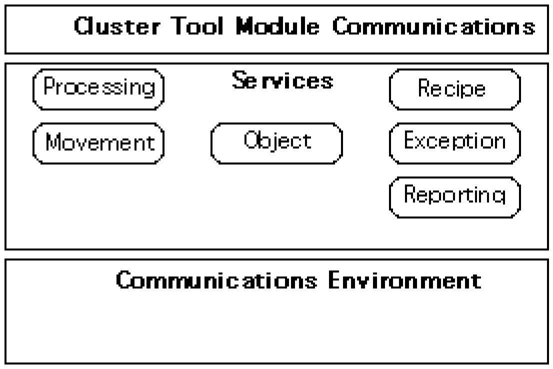
Cluster Tool Module Communications
Services
Processing
Recipe
Movement
Object
Exception
Reporting
Communications Environment

Figure 2
Cluster Tool Module Communications Standards

This standard describes the cluster tool model on which the communications are based. It then defines the application of each of the services required within a cluster tool. This includes limitations imposed by the cluster architecture as well as clear a statement of the fundamental requirements in order to minimize the implementation effort. Connection establishment and maintenance are defined together with the requirements of the communications environment. Communications environment is described in a subsection in order to allow additional options to be standardized in the future. The application notes include factory integration issues and an example scenario of standard communications for the lifecycle of a single material in a cluster tool.

6.1 *Compliance* — Cluster tool module communications compliance between entities in a cluster tool requires that all applicable exposed interfaces conform to the standards specified in this document.

The standardized interfaces are:

- Process module to Service-user (cluster controller)
- Transport module to Service-user (cluster controller)
- Cassette module to Service-user (cluster controller)
- Transport module to Cassette module
- Transport module to Process module

The exposed interfaces are those which result from a particular partition of the control entities in a cluster tool which are from independent suppliers. Three examples follow:

1. All modules independent, single cluster controller platform: Interfaces are required between the cluster controller and each module, and between the transport module and each process and each cassette module.
2. Material handler, process modules independent, single cluster controller platform: Interfaces are required between the cluster controller and each process module, between the cluster controller and each logical module in the material handler (transport and cassette modules), and between the material handler and each process module.
3. One independent process module integrating to a cluster tool control system: Interfaces are required between the cluster tool control system and the process module supporting the process module to cluster controller communications, and the process module to transport module communications.

For fundamental compliance, each exposed interface shall support the fundamental requirements of each of the service groups required by the entities communicating. The required service groups for each interface are detailed in Section 8.6, and the required services for each service group are detailed in Sections 8.1 to 8.6.

7 Concepts

The Cluster Tool architecture and requirements are modeled in order to extract only the entities necessary for robust automated control and monitoring communications within a cluster tool.

Figure 3 illustrates the major module domains and the primary relationships between them. Module resources and material domains are represented differently to others because they embody the physical entities acting and being acted upon respectively.

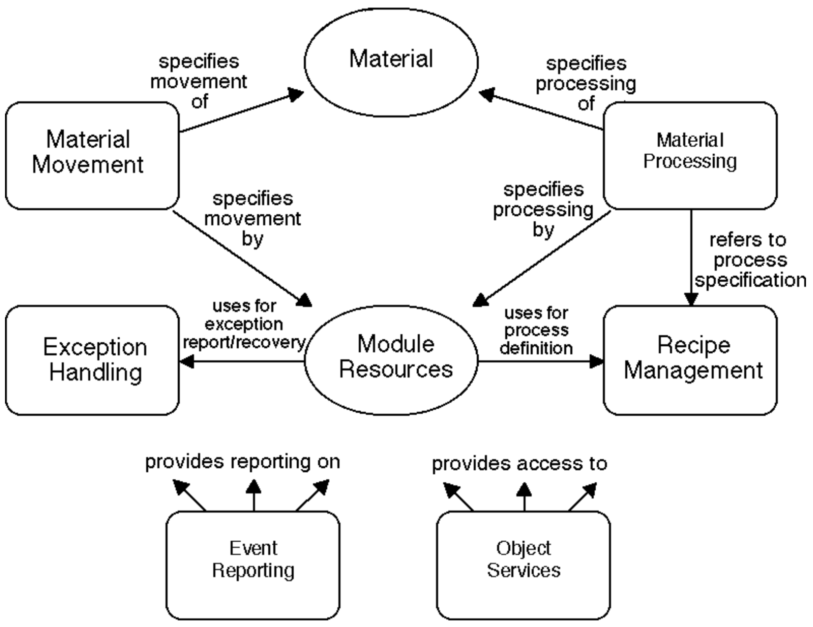

Figure 3
Cluster Tool Module Illustration

The domains illustrated are as follows:

Material — The material, or carrier of material, which is being received and processed in the cluster tool.

Material Processing — Specifies the required application of processing to material as defined by the assigned recipe, adding value to the material.

Material Movement — Includes exchange of material with the factory and material movement within the cluster, to get the material to the correct processing resource at the right time.

Module Resources — Perform and coordinate the processing and material movement activities.

Recipe Management — Provides the services to transfer recipes to be loaded as required for material processing, and manages recipe change control.

Exception Handling — Provides for interactive handling of exception conditions, including reporting and error recovery.

Event Reporting — Enables flexible event-based reporting of variable data as required by service-users.

Object Services — Provide services to obtain and modify standard object attribute data.

Each partition is presented in a subsection below with detailed information models.

The concepts are presented in this section using two types of formal models: OMT object information models describing entities and relationships, and Harel state models describing the behavioral characteristics of the cluster tool. The graphical representations of these models are described in Conventions, Section 5.

The events and actions which result from objects dynamically interacting form the basis for the communication requirements embodied in the standards. It is this mapping that specifies the communication services messages to be used.

The major requirement of the cluster tool control system is that it provide robust supervisory control with minimal overhead. Within a cluster tool there is a significant amount of parallel processing. In addition to concurrent processing in multiple independent process modules, each material processing requires the use of recipe management (to get the correct recipe with the correct material) and the use of material movement (to get the correct material to the correct processing resource at the right time).

The cluster controller maintains the overall mission of the system. Sub-missions are defined for, and responsibility given to, particular available module resources for execution. Once this responsibility is given, that module resource is fully in control, including direct communication with resources of other modules as required to attain the objective.

7.1 *Object Information Models* — The cluster tool objects relevant to distributed control in a number of models, each emphasizing a particular aspect of the system as partitioned (see Figure 4).

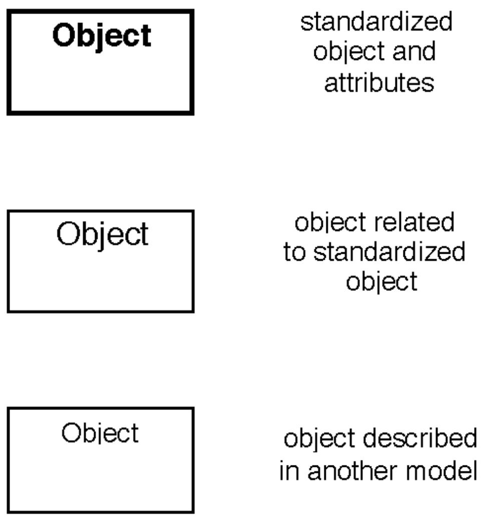
Object
standardized object and attributes
Object
object related to standardized object
Object
object described in another model

Figure 4
Object Designation

In addition to OMT conventions (see Section 5.2), each object in the information models has one of three designations: service objects with standardized attributes (thick border); other objects described in the sub-section (medium border); and objects described in other subsections but included to show significant relationships (thin border).

The models are process- and configuration-independent and do not dictate the internal architecture of the controlled modules beyond the physical requirements of cluster tools.

The first two models show major entities of the processing equipment and material being processed. Subsequent models are oriented towards one of the major functional domains.

7.1.1 *Cluster Tool Module Model* — The fundamental component of a cluster tool is the module. Figure 5 shows that there are three module types within a cluster tool:

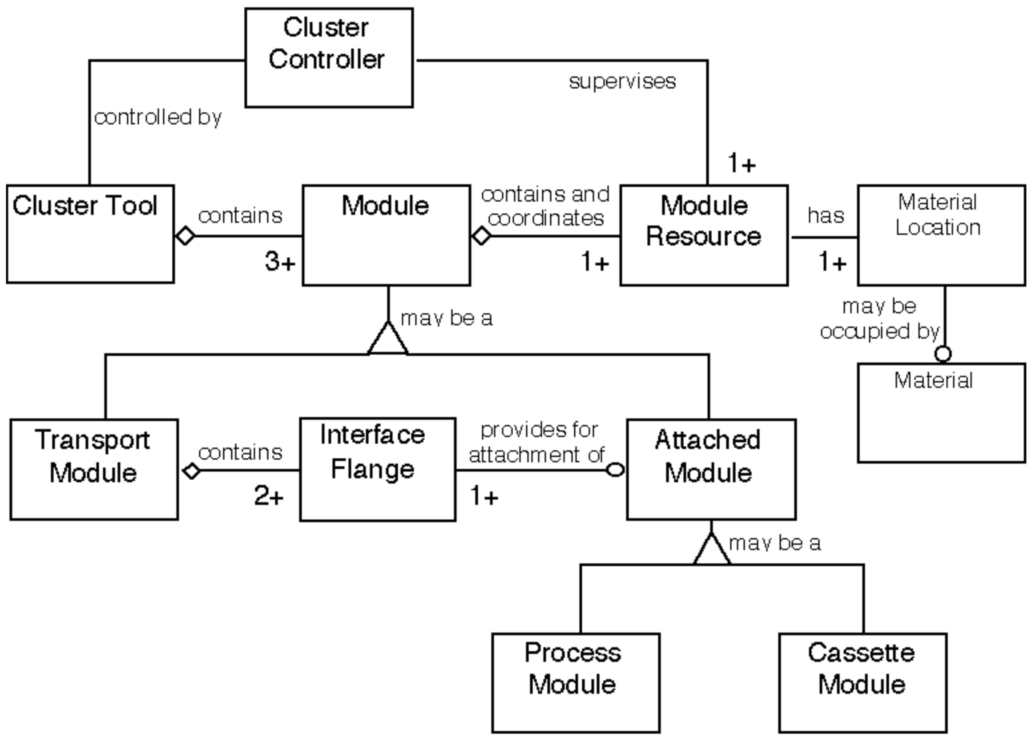

Figure 5
Cluster Tool Module Information Model

- process module (PM)— provides manufacturing value to material
- transport module (TM) — transfers material within cluster
- cassette module (CM) — exchanges material with factory

A cluster tool must contain one or more of each of these module types, as each performs a critical function in the cluster tool mission. Any module can contain processing resources.

Supervisory control within the cluster tool is the responsibility of the cluster controller. The cluster controller achieves this by supervising the various module resources using the standard communications services. It is the primary service-user in the cluster tool. The form of the cluster controller is not specified in this standard. It may be a single platform, distributed, or incorporated with one or more modules.

A module contains and coordinates a number of module resources dedicated to fulfilling a particular control function. Each module resource contains at least one material location, which may or may not contain material. Note that the location for material processing is defined separately from the location where the material is received or sent. This may be virtual if they are physically the same position, but they cannot be assumed to be so as this would exclude certain architectures.

The purpose of each type of module is described in detail below.

Cassette Module — The component of a cluster used to interface the cluster to the rest of the factory. It provides material input and output for the cluster. It makes only one material at a time available to the cluster transport (the intratool environment), regardless of how the material enters the cassette module from outside the cluster (the intertool environment). It is a specialization of attached module (AM).

Process Module — A process module is any component within a cluster which provides one or more steps of material processing for the cluster system. Material enters the process module through an interface flange, undergoes some transformation, and exits through the same or another interface flange. It is also a specialization of attached module (AM).

Transport Module — The transport module is responsible for transfer of material from one cluster-attached module to another. It consists of a robotic handler capable of exchanging material with each of the attached modules at the interface flanges. It can be a variety of physical topologies (radial or linear, for example). In SEMI standard cluster tools, the transport module has an interface isolation valve at each point of attachment to the attached modules. The transport module is responsible for managing constraints on the simultaneous opening of isolation valves as it relates to contamination of the transport module's environment by that of an attached module, as well as cross contamination between attached modules.

The attributes of modules and related objects in Figure 5 are not standardized as these objects are equipment and process-dependent.

7.1.2 *Material Model* — Material is received from the factory, in a carrier or singly, at an intertool port resource. They are transported within the cluster and processed by processing resources. Once processed, they are returned to the factory at the same, or another, intertool port resource.

The processing which is to be performed on material is determined from the process specification for that material.

The material model in Figure 6 establishes the relationships between the material and material locations. Two material types are of interest in cluster tool module communications, single material and carriers. Single material may be grouped in carriers. This is modeled by defining the material slot object, which is a type of material location. A carrier is made up of an ordered set of material slots. A single material occupies a material slot when in the carrier. In addition, a single material may be assigned a particular material slot which it does not yet occupy, indicating that it is to be received into that material slot when it arrives.

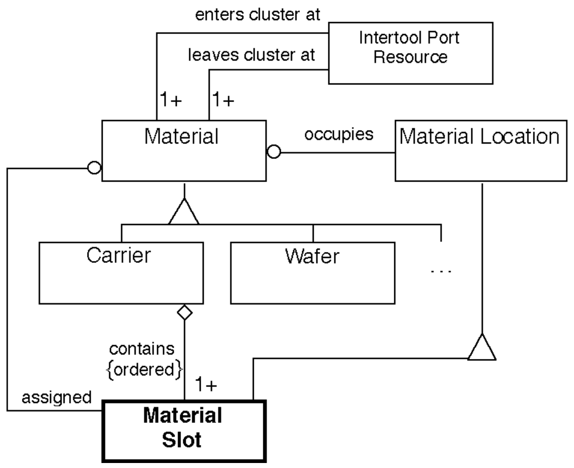

Figure 6
Material Information Model

Carriers may be cassettes used to transport single material in the factory and to input single material to the cluster tool, or they may be material boats in batch process modules.

The attributes of the material slot object are standardized to enable mapping of single material in carriers. Other objects in the diagram are not standardized. However, the identifier of the material, MaterialID, is an attribute of the material slot object.

Other materials (consumable gases and fluids, etc.) are present in the cluster tool, but outside the scope of these standards.

7.1.3 *Material Processing Model* — The material processing model in Figure 7 shows the use of the process job to direct the processing resource to apply the appropriate process, as specified by the recipe, to one or more materials in a process module. A processing resource in a process module has processing capabilities which are specified by corresponding recipes.

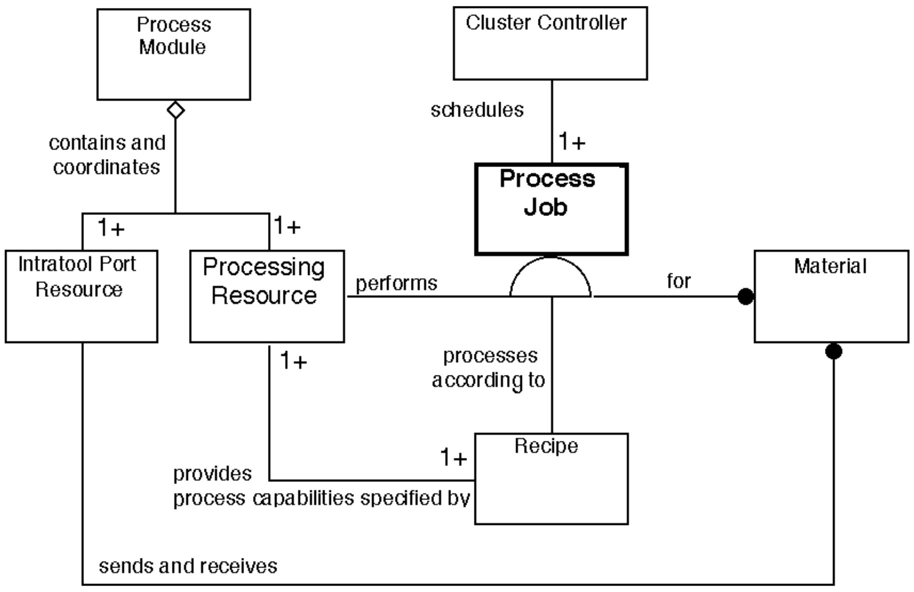

Figure 7
Material Processing Information Model

A process job can control multiple material only if identical processing is applied to all. Processing should begin and end simultaneously, synchronized with the arrival of the whole group of material.

The process job lifecycle extends beyond the active processing of the material. It exists from just before material arrival at the intratool port resource, through setup and processing, and until just after the material departure at the same, or another, intratool port resource. This allows for material-related pre-conditioning and post-conditioning of the processing resource before the material is received and after it is sent. As material input and output is controlled by the intratool port resource, it is the responsibility of the process module to ensure coordination of the intratool port and processing resources.

This is the model used to establish the Processing Management definition. There is no standardization beyond creation and control (starting, canceling, stopping, aborting, and pausing) of the high-level process job since the low-level control of process modules is application-dependent.

There is a component of scheduling embodied in this model. The material, recipes, and processing resources are scheduled by the cluster controller according to the process specification to a particular process module through the use of the process job. In order to perform the required processing, material is transferred to the processing resource, as directed by the cluster controller.

7.1.4 *Material Movement Models* — Material movement in a cluster tool includes all services required within the cluster tool to move material from the factory to a sequence of process modules in order to be processed and then back to the factory. Material is moved individually within a cluster tool and may be received from the factory in cassettes or individually.

The material movement model is divided into three parts as follows:

- intratool material movement defines the transfer of material between attached modules, with the transfer agent being a transport module.
- intertool material movement defines the exchange of carriers and single material with the factory at the cluster tool cassette module.
- carrier mapping defines presence and identifiers of single material contained in carriers.

These three models are used to establish the Material Movement Management definition within a cluster tool.

7.1.4.1 *Intratool Material Transfer* — The intratool material movement model is shown in Figure 8. An intratool material transfer is the movement of material from one attached module to another attached module through the linking transport module.

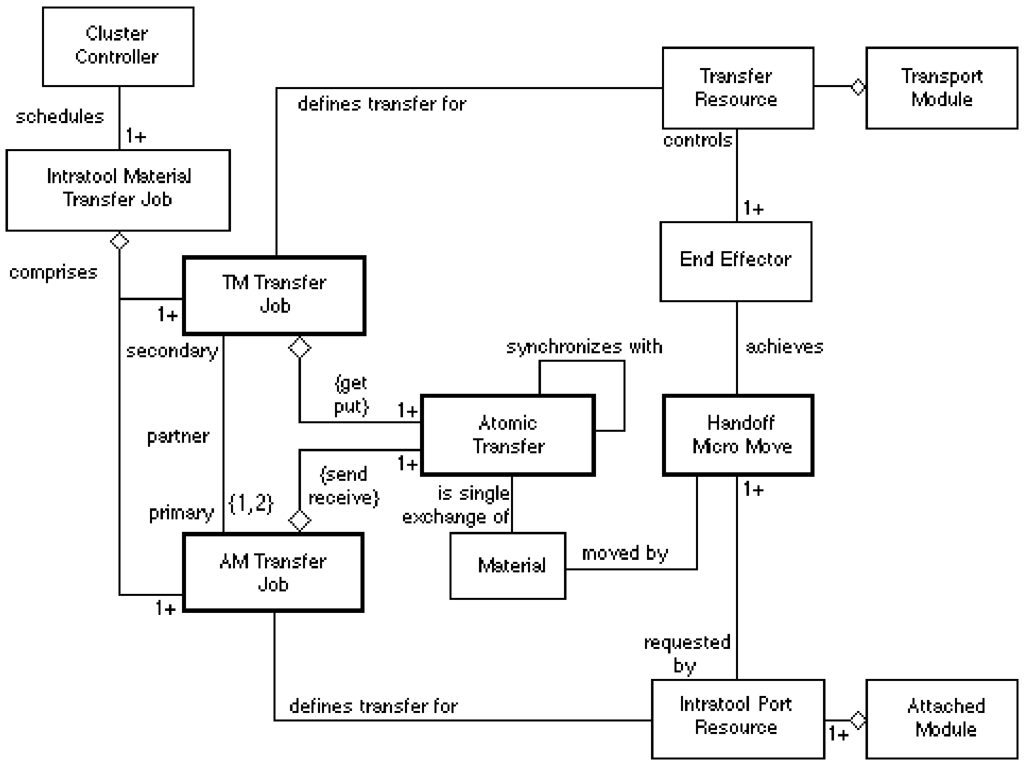

Figure 8
Intratool Material Movement Information Model

The cluster controller schedules an intratool material transfer job when intratool material transfer is required. This is achieved by TM and AM transfer jobs assigned to the appropriate module resource in each of the participating modules:

- to the transfer resource in the transport module to get the material from the source attached module and put it into the destination attached module.
- to the intratool port resource in the source attached module to send the material to the transport module.

- to the intratool port resource in the destination attached module to receive the material from the transport module.

Transfer jobs are made up of atomic transfers. An atomic transfer in one transfer partner synchronizes with the corresponding atomic transfer in the other partner to achieve the material transfer through handoff micro moves. In a cluster tool, the attached module is the primary partner, requesting the handoff micro moves, and the transport module is the secondary partner achieving those micro moves by controlling the end effector on which the material is transported.

The transport module is responsible for managing constraints on the simultaneous opening of isolation valves as it relates to contamination of the transport module's environment by that of an attached module, as well as cross contamination between modules. The attached module ensures that its environment will not contaminate the transport module before synchronizing for handoff. The rules for re-establishing isolation at verification are determined by the transport module.

7.1.4.2 *Intertool Material Transfer* — The cassette module performs material exchange with the factory. The material generally consists of cassettes with single material or empty cassettes, but may be individual single material. The intertool material movement model, shown in Figure 9, is similar to the intratool material movement model. An intertool material transfer is a transfer in the factory which includes the cluster tool as one of the transfer partners.

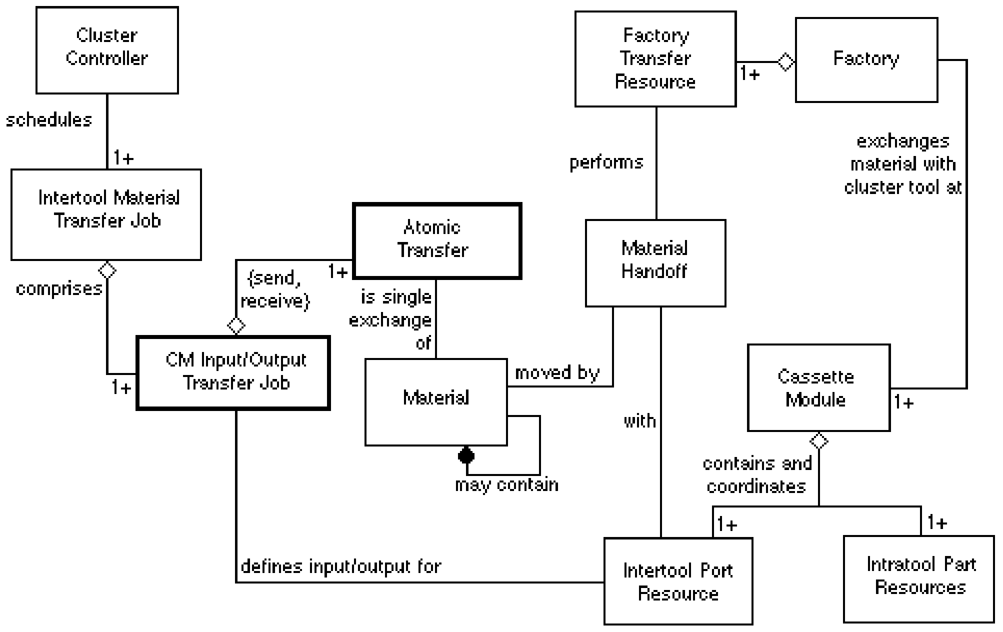

Figure 9
Intertool Material Movement Information Model

In the cluster tool, the cluster controller schedules an intertool material transfer job when material exchange with the factory is required. This job may be an element of a factory material transfer job in which the cluster tool is a partner.

The intertool material transfer job is achieved by a CM input/output transfer job assigned to the intertool port resource in the participating cassette module. It defines the material exchange with the factory for the cassette module. The CM input/output transfer job comprises an atomic transfer, which defines the roles in the material handoff.

In the cassette module, material transfer with the transport module is controlled by the intratool port resource. It is the responsibility of the cassette module to coordinate its intertool and intratool port resources.

The factory transfer resource may be an operator, SMIF, AGV, or other mechanism. Its definition, associated transfer jobs, and the handoff, are beyond the scope of this standard. Factory material movement may be achieved automatically using applicable SEMI standards.

7.1.4.3 *Carrier Mapping* — The final element of material movement in a cluster tool is the mapping of single material in a carrier. Once a cassette is received by the cluster tool, it is necessary to communicate the presence and identifiers of the material it contains. Batch process modules also require these carrier mapping services in order to define loading of the process carrier.

In carrier mapping, the carrier may be a cassette used to hold single material loaded at the cassette module from the factory, or a process carrier used to hold multiple material for batch processing. For the purposes of carrier mapping, these are equivalent, and the relationships are shown in the material model, Figure 6.

A carrier has an ordered set of material slots, a specialization of material location, each of which can hold a single material. The material slot of a carrier is only of interest when the carrier is in the cluster tool, so it is viewed as a location of the module which contains the carrier. In addition to occupancy, a single material may be assigned to a material slot, indicating, for example, the particular slot to which a single material is to be moved when available. Information on material presence and assignment of the material identifier may be detected in the module or determined by the cluster controller or a supervisory controller.

7.1.5 *Exception Model* — In an automated control system where the operator interface is remote from the module controller, it is necessary to have interactive exception handling for error recovery. In addition to exception reporting, a module requires input from a decision authority to resolve recoverable abnormal situations. These include exception conditions which extend beyond the module domain and those for which the module has insufficient information, such as in hardware failure.

The decision authority in a cluster tool is the cluster controller. It may interact with an operator to determine the appropriate recovery action to perform.

Figure 10 shows the exception model. Module resources detect appropriate exception conditions, which may be set or cleared. An abnormal situation is indicated by the corresponding exception condition being in the set state. A significant change in exception condition information generates an exception report to notify the decision authority. Both detection of the abnormal situation and its resolution generate exception reports. Reporting of each exception condition may be enabled and disabled independently in order to mask nuisance exceptions.

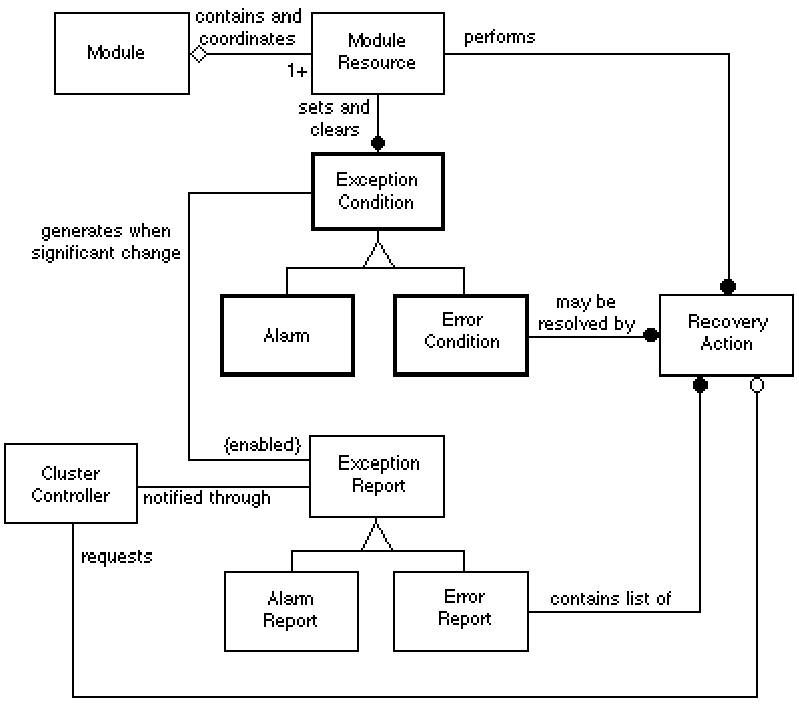

Figure 10
Exception Information Model

Error exception conditions may have associated recovery actions which can be performed by the module resource to attempt to recover from the abnormal situation. Alarm exception conditions cannot, by definition, be resolved using recovery actions.

A list of possible recovery actions is included in a posted error report. The decision authority can request one of the offered recovery actions to resolve the error condition. The selected recovery action is performed by the appropriate module resource and may result in clearing the error condition.

This model is used to establish the Exception Management definition.

7.1.6 *Recipe Management Model* — Recipe management in a module varies with the requirements of the application. Simple modules do not provide local recipe editing and don't require management of multiple logical domains (namespaces) and local recipe version control.

Figure 11 shows the full recipe management model for a sophisticated module.

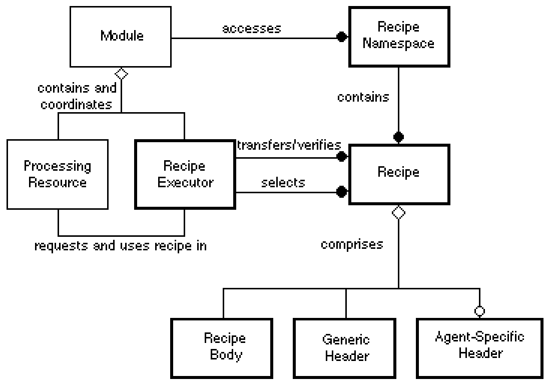

Figure 11
Recipe Management Information Model

Process recipes specify the actual parameters of the processing to be achieved. The module processing resource identifies, through the process job, the required recipe(s) to be loaded by the recipe executor in order to process a particular material.

The recipe executor accesses and selects recipes for execution. It uploads and downloads recipes from the cluster controller and is capable of verifying that a recipe is syntactically correct. The recipe executor also provides for a service-user to select a recipe to be loaded into the recipe execution area. The recipe is made up of a recipe body and may have headers if the module supports recipe namespace management.

The module accesses recipe namespaces where requested by service-users. A module which does not provide local editing and linking of recipes does not need recipe namespace management. Recipe namespace management requires management of recipe generic and agent (module) specific headers. For example, a process module could manage a recipe namespace containing its process and service recipes.

This model is used to establish the Recipe Management services requirements for cluster tool modules.

7.1.7 *Event Reporting Model*

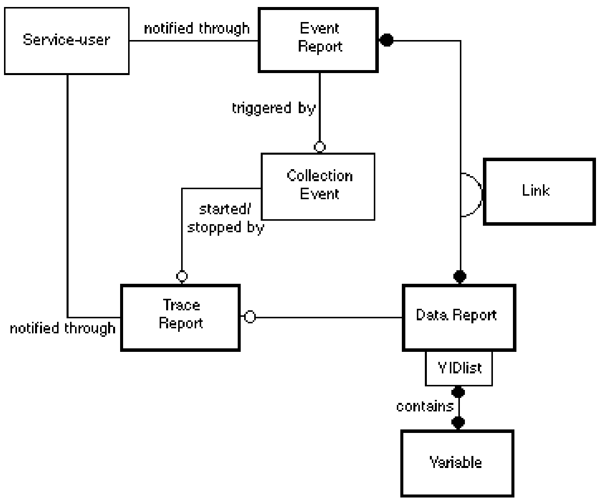

Figure 12
Event Reporting Information Model

Event reporting provides a dynamic and flexible means by which a service-user can receive notification of events and data relating to module resources.

The services-user can define and enable two types of reporting, event reporting and trace reporting. Both may convey data, the values of identified variables at the data collection time. The variable identifiers required to be reported are specified in the data report definition.

In event reporting, data reports are linked to event report definitions associated with each collection event type. On occurrence of the collection event, an event report message is generated according to the event report definition and the linked data report definitions. Reporting may be enabled and disabled for each collection event type. Data reports and links may be predefined or specified dynamically by the service user.

Trace reporting is time-based data reporting. Trace reporting is specified by the service-user dynamically as required. Collection events may initiate and terminate the tracing in order to link reporting to significant conditions, such as processing. The trace data is collected at a frequency specified by the service-user. Trace reports are generated according to the trace report definition and the associated data report definition.

Through associating the event report and trace report definition with the service-user, the model provides for the association of multiple service-users to a single service-provider. This allows for event reporting between modules where needed as well as the ability for direct connection of a data acquisition entity independent of the cluster controller.

7.2 *Cluster Tool Module Message Flow* — The appropriate objects in the information models in the previous section are grouped in the cluster tool module types. The communications with the essential objects are shown in the message flow diagrams in this section.

The diagrams show message flow between entities within an agent (in this case a module) and those outside the agent. Entities are rectangular boxes, and the agent is a rounded box. As emphasis is on the module and its services, only one agent is shown on each diagram. Outside entities in the cluster tool module communications domain appear within an agent on another diagram. The messaging is shown by arrows in each direction, with a brief label indicating the services requested or provided.

The communications are shown for the primary control of the process module (Figure 13), cassette module (Figure 14), and transport module (Figure 15). Note that the transport module transfer resource communicates with multiple process and cassette module intratool port resources.

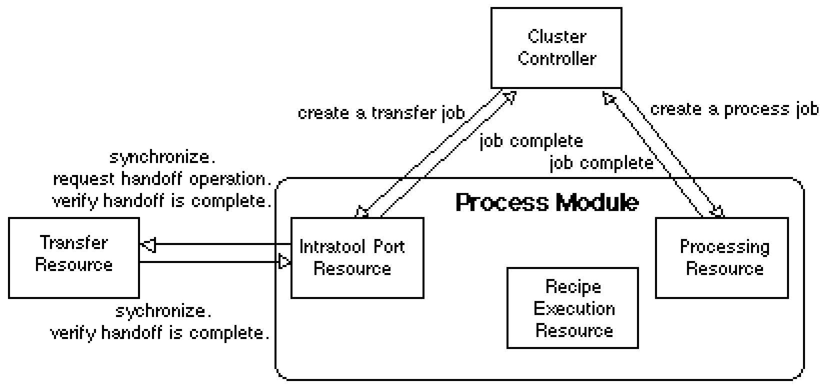

Figure 13
Process Module Primary Control Message Flow

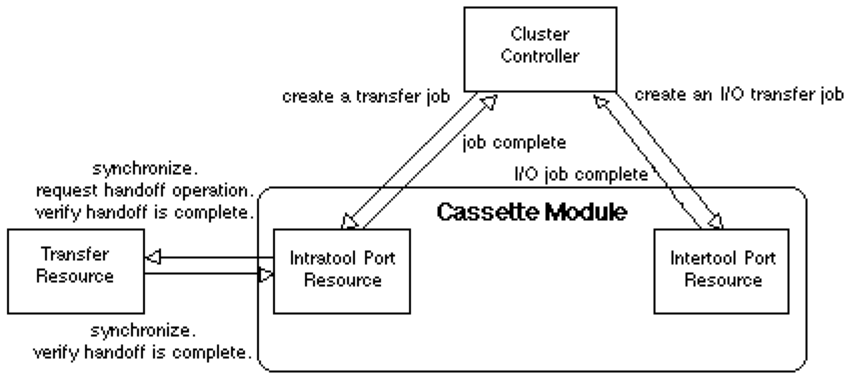

Figure 14
Cassette Module Primary Control Message Flow

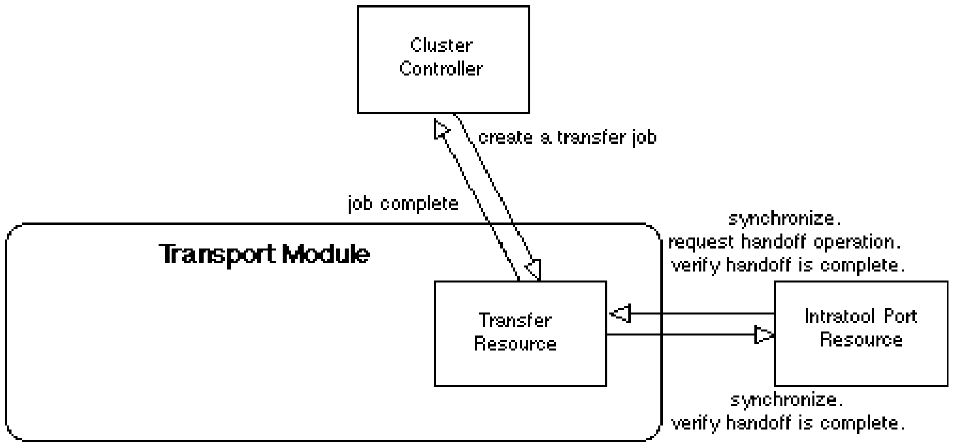

Figure 15
Transport Module Primary Control Message Flow

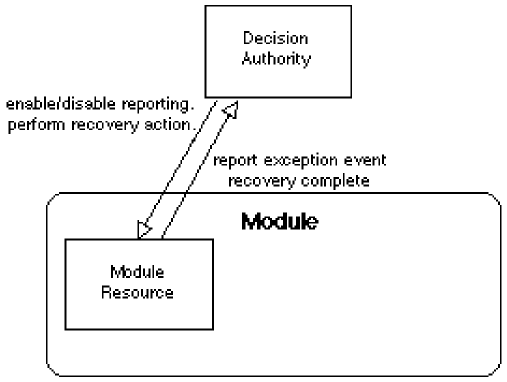

Figure 16
Module Exception Message Flow

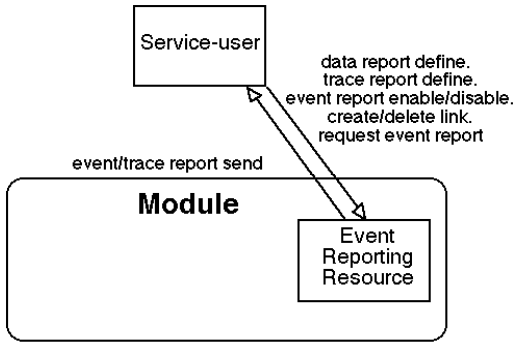

Figure 18
Module Event Reporting Message Flow

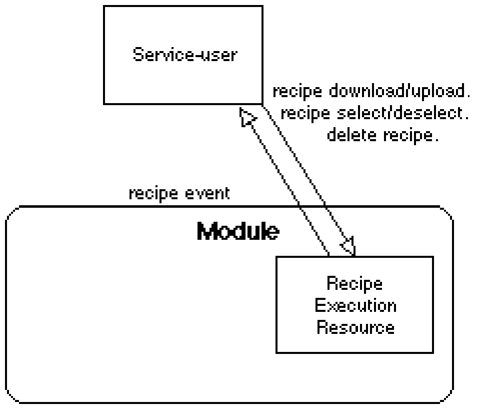

Figure 17
Module Recipe Message Flow

The communications for support functionality are shown for any module, as the services are common to all module types. The support functionality modeled includes exception handling (Figure 16), recipe management (Figure 17), and event reporting (Figure 18). A typical decision authority or service-user is the cluster controller.

7.3 *Cluster Tool Behavioral Model* — The process material is the fundamentally important object in a cluster tool and its lifecycle is the key component to establishing cohesiveness in the communication and controls standards.

The finite state model in Figure 19 presents the high-level behavior of a single material within the domain of the cluster tool. Bold transitions trace the normal lifecycle of the material and are described in the transition table, Table 1. Other transitions occur in exception situations and are not detailed here. The transition table actions indicate the primary control operations which are requested of the modules by the cluster controller.

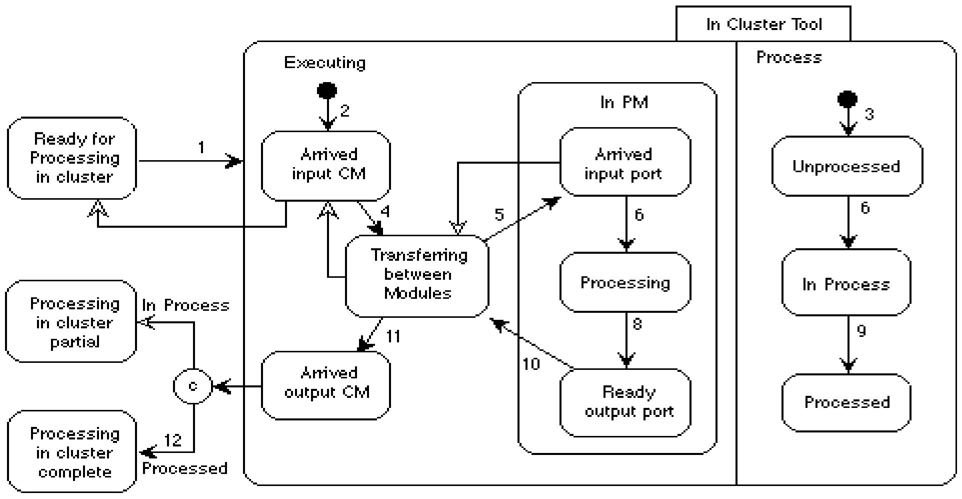

Figure 19
Cluster Tool Material State Model

When ready to be processed in the cluster tool, the material is presented and loaded at the input cassette module (intertool material transfer). It is usually contained with other single material in a cassette. The material process sub-state in the scope of this visit to the cluster tool is initially unprocessed. The material is then transferred to the first process module (intratool material transfer), ready for process. At this point, the material may be returned to the input cassette module and back to the factory, as the material has not been physically altered.

The material then begins processing in the process module according to the recipe for that process step (process job), and the process sub-state changes to in-process.

On successful completion of the process step, the material is then ready to be transferred to the next module (intratool material transfer). This may be another process module for the next process step or to the output cassette module if processing in the cluster tool is complete, in which case the process sub-state changes to processed. It is also possible that the next process step be performed in the same process module.

Table 1 Cluster Material State Transition Table

#	*Current State*	*Trigger*	*New State*	*Action(s)*
1	Ready for Processing in cluster	Material being received at cluster CM.	In cluster tool	CM Input/Output Transfer Job. CM Carrier mapping.
2	Not in Cluster tool	Material being received at cluster CM.	Arrived input CM	
3	Not in Cluster tool	Material being received at cluster CM.	Unprocessed	
4	Arrived input CM	Initiate material transfer to PM.	Transferring between modules	Initiate PM Process Job. TM Transfer Job. Source AM Transfer Job (CM send). Dest. AM Transfer Job (PM receive).
5	Transferring between Modules	Material arrived in PM.	In PM/Arrived input port	
6	In PM/Arrived input port	Processing initiated on material.	In PM/Processing	PM Process the material.
7	Unprocessed	Processing initiated on material.	In Process	
8	In PM/Processing	PM processing complete.	In PM/Ready output port	
9	In Process	All required process steps in cluster completed.	Processed	
10	In PM/Ready output port	Initiate intratool material transfer.	Transferring between modules	Initiate next PM Process Job if material processing still required. TM Transfer Job. Source AM Transfer Job (send). Dest. AM Transfer Job (receive).

#	*Current State*	*Trigger*	*New State*	*Action(s)*
11	Transferring Between Modules	Arrived output CM.	Arrived output CM	CM Input/Output Transfer Job.
12	Arrived output CM	Transfer back to factory complete.	Processing in cluster complete	

The material is finally unloaded from the output cassette module to the factory (intertool material transfer). If all process steps in the cluster tool were completed and were successful, the processing in the cluster tool is complete; otherwise, processing in the cluster tool was only partially accomplished.

Detailed behavioral models are documented in the individual Cluster-Specific Services and in the standards which they reference. The Typical Operating Scenario in the application notes illustrates the behavioral characteristics of cluster tool control by showing the detailed messaging among modules and supervisory controller through the lifecycle of material as it enters the cluster, is processed, and then leaves the cluster.

7.4 *Cluster Tool Control Topology* — The natural decomposition of a cluster tool is into its modules. The module controller is the entity which performs internal control of a module and provides standard services. Module controllers provide certain application-specific functions (providers of services and user of services) and consist of hardware and software components that interact with the communications and control environment of the cluster tool.

Traditional definitions of controllers have implied certain PHYSICAL, hardware control topologies (such as distributed or centralized processors). These cluster tool standards specify the communications in terms of services-providers, enabling the module controllers to be based on LOGICAL control topologies.

The list of possible physical controllers which could be used to implement the logical services of the standards are listed below. It is emphasized, however, that the standard is not based on any constraint that each of these physical controllers be present in a cluster control system, but instead requires that the logical services (i.e., software functional components) be present somewhere in the cluster control system.

Possible Controllers:

- Cassette Module Controller
- Process Module Controller
- Transport Module Controller
- Cluster Controller

The logical control topology allows any combination of module controllers on a single control unit. Standard communication is not required among them. A common example is the grouping of transport module and cassette module control into a single material handler control unit as shown in Figure 1. For interface compliance, all communication between a logical module controller and remote controller or service-user must be compliant with the standard services.

8 Cluster Tool Module Serv ices

The Cluster Tool control system elements, as represented in the object information, object communications, and behavioral models, require standard services in order to interface with one another.

The objects which are fully standardized (attributes, behavior, and operations) are only those essential to this communication, and the services are the interactions with these standard objects.

The services are grouped into the following functional areas:

- Object Services
- Processing Management
- Material Movement Management
- Exception Management
- Recipe Management
- Event Reporting
- Clock

The services of each of these groups are described in detail in this section. The services are defined in terms of services provided by independent SEMI standards, specifying their application and limitations with respect to the fundamental requirements of cluster tool module communications.

8.1 *Object Services* — All standar dized service object attribute access across a cluster tool module communications interface is achieved using object services. Access may be to read attribute values of objects, to get a list of objects corresponding to certain criteria, and to set the values of writable attributes of objects. The message services provided to accomplish this are Get and Set.

The specific object types, relations, attributes, and access rules are defined in each specific services group.

The object services are defined in SEMI E39 (Object Services Standard: Concepts, Behavior, and Services). Cluster tool module communications require only the fundamental requirements of that standard except where stated in a specific services group below.

8.2 *Processing Management Services* — Processing management across a cluster tool module communications interface is achieved using the processing management services. These services provide the communications and behavior required of the processing resource by the material processing models through defining the process job object and its operations.

The processing management services follow SEMI E40 (Standard for Processing Management). Cluster tool module communications require full compliance with the information, behavior, and messaging services of the fundamental requirements of that document.

The process job creation is requested by the cluster controller using the PRJobCreate service. If the processing resource accepts the job, it performs the required processing to completion once the material arrives. The processing resource shall support the PRJobCommand Abort service, which ceases any processing and terminates the job. The processing resource reports process job milestones achieved to the cluster controller using the PRJobAlert service.

Process job object attributes required to be accessible through object services follow:

Attribute Name	*Description: Requirements*
ObjType	The object type: process job.
PRJobID	Identifier for the process job.
PRMtlType	MaterialType.
PRMtlNameList	Material identifier: single material only.
PRRecipeMethod	Recipe specification type: recipeID only.
RecID	Identifier of the recipe applied.
PRJobStateList	All concurrent substates of the process job.

Processing management messaging services required:

Message Name	*Description: Requirements*
PRJobCreate	Create process job: single material, recipe identifier only, auto-start.
PRJobCommand	Command on a process job: abort.
PRJobAlert	Notification: setup, processing, processing complete, complete.

Processing resource capabilities required, but not already specified above:

- Detect and report success or failure of a process job.
- Reject incomplete, invalid, and unsupported requests.

Processing resource capabilities permitted, but not to be required, by a service-user:

- Pre-conditioning and post-conditioning.
- Stop, Pause, and Resume of a process job.
- Manual process start.
- Process job queuing and queued job Cancel.
- Process tuning.
- Processing of material groups.
- Multiple concurrent process jobs.
- Multiple consecutive process jobs.
- Process job with no material.
- Notification of waiting for material and of process job state changes.

8.3 *Material Movement Services* — Each material movement in a cluster tool involves interaction among the cluster controller and all of the modules participating in the transfer. The communications interfaces and behavior resulting from the material movement models are defined for each association by the material movement services.

The material movement services are partitioned into intratool material movement services, intertool material movement services, and carrier mapping, as in the models.

The material movement services follow the Material Movement Management Standard [SEMI E32] and Object Services Standard: Concepts, Behavior, and Services [SEMI E39]. Cluster tool module communications require full compliance with the information, behavior, and messaging services of the fundamental requirements of those standards.

8.3.1 *Intratool Material Movement Services* — From the intratool material movement models, an intratool material transfer — the transfer of material from one attached module to another through a transport module — requires the following interactions:

1. The cluster controller decides that material is to be moved as the result of either arrival of material or the completion of some processing step on the material. It determines which material is to be

moved and the source and destination of that movement (cluster-application-specific). It requests all the modules involved to perform the appropriate TM or AM transfer job.

2. The source- and destination-attached modules inform the transport module that they are prepared to do their specific activities during the material transfer. The transport module informs the source- and destination-attached modules, each at the appropriate point in time, that it is synchronized for material handoff.

3. The detailed transactions required to handoff the material during the transfer are accomplished, first at the source and then at the destination.

These interactions may require a number of communications interfaces for intratool material transfer from the following:

- Cluster controller to transport module
  - transfer resource: TM transfer job source AM to destination AM
- Cluster controller to source-attached module
  - intratool port resource: AM transfer job send
- Cluster controller to destination-attached module
  - intratool port resource: AM transfer job receive
- Transport module to source-attached module
  - intratool port to transfer resource: handoff get
- Transport module to destination-attached module
  - intratool port to transfer resource: handoff put

The material movement services define the communications and behavior required of the attached module intratool port resource and the transport module transfer resource. They define the TM transfer job, TM atomic transfer, AM transfer job, and AM atomic transfer objects and their operations, as well as the handoff operations between the transfer resource and the intratool port resource.

Each interface is described below.

Cluster Controller to Transport Module Interface — The TM transfer job creation is requested by the cluster controller using the TRJobCreate service. If the transfer resource accepts the job, it performs the transfer of the material to completion, coordinating with the attached modules. The TM transfer job comprises two atomic transfers for the material, one to get (receive) it from the source-attached module and another to put (send) it to the destination-attached module.

The transfer resource shall support the TRJobCommand Abort service, which ceases any transfer activity and terminates the job. The transfer resource reports transfer job milestones achieved to the cluster controller using the TRJobAlert service.

The transfer resource coordinates material handoff with each attached module through the transport module/attached module interface, described below.

TM transfer job object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: transfer job.
TRJobID	Identifier for the transfer job.
(Ordered list of) TRAtomicID	Ordered list of identifiers for the atomic transfers in the transfer job: two atomic transfers for the material.
(List of)TRJobState	All concurrent substates of the transfer job.

TM atomic transfer object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: atomic transfer.
TRAtomicID	Identifier for the atomic transfer.
TRLink	Identifier to coordinate handoff.
TRObjType	Material Type.
TRObjName	Material identifier.
TRRole	Role in transfer control: secondary.
TRPartner	Identifies partner-attached module.
TRDirection	Handoff direction: receive from source AM, send to destination AM.
TRType	Active/passive in transfer: active.
(List of)TRAtomic-State	All concurrent substates of the atomic transfer.

Material movement messaging services required:

Message Name	*Description: Requirements*
TRJobCreate	Create transfer job: single material, source to destination AM, auto-start.
TRJobCommand	Command on transfer job: abort.
TRJobAlert	Notification: started, complete.

Transfer resource capabilities required, but not already specified above:

- Detect and report success or failure of a transfer job.

- Reject incomplete, invalid, and unsupported requests.

Transfer resource capabilities permitted, but not to be required, by a service-user:

- Stop, Pause, and Resume of a transfer job.
- Manual atomic transfer start.
- Transfer job queuing and queued job Cancel.
- Transferring of material groups.
- Multiple concurrent transfer jobs.
- Notification of transfer events.

Cluster Controller to Attached Module Interface — The AM transfer job creation is requested by the cluster controller using the TRJobCreate service. If the intratool port resource accepts the job, it coordinates with and directs the transport module to achieve the material handoff. The AM transfer job comprises an atomic transfer for material which is to receive it from, or send it to, the transport module.

The intratool port resource shall support the TRJobCommand Abort service, which ceases any transfer activity and terminates the job. The intratool port resource reports transfer job milestones achieved to the cluster controller using the TRJobAlert service.

The intratool port resource coordinates material handoff with the transport module through the transport module to attached module interface, described below.

AM transfer job object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: transfer job.
TRJobID	Identifier for the transfer job.
(Ordered list of) TRAtomicID	Ordered list of identifiers for the atomic transfers in the transfer job: one atomic transfer for each material.
(List of)TRJobState	All concurrent substates of the transfer job.

AM atomic transfer object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: atomic transfer.
TRAtomicID	Identifier for the atomic transfer.
TRLink	Identifier to coordinate handoff.
TRPort	AMPort through which transfer occurs.
TRObjType	Material type.
TRObjName	Material identifier.
TRRole	Role in transfer control: primary.
TRPartner	Identifies partner transport module.
TRDirection	Handoff direction: receive or send.
TRType	Active/passive in transfer.
(List of)TRAtomicState	All concurrent substates of the atomic transfer.

Note that TRLocation is not required to be supported as it is the responsibility of the attached module to determine the appropriate location to be used based on the material identifier.

Material movement messaging services required:

Message Name	*Description: Requirements*
TRJobCreate	Create transfer job: single material, send or receive, auto-start.
TRJobCommand	Command on transfer job: abort.
TRJobAlert	Notification: started, complete.

Transfer resource capabilities required, but not already specified above:

- Detect and report success or failure of a transfer job.
- Reject incomplete, invalid, and unsupported requests.

Transfer resource capabilities permitted, but not to be required, by a service-user:

- Stop, Pause, and Resume of a transfer job.
- Manual atomic transfer start.
- Transfer job queuing and queued job Cancel.
- Transferring of material groups.
- Multiple concurrent transfer jobs.
- Notification of transfer events.

Transport Module to Attached Module Interface — The control interface for the handoff of material between the end effector of a transport module and any attached module supports a sequence of three phases:

1. a synchronization phase that establishes that both the attached module intratool port resource and the transport module transfer resource are ready to handoff the material.
2. a physical handoff phase controlled by the intratool port resource.

3. a verification phase to determine that actual handoff took place. This action completes the handoff.

The transport module is responsible for managing constraints on the simultaneous opening of isolation valves as it relates to contamination of the transport module's environment by that of an attached module, as well as cross-contamination between modules. The attached module ensures that its environment will not contaminate the transport module before synchronizing. The rules for re-establishing isolation at verification are determined by the transport module.

The synchronization between the transfer resource and the intratool port resource is achieved by each, indicating readiness to transfer the material, using the HOReady service. Once both partners have received the HOReady, the intratool port resource directs the physical handoff. The intratool port resource commands the transfer resource using the defined HOCommand services. Once the physical handoff is complete, the intratool port resource requests verification using the HOVerify service.

The transfer resource and intratool port resource shall support the HOCancelReady service which cancels a previously sent HOReady while still in the synchronization phase. The transfer resource shall support the HOHalt service, which ceases any transfer activity and terminates the handoff.

Material movement messaging services required:

Message Name	*Description: Requirements*
HOReady	Synchronize with partner.
HOCommand	Command to the transport resource to perform an activity in the transfer envelope. Detailed below.
HOVerify	Verification of handoff completion with transfer resource.
HOCancelReady	Rescinds previous HOReady message.
HOHalt	Command to the partner to stop all action.

HOCommands to be supported by the transfer resource:

Command	*Description: Requirements*
Pick	Commands the transfer resource to take all actions necessary to remove the material from the intratool port.
Place	Commands the transfer resource to take all actions necessary to place the material in the intratool port.
Extend	Commands the transfer resource to reach into the intratool port to a predefined position at which the material can be accepted or released.
Retract	Commands the transfer resource to withdraw from the intratool port.

Where the robotic hardware does not provide for vertical motion, Pick and Place are not required to be supported.

Transfer partner capabilities required, but not already specified above:

- Detect and report success or failure of handoff commands.
- Detect and report success or failure of the handoff, through the verification.
- Reject incomplete, invalid, and unsupported requests.

Transfer resource capabilities permitted, but not to be required, by either partner:

- Multiple concurrent handoffs.
- Additional handoff commands.

8.3.2 *Intertool Material Movement Services* — Factory-to-cluster tool carrier or single material input/output management across a cluster tool module communications interface is achieved using the intertool material movement services. These services provide the communications and behavior required of the intertool port resource by the intertool material movement models through defining the CM input/output transfer job object and its operations.

This standard specifies only the required communication between the cluster controller and the cassette module and does not assume automated material movement in the factory. The CM input/output transfer job is compliant with SEMI E32 (Material Movement Management) and may be extended to support automated material movement in the factory.

The CM input/output transfer job creation is requested by the cluster controller using the TRJobCreate service. If the intertool port resource accepts the job, it coordinates with the operator or some entity to achieve the material handoff. The CM input/output transfer job comprises an atomic transfer for a cassette or single material which is to receive it from, or send it to, the factory.

The intertool port resource shall support the TRJobCommand Abort service, which ceases any transfer activity and terminates the job. The intertool port resource reports transfer job milestones achieved to the cluster controller using the TRJobAlert service.

CM input/output transfer job object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: transfer job.
TRJobID	Identifier for the transfer job.
(Ordered list of) TRAtomicID	Ordered list of identifiers for the atomic transfers in the transfer job: one atomic transfer for each material.
(List of)TRJobState	All concurrent substates of the transfer job.

Atomic transfer object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: atomic transfer.
TRAtomicID	Identifier for the atomic transfer.
TRLink	Identifier to coordinate handoff: none.
TRPort	CMPort through which transfer occurs.
TRObjType	Material Type: cassette or single material.
TRObjName	Material identifier.
TRRole	Role in transfer control: secondary.
TRPartner	Identifies partner if automated: none.
TRDirection	Handoff direction: receive or send.
TRType	Active/passive in transfer: passive.
(List of)TRAtomicState	All concurrent substates of the atomic transfer.

The requirements above indicate the settings if the transfer is to be manual, with the operator. The attributes TRLink, TRRole, TRPartner, and TRType are not significant in this situation. These become significant in automated material transfer.

Material movement messaging services required:

Message Name	*Description: Requirements*
TRJobCreate	Create transfer job: single material send or receive, auto-start.
TRJobCommand	Command on transfer job: abort.
TRJobAlert	Notification: started, complete.

Intertool port resource capabilities required, but not already specified above:

- Detect and report success or failure of a transfer job.
- Reject incomplete, invalid, and unsupported requests.

Intertool port resource capabilities permitted, but not to be required, by a service-user:

- Automated transfer of material within factory.
- Stop, Pause, and Resume of a transfer job.
- Manual atomic transfer start.
- Transfer job queuing and queued job Cancel.
- Transferring of material groups.
- Multiple concurrent transfer jobs.
- Notification of transfer events.

8.3.3 *Carrier Mapping Services* — Mapping of material in carriers across a cluster tool module communications interface is achieved using the carrier mapping services. These services provide the communications required of the appropriate module resources by the material models through defining the material slot object.

The carrier-mapping services are fully specified in terms of the material slot attributes accessed through the get and set services of SEMI E39 (Object Services Standard: Concepts, Behavior, and Services). Cluster tool module communications require full compliance with the information, behavior, and messaging services of the fundamental requirements of that standard.

The mapping information is only of interest to the cluster tool for a carrier contained in the module resource. The material slot objects are, therefore, associated with, and managed by, the module resource which uniquely identifies all of its material slots. The carrier-mapping services provide for communication of material slot information in both directions between the cluster controller and the module resource. The information communicated is material presence and material identification. Whether the material presence is detected by the module resource or determined by the cluster controller is application-specific. This is also the case for material identification.

A material slot may be assigned the identifier of material which is not yet occupying the material slot. This provides for determination by the cluster controller of which material slot the material is to be moved into when it arrives at the module.

Material slot object attributes required to be accessible through object services are detailed in Table 2.

Table 2 Material Slot Attributes

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ObjType	The object type.	Y	RO	Text: “MATERIALSLOT”
ObjID	Module unique identifier for the material slot.	Y	RO	Text: Unique with respect to the module.
MtlName	Identifier of the material occupying or assigned to the slot.	Y	RW	Text
OccState	Current state of the material slot.	Y	RW	Text: SlotUnknown SlotEmpty SlotOccupied SlotActive SlotDisabled

The material slot occupancy attribute indicates the presence of material as follows:

- SlotUnknown — the carrier is present and slot occupancy is not known, such as from when a carrier is loaded until the material is detected or the state is set by the cluster controller.
- SlotEmpty — no material in the slot.
- SlotOccupied — material is present in the slot.
- SlotActive — material is being removed from, or is arriving at, the slot.
- SlotDisabled — material slot unusable, such as when the carrier is not present.

Module resource capabilities required, but not already specified above:

- Reject incomplete, invalid, and unsupported requests.

Module resource capabilities permitted, but not to be required, by the cluster controller:

- Detection of the presence of material in a material slot.
- Detection of the identifier of material in a material slot.

Table 3 lock Attributes

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ObjType	The object type.	Y	RO	Text: “CLOCK”
ObjID	Module unique identifier for the clock.	Y	RO	Text: Unique with respect to the module. Services timestamp clock is “SERVICESCLOCK”
DateTime	Current time.	Y	RW	Text: yyyymmddhhmmsscc

8.4 *Exception Management Services* — Exception management across a cluster tool module communications interface is achieved using the exception management services. These services provide the communications and behavior required of the module resource by the exception models described in Section 7.1.5 through defining the exception condition object and its operations.

The exception management services follow SEMI Draft Document 2013C (SECS II Support for SEMI E41 (Exception Management Standard)). Cluster tool module communications require full compliance with the information, behavior, and messaging services of the fundamental requirements of that standard.

Exception conditions (alarms and error conditions) are created and managed by the module resource detecting the related abnormal situation. The module resource uses the EXPost service to report the occurrence of the abnormal situation or some significant change in exception condition information while the situation exists. EXCleared reports that the abnormal situation is no longer apparent or relevant. Reporting on an exception condition is enabled and disabled by setting its EXEnabled attribute.

The cluster controller may direct recovery of error conditions using the EXRecover service. The module resource performs the recovery action and reports its completion using the EXRecoveryComplete service. The module resource is only required to permit a single recovery action at a time on all set exception conditions, but may support multiple recovery actions. The module resource shall support the EXRecoveryAbort service, which ceases any activity and terminates the recovery action.

Exception condition object attributes required to be accessible through object services:

Attribute Name	*Description: Requirements*
ObjType	The object type: EXCEPTION.
EXID	Identifier for the exception condition.
EXType	Type of exception condition:alarm or error condition.
EXMessage	Text message describing the abnormal situation monitored.
EXEnabled	Indicates reporting to decision authority enabled/disabled.
EXRecActList	Possible recovery actions (none for alarms).
EXStateList	All concurrent substates of the exception.

Exception management messaging services required:

Message Name	*Description: Requirements*
EXPost	Notify/update on abnormal situation.
EXCleared	Notify resolution of abnormal situation.
EXRecover	Command to perform recovery action.
EXRecoveryComplete	Notify completion of recovery action.
EXRecoveryAbort	Command to abort a recovery action.

Reporting of all exception conditions is set to be enabled on establishment of an association, and those in the set state are posted.

Module resource capabilities required, but not already specified above:

- Reject incomplete, invalid, and unsupported requests.

Module resource capabilities permitted, but not to be required, by a service-user:

- Multiple concurrent recovery actions.
- Dynamic update of exception condition message and valid recovery actions.

8.5 *Recipe Management Services* — Recipe management across a cluster tool module communications interface is achieved using the recipe management services. These services provide the communications and behavior required of the module by the recipe models described in Section 7.1.6 of this document.

The recipe management services follow SEMI E42 (Recipe Management Standard). Cluster tool module communications require full compliance with the information, behavior, and messaging services of the fundamental requirements of a recipe executor, and optionally the recipe namespace management, as defined in that standard.

The module requiring recipe services contains a recipe executor that is capable of performing recipe download, verification, selection, deselection, and deletion as requested by the cluster controller. The module may also support recipe namespace services.

In certain configurations, the module may require recipe management services to be provided by the cluster controller, in order that it may request a selected recipe that is not stored locally. The module is the service-user in this case, and the cluster controller should support the necessary recipe namespace services.

The module coordinates the activities of the processing resource and the recipe executor to ensure the appropriate recipes are loaded into the execution area for material processing.

Recipe executor object attributes required to be accessible through object services:

Attribute Name	*Description*
ObjType	The object type: “RcpExec”
ObjID	Identifier for the recipe executor
DefaultNamespace	Namespace for all hardware-dependent recipes
RecipeSelectID	List of recipe identifiers for the currently selected recipe

Recipe management messaging services required:

Message Name	*Description: Requirements*
RMEDnldVer	Receive a recipe, optionally verify it, and put it into storage
RMEDelete	Delete a recipe from storage
RMESelect	Select recipes for execution
RMEDeselect	Deselect a recipe to prevent its execution
RMEComplete	Notify service-user of completion of an action

Modules that require more than one recipe class for an execution cycle shall support multi-part recipes. That is, the ability to select multiple recipes in a single process job request is not supported in processing management.

Recipe executor capabilities required and not already specified above:

- Reject incomplete, invalid, and unsupported requests

Recipe executor and module capabilities permitted by not to be required by a service-user:

- Support for variable parameters (attributes and select service),
- Support for multi-part recipes,
- Upload a recipe,
- Support recipe namespace services for local recipe management,
- Local editing.

8.6 *Event Reporting Services* — Event reporting services are to be supported in accordance with the models described in Section 7.1.7. Both event-based reporting and tracing are required. Support for user-defined reports is not required.

Event reporting for CTMC-compliant modules shall be based on services as described in SEMI E53.

8.6.1 The attributes defined in the following table shall be supported by CTMC compliant modules.

Attribute Name	*Description: Requirements*
ObjID	The identifier for Data Reports, Event Reports, and Trace Reports.
DataReportList	List of data report identifiers.
Enabled	A flag which controls generation of Event and Trace Reports.
SamplePeriod	Time delay between samples in a Trace report.
TotalSamples	Maximum number of samples to include in a Trace report.
GroupSize	Number of samples to include before sending a report to a service user.

8.6.2 Support for the following messages from SEMI E53 shall be provided.

Message Name	*Description: Requirements*
EventReportRequest	User can ask for a report to be generated and sent.
EventReportSend	Provider generates and sends report on an event.
DataReportCreate	Define attributes to be sampled for a report.
DataReportDelete	Delete report definitions
DataReportRequest	User requests sample and send of a report.
CollectionEventLink	Link data reports to a collection event.
CollectionEventUnlink	Unlink a data report
TraceReportCreate	Define a trace report
TraceReportDelete	Delete trace report definitions
TraceReportRequest	User requests a sample and send of a trace data.
TraceReportSend	Service provider sends a trace report to the user.
TraceReportReset	A trace reporting is set to its IDLE state.

8.6.3 Support for dynamic and user defined reports is optional. However, implementations which support these advanced capabilities, shall comply with the specifications of SEMI E53 and SEMI E39. SEMI E39 service messages shall use the optional object specifier arguments, such as EvtSrcSpec. Object services can then be used to interrogate objects for their reportable attributes.

8.7 *Clock Services* — The clock services provide for synchronization of the clocks on the modules with that of the cluster controller. These services provide the communications required of the appropriate module resources through defining the clock object.

The clock services are fully specified in terms of the clock attributes accessed through the get and set services of SEMI E39 (Object Services Standard: Concepts, Behavior, and Services). Cluster tool module communications require full compliance with the information, behavior, and messaging services of the fundamental requirements of that standard.

The clock object is uniquely identified for the module. Table 3 describes the attributes of the clock object. The identifier of the clock used for communications services is defined. The clock attribute Time is kept current and used to determine the value of the timestamp used in service messages. The module returns the current Time when it receives a get.

The clock time is synchronized by using the object set service on the Time attribute. On receipt of a set request, the module performs the operations necessary to set the clock to the value supplied, within its resolution, from which it shall continue immediately.

The DateTime attribute of the Clock object as used in Clock Services has a resolution in the range of seconds to centiseconds. If it is possible to specify the DateTime accurate to centiseconds, then centiseconds should be specified. If it is not possible to resolve time to less than a second, then centiseconds shall be reported as “00.” This requirement is for the reporting of the DateTime attribute of the Clock object in Clock Services only; other services may require greater resolution, in which case the Clock Services DateTime attribute resolution shall also be greater.

8.8 *Required Services* — Within a standards-compliant Cluster Tool, a number of services must be supported, at least to the fundamental requirements level. Due to the specialization of the modules in a Cluster Tool as detailed in the concepts section above, each module type only needs to support a subset of the service groups as shown in the message flow diagrams.

The services requirements for each module controller are described in this section. An example of use of services by a cluster controller is also given.

8.8.1 *Transport Module Controller*

- Material transfer functions for the end-to-end establishment of material movement within the cluster (supported through the Material Movement services). Material transfer is set up through a Transfer Job requested of the transport module by the service-user (e.g., cluster controller).
- Material Handoff between the process and cassette modules and the transport module (supported through the Material Movement services).
- Exception capabilities to post and manage exception conditions within the transport module (supported through the Exception Management services).
- Reporting of module variable data linked to events to the service-user (supported through the Event Reporting services).
- Access to standard objects of the specific services above as required by the service-user (supported through the Object services).

8.8.2 *Cassette Module Controller*

- Material I/O functions for the management of moving cassettes in and out of the cluster as well as for moving material between the cassette module and the transport module (supported through the Material Movement services). Material I/O is set up through a Transfer Job requested of the cassette module by the service-user (e.g., cluster controller).
- Material Handoff between the cassette module and the transport module (supported through the Material Movement services).
- Exception capabilities to post and manage exception conditions within the transport module (supported through the Exception Management services).
- Reporting of module variable data linked to events to the service-user (supported through the Event Reporting services).
- Access to standard objects of the specific services above as required by the service-user (supported through the Object services).

8.8.3 *Process Module Controller*

- Material processing functions (supported through the Processing Management services). Processing is specified through a Process job requested of the process module by the service-user (e.g., cluster controller).
- Recipe Management to verify and transfer recipes between the process module and the service-user (supported through the Processing Management services).
- Material I/O functions for the management of moving material in and out of the process module (supported through the Material Movement services). Material I/O is set up through a Transfer job requested of the process module by the service-user (e.g., cluster controller).
- Material Handoff between the process module and the transport module (supported through the Material Movement services).
- Exception capabilities to post and manage exception conditions within the transport module (supported through the Exception Management services).
- Reporting of module variable data linked to events to the service-user (supported through the Event Reporting services).
- Access to standard objects of the specific services above as required by the service-user (supported through the Object services).

8.8.4 *Example Cluster Controller*

- Scheduling and primary module control. Schedules Transfer and Process jobs in order to achieve the mission of the cluster tool, the processing of material.

- Recipe management for storage, archival, and editing of process module recipes (supported through the Recipe Management services).
- Human interfaces for the attached process, transport, and cassette modules (supported through the Event Reporting and Object Service definitions).
- Exception resolution services for making recovery decisions on exception conditions within the cluster (supported through the Exception Management services).

APPENDIX 1
APPLICATION NOTES

NOTE: This appendix was approved as a part of SEMI E38 by full letter ballot procedure.

A1-1 Factory Integration

The Cluster Tool Module Communications (CTMC) Standard and its associated services define the interactions of the cluster modules and the cluster controller. The standard does not address the communications between the cluster and an external factory "host." The SEMI standard which addresses the equipment to host communications interface is SEMI E30 (GEM).

The purpose of this application note is to discuss the requirements and possible pitfalls of the factory host-cluster tool interface. The bulk of the text will address the use of the current (at this writing) version of GEM (SEMI E30). However, some consideration will be given to the possibility of direct access to the components of the cluster through the cluster communications environment.

A1-1.1 *Direct Access to Cluster Modules* — The CTMC was designed to allow for interoperability among the entities in a cluster. It is possible to add a new entity to the cluster and share in this interoperability. This means that a software application that is "plugged" into the cluster communication environment will have access to all of the cluster modules via the standard message set. In this way, a factory host computer can gain direct access to cluster modules, by-passing the cluster controller.

While direct access to the cluster modules has some advantages, intrusions into the cluster tool communications environment can have a profound effect on the system. Below are listed a few of the positive benefits direct access can bring, followed by some of the negatives of such an approach.

Positives:

- If the cluster controller does not have to act as the intermediary in transactions between module and host, it may be simplified.
- Direct access of the modules can be a more efficient means of obtaining information.
- The factory host would have the flexibility to perform scenarios which the cluster controller suppliers had not imagined.

Negatives:

- A new, unplanned load on the cluster network may have a negative impact on system performance. If the delivered system's network is highly loaded and has been tuned for that load, added pressure on the system may have unexpected results.
- Direct access of the cluster modules may have a negative impact on system performance. The cluster modules are performing the physical work for the cluster. If they become busy answering inquiries, it may detract from their speed of processing. For example, a simple query, such as temperature of chamber, may actually result in messaging to the sensor itself along the same real-time communication path used to open and close valves, ect.
- System integrity of the cluster tool may be at risk if an external entity (e.g., Factory Host) takes even minor control actions. A changed equipment constant may put the cluster controller out of synch with its module. An external command to perform an action may directly conflict with the cluster's current actions.
- The CTMC defines no access security to prevent undesirable actions on the part of external entities.

If there is to be this sort of direct communication with cluster modules, the implementer should exercise extreme caution in the design of the add-on application. It is recommended that no active control be attempted and that no changes to the system configuration be attempted (e.g., do not set attributes). If data is to be accessed, it is best done by asynchronous event based reporting, rather than by polling. Above all, it is recommended that any plans to access the cluster through the cluster communication environment be discussed thoroughly with the system supplier.

A1-1.2 *GEM Control of a Cluster Tool* — This section discusses the interactions between a cluster tool and factory host. The basis for that communication is assumed to be SEMI E30, the Generic Equipment Model.

GEM was designed relative to the prevalent class of equipment at that time: proprietary, single supplier, and limited to a single process run at a time. The advent of multi-chamber processing equipment and later of multi-supplier cluster tools has changed the general requirements set for the factory-equipment interface. In the future, GEM may evolve to meet these new requirements. Regardless, the point of this application note is to explore how best to apply the existing factory host communication standards to cluster-based equipment.

Below, each GEM capability will be addressed separately, followed by a look at capabilities which GEM does not cover. The capabilities are taken in roughly the order given by SEMI E30 (GEM) table of contents.

Communications State Model/Establish Communications — The Communications State Model is related to the ability to communicate. It is independent of the functionality offered by the equipment. There is no conflict with the cluster tool supporting this GEM capability.

Control State Model — The control state model assumes a single operator interface at the equipment. A cluster tool is expected to supply such an interface connected to the cluster controller.

However, there is a potential disconnect with attached modules which supply user interfaces. Since CTMC does not provide anything similar to the control state model at this time, the cluster controller has no control over access to these separate user interfaces, nor even any knowledge of what might be happening. In this situation, the cluster controller cannot guarantee compliance to the control state model.

To assure GEM compliance, it is recommended that a module request and receive closure of all control-related service connections before allowing use of its local operator interface in a control mode. This assumes the use of the communication environment as defined in SEMI E38.1.

Equipment Processing States — GEM specifies that there be a processing state model. The model given in the document is only an example, which shows the approximate depth needed and suggests a form. The example applies to the classic single chamber, single-process machine. Thus, it may not be directly applicable to a cluster tool. The cluster tool is required to provide a processing state model, but the form should match the cluster tool's function. The Plasma Etch Specific Equipment Model (document available from SEMI) suggested one way to model a multi-chamber, multi-process job machine.

Data Collection — Data collection is a combination of several different capabilities. Each will be discussed in turn.

There is also a high-level issue with data collection. GEM assumes a flat address space. That is, it assumes that the smallest granularity of object is “equipment.” The equipment has attributes. Some attributes are read-only (Status Variables). Some have read/write access (Equipment Constants). There are a few read-only attributes which are valid only at certain times (DVVALs).

Cluster tools do not fit this model well. The CTMC was based on a model which includes a hierarchy of objects which contain (or control) other objects. The CTMC requirements are a subset of that model and include a cluster controller (which does not have to be a single entity), attached modules, transport module, and specific objects associated with the services. Thus, to request information about the processing of a wafer in a process module, the attributes of the process job must be accessed. The chain then includes Cluster Controller->Process Module->Process Job->Job Attribute. It is quite a challenge to present this attribute as SVID #532678. And yet, this is what must be done.

The cluster controller should provide a flat name space from which to access critical module data. Object services with scopeing capabilities may also be supported through the host interface, but if used, this would be an extension to the host's GEM interface. The data collection categories below give more detail on the challenge and possible solutions.

Event Notification — Many of the events of interest to the host actually occur at the attached modules (e.g., “Etch step complete for wafer x”). The cluster controller is responsible for assigning each of these events a unique CEID. There are two challenges: (1) to assure uniqueness of events from duplicate modules, and (2) to provide a scheme whereby the host can determine the source of the event. Embedding a module number within the CEID is one solution which has been used to address both challenges. For instance, the highest order byte of a four-byte integer might contain the module ID.

Dynamic Event Report Configuration — The host must be able to attach the information of interest to the event notifications. The cluster controller is responsible for providing access to the attributes of the cluster modules. It is also responsible for assuring that the data contained in the event reports is representative of the state of the cluster at the time the event occurred.

There are three sources of data for the event reports.

1. Cluster Controller: Data which is local to the cluster controller may be accessed directly.
2. Module (not source of collection event): This data must be polled unless a data trace for the attributes of interest is in progress.
3. Module (source of collection event): This data tends to be most closely related to the event and thus the most time-critical. While polling and data trace may be used in this case, it is better to use the

CTMC Event Reporting Services to attach the data of interest directly to the module collection event.

Variable Data Collection/Status Data Collection — Again, the cluster must take measures to present a flat address space. See various discussions above.

Limits Monitoring — Clearly, most variables which would be of interest for limits monitoring reside in the cluster modules. The CTMC does not provide specific services for configurable limits monitoring. A module may provide for some limits checking via "limits" attributes of objects which may be set via object services. Notification of excursions beyond limits would be via exceptions and/or collection events. If this method is not available, the cluster controller would have to institute a data trace on the attribute of interest and then itself perform the limits checking activity. The time period of the trace would be crucial to assuring that short-term excursions are not missed.

Trace Data Collection — Trace data collection maps well from GEM to CTMC. In fact, the CTMC services in this area are a superset of the GEM functionality.

On-Line Identification — At first glance, this seems to be an easy requirement to meet. However, the model number and software revision included in the message are ambiguous on a cluster with separate model numbers for each module (and the cluster controller itself) and separate software releases running on each. The purpose of these data items is to allow the host to determine when the equipment has been changed in such a way that it may no longer be compatible with the host software. At a minimum, the data items should indicate a change to the cluster controller software (e.g., a new revision of the software, or a significant reconfiguration such as the addition of a new module).

Alarm Management — GEM alarm management is a subset of the functionality of the CTMC exception services. Any exception message from a module can, where desired, be translated into a GEM-compliant alarm message. The set of CTMC exceptions is a superset of GEM alarms. GEM alarms are of a serious nature and expect human intervention to be required. The CTMC extends this set to include problems which the controller may be able to solve, and provides for modules to suggest appropriate recovery actions.

Those CTMC exceptions which fall into the GEM alarm category should be passed on via the GEM Alarm capability. Exceptions which are not GEM alarms should be reported only as events.

Remote Control — Cluster tools need to be able to handle multiple tasks simultaneously. Most need to handle multiple lot or batch jobs at once. GEM's support for (process) job control is minimal. The implicit assumption in GEM is that only one job at a time will occur on a machine. It provides for a "START" command, collection events when the job is actually started and when it finishes, and also job control commands (e.g., PAUSE, RESUME, ABORT, STOP). What GEM lacks is differentiation of one job from another (e.g., a job ID).

However, the GEM remote command message does provide for parameters. A cluster tool may be able to implement a scheme which is similar to that defined by the Process Management Services. For instance, a "job ID" parameter may be included in all job-related messages. Thus, the equipment would set a job ID in the "Start" or the "PP-Select" message (whichever first refers to the job), and would then refer to that ID in any subsequent job control commands. The job ID could also be available as a data value in collection event reports. While it might be preferable to have the equipment assign the job ID, this scheme may be workable.

Equipment Constants — Again, the flat address space issue applies. (See the Data Collection section above.)

Process Program Management — The CTMC uses Recipe Management Services (RMS) to manage recipes internal to the cluster. GEM's processing services is a different mechanism. The two can be made to interact, but the combination is not ideal. It is possible for the cluster to translate a "recipe" to a "process program" for communication with the host. However, if a recipe is created or modified at the host and then downloaded, the editor at the host must be compatible with RMS concepts, and care must be taken to maintain the integrity of the RMS system.

Material Movement — GEM requires only collection events when material is received or sent. A cluster can easily satisfy this requirement.

Equipment Terminal Services, Error Messages, and Spooling — Equipment Terminal Services, Error Messages, and Spooling are in the domain of the cluster controller. Cluster modules do not provide these functions, nor do they contribute to the cluster controller's delivery of the services.

Clock — A key purpose of clock is to allow the host to determine ordering of events from an equipment. The cluster controller should ensure that this need is met. The cluster should attempt to report consistent time relative to its modules. Also, when the factory host sets the time on the cluster, the cluster controller should resynchronize the time on its modules. There should be no compatibility problems in this area.

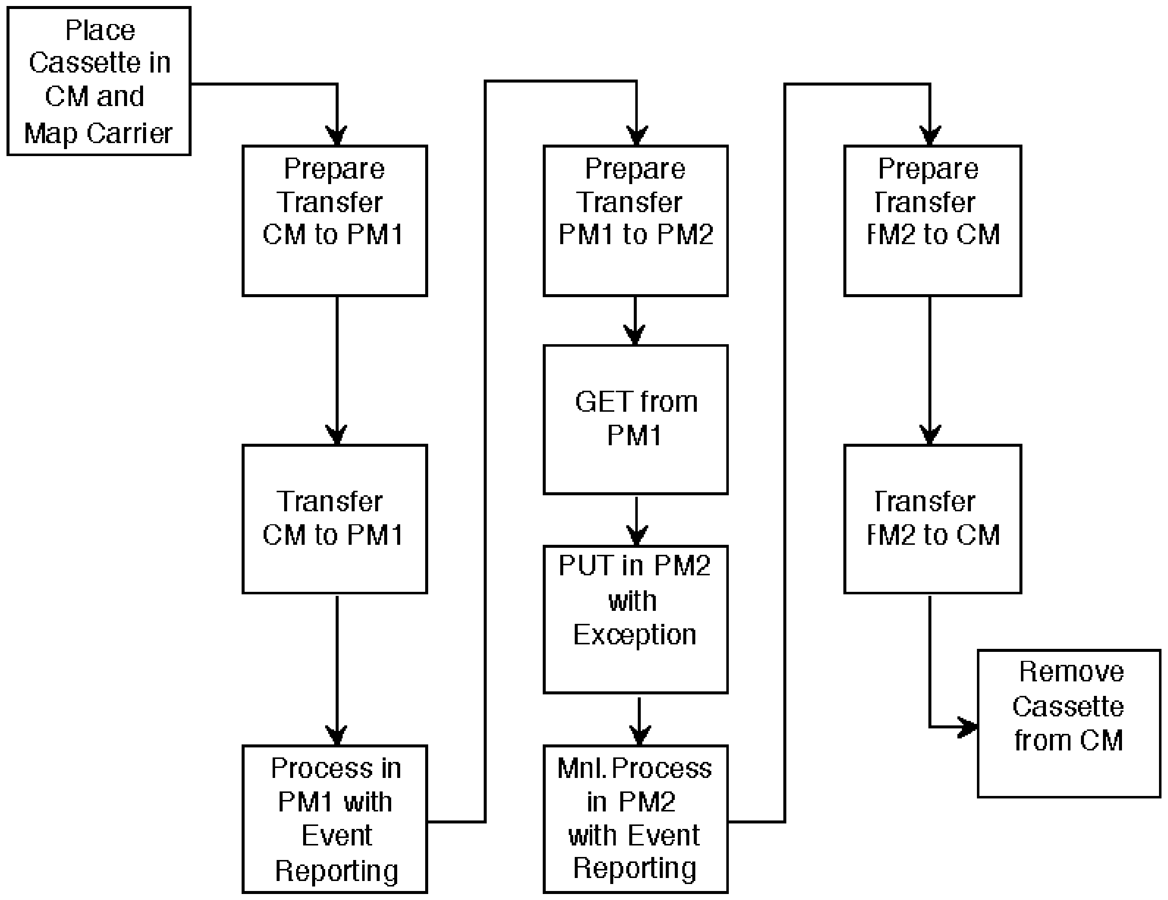
Place Cassette in CM and Map Carrier
Prepare Transfer CM to PM1
Transfer CM to PM1
Process in PM1 with Event Reporting
Prepare Transfer PM1 to PM2
GET from PM1
PUT in PM2 with Exception
Mnl. Process in PM2 with Event Reporting
Prepare Transfer PM2 to CM
Transfer PM2 to CM
Remove Cassette from CM

Figure 20
Typical Operating Scenario

A1-2 Typical Operating Scenario

The purpose of this section is to give an example of a typical operating scenario that will illustrate the various services being used to perform cluster tool control functionality. This operating scenario is **NOT** intended to be the only scenario to be implemented in a standards-compliant cluster. The intent is to provide guidance in navigating the various services standards. An exhaustive use of Object Services and Exception Management Services will not be done since these two areas tend to be very application-specific.

This scenario concentrates on the material processing, material movement aspects of the cluster-specific services.

The operating scenario shown in Figure 20 will be given in the form of a communication timing diagram with messages being sent between the various cluster tool components and will be indicated in the following manner:

ServiceGroup_ServiceName.Request or Response (Service Parameters)

Example: TRJobCreate.Req(...)

- TR is the service group that contains the service, the material movement services.
- JobCreate is the service being used to cause communication to take place between two cluster tool components. Specifically, this service is the material movement service that initiates the transfer of material between modules.
- Req indicates that the service is a Request.
- (...) indicates any other parameters that might be present with the service definition and will be service-dependent.

Definitions:

CC = Cluster Controller

CM = Cassette Module

TM = Transport Module

PM1 = Processing Module #1

PM2 = Processing Module #2

The general routing for the scenario is illustrated in the message sequence below:

1. A cassette of wafers arrives from the factory and is given to the cassette module.
2. A mapping of the wafers in the various slots of the cassette is determined for use inside the cluster.
3. A transfer job is initiated to move a wafer from the cassette to a process module (in this case PM1).
4. The cassette module is commanded to send the wafer to the transport module.
5. The PM1 is instructed to create the process job and to retrieve any pertinent recipe information in order to process the wafer.
6. The wafer is handed off between the cassette module and the transport module (Wafer GET operation).
7. The wafer is handed off between the transport module and PM1 (this is a Wafer PUT operation).
8. The wafer is processed in PM1.
9. A transfer job is initiated to the transport module to move a wafer from PM1 to PM2.
10. The PM1 is commanded to send the wafer to the transport module.
11. The PM2 is instructed to create the process job and to retrieve any pertinent recipe information in order to process the wafer.
12. The wafer is handed off between the PM1 and the transport module with a GET.
13. The wafer is handed off between the transport and PM2 with a PUT.
14. The wafer is processed in PM2 when manually started.
15. A transfer job is initiated to the transport module to move a wafer from PM2 back to the cassette module.
16. The PM2 is commanded to send the wafer to the transport module.
17. The cassette module is instructed to receive the wafer.
18. The wafer is handed off between PM2 and the transport module.
19. The wafer is handed off between the transport module and the cassette module.
20. The cassette module is instructed to send the cassette to the factory.

1. FACTORY TO CM CASSETTE HANDOFF

The cassette containing a single wafer in wafer slot 1 to be processed by the cluster tool is passed from the factory environment to the cluster tool.

At this point the user (human or host) will be prompted to place the cassette on the cassette module loading system. The cassette module is then informed to receive a cassette from the external environment.

```
CC                                      CM
>>-------------------------------------->>
TRJobCreate.Req(MID="Carrier1",Port=Port2
, Dir=Receive)
CC                                      CM
<<--------------------------------------<<
TRJobCreate.Rsp(JobID=IOJob1,Status=OK)
```

The cassette module prepares to receive the cassette by such actions as loadlock venting, door opening, and then requesting handoff from the Factory.

```
CC                                      CM
<<--------------------------------------<<
TRJobStarted.Req(JobID=IOJob1, TimeStamp)
```

The actual cassette transfer takes place, the cassette module takes such actions as door closing and loadlock pumping down to internal transfer pressures, and then reports completion of the material transfer.

```
CC                                      CM
<<--------------------------------------<<
TRJobComplete.Req(JobID=IOJob1,TimeStamp,
Status=OK))
```

2. WAFER MAPPING

The wafers in each of the cassette wafer slots are assigned a unique wafer ID. In this case, the wafer is identified as “Wafer1.” If there were more than one wafer, this process would be repeated until all the wafers had been given their appropriate identification that is used later when specific wafers are requested for transfer and processing.

```
CC                                    CM
>>-------------------------------------->>
Set.Req("WAFERSLOT",SlotID=1,MID=
"Wafer1")
CC                                  CM
<<------------------------------------<<
Set.Rsp("WAFERSLOT",SlotID=1,Status=OK)
```

3. CC TO TM TRANSFER JOB SETUP

The CC informs the TM that a wafer transfer is required from the CM to PM1.

```
CC                                  TM
>>------------------------------------>>
TRJobCreate.Req(MID="Wafer1",TRSourceAmID
="CM",TRDestinationAmID="PM1")
```

The transport module assigns an identifier for this particular transfer job. This is important in an actual cluster implementation since multiple transfer jobs may be queued and active at any given time. The transfer job is then started by the transport module.

```
CC                                   TM
<←-------------------------------------<<
TRJobCreate.Rsp(JobID=TMJob1,Status=OK)
CC                                   TM
<<--------------------------------------<<
TRJobStarted.Req(JobID=TMJob1,TimeStamp)
```

4. CC TO CM SEND WAFER

The CC informs the CM that a wafer is to be transferred to the TM.

```
CC                                    CM
>>-------------------------------------->>
TRJobCreate.Req(MID="Wafer1",PortID=
"Port1",Dir=Send)
CC                                    CM
<<--------------------------------------<<
TRJobCreate.Rsp(JobID=IOJob2,Status=OK)
```

When the cassette module is ready, it directly communicates with the cluster controller and the transport module to indicate its readiness for wafer handoff.

```
CC                                      CM
<<--------------------------------------<<
TRJobStarted.Req(JobID=IOJob2, TimeStamp)
TM                                      CM
<<--------------------------------------<<
HOReady.Req(PortID="Port1",AmID="CM",
HODir=GET,MID="Wafer1")
```

5. CC TO PM1 PROCESSING JOB SETUP

The CC informs PM1 that a wafer will be delivered (which requires a TRJob to be created) and is to be processed (this requires that a process job be established). This involves creating the jobs and transferring the appropriate recipes to the process module as required. At some point in this process, the PM will inform the transport module that it is ready to receive the wafer with the issuing of the HOReady Request.

The CC informs PM1 that a wafer is to be transfered from the TM.

```
CC                                    PM1
>>-------------------------------------->>
TRJobCreate.Req(MID="Wafer1",Port="Port1"
, Dir=Receive)
CC                                    PM1
<<--------------------------------------<<
TRJobCreate.Rsp(JobID=IOJob3,Status=OK)
```

When the process module is ready, it directly communicates with the cluster controller to indicate its readiness for wafer handoff.

```
CC                                     PM1
<<--------------------------------------<<
TRJobStarted.Req(JobID=IOJob3, TimeStamp)
```

The process job is created on the process module.

```
CC                                     PM1
>>-------------------------------------->>
PRJobCreate.Req(RecID="PP1",MID="Wafer1")
CC                                     PM1
<<--------------------------------------<<
PRJobCreate.Rsp(JobID=PM1Job1,Status=OK)
CC                                     PM1
<<--------------------------------------<<
PRJobSetup.Req(JobID=PM1Job1)
```

The process module requests that all sections (headers and the body) of the recipe be downloaded.

```
CC                                     PM1
<<------------------------------------<<
RCRequest.Req(RecID="PP1",RCTransfer=all)
CC                                     PM1
>>------------------------------------>>
RCRequest.Rsp(RecID="PP1",Status=OK)
CC                                     PM1
>>------------------------------------>>
RCSend.Req(RecID="PP1",<Headers & Body>)
CC                                     PM1
<<------------------------------------<<
RCSend.Rsp(RecID="PP1",Status=OK)
```

The process module informs the transport module that it is ready to receive the material.

```
TM                                     PM1
<<------------------------------------<<
HOReady.Req(Port=Port1,AmID="PM1",HODir=P
UT,MID="Wafer1")
```

6. CM TO TM WAFER HANDOFF

The CM transfers the wafer to the TM through the handoff sequence, and the CM and TM will keep the cluster controller informed of its transfer progress.

The transport module informs the cassette module that it is proceeding with the "getting" of the wafer.

```
CM                                      TM
<<------------------------------------<<
HOReady.Req(Port=Port1,AmID="CM",HODir=GE
T, MID="Wafer1")
```

The cassette module informs the transport module to use the PICK sequence to get the wafer.

```
CM                                      TM
>>------------------------------------>>
HOPick.Req(Port=Port1,AmID="CM")
CM                                      TM
<<------------------------------------<<
HOPick.Rsp(Port=Port1,AmID="CM",
Status=OK)
```

When the getting of the wafer has been successful, the cassette module informs the transport module with a verify sequence.

```
CM                                      TM
>>------------------------------------>>
HOVerify.Req(Port=Port1,AmID="CM")
CM                                      TM
<<------------------------------------<<
HOVerify.Rsp(Port=Port1,AmID="CM",
Status=OK)
```

The cassette module then informs the cluster controller of its current status.

```
CC                                      CM
<<------------------------------------<<
TRJobComplete.Req(JobID=IOJob2,TimeStamp,
Status=OK)
```

7. TM TO PM1 WAFER HANDOFF

The TM now has possession of the wafer and is ready to proceed with the putting of the wafer into the process module. The TM and PM1 will inform the cluster controller of the progress of the transfer of material.

The TM informs the PM that it is proceeding with the putting of the wafer.

```
PM1                                     TM
<<------------------------------------<<
HOReady.Req(Port=Port1,AmID="PM1",HODir=P
UT,MID="Wafer1")
```

The process module informs the TM to use the PLACE scenario as the put operation.

```
PM1                                     TM
>>------------------------------------>>
HOPlace.Req(Port=Port1,AmID="PM1")
PM1                                     TM
<<------------------------------------<<
HOPlace.Rsp(Port=Port1,AmID="PM1",
Status=OK)
```

When the putting of the wafer has been successful, the PM informs the transport module with a verify sequence.

```
PM1                                   TM
>>------------------------------------>>
HOVerify.Req(Port=Port1,AmID="PM1")
PM1                                   TM
<<------------------------------------<<
HOVerify.Rsp(Port=Port1,AmID="PM1",
Status=OK)
CC                                   PM1
<<------------------------------------<<
TRJobComplete.Req(JobID=IOJob3,
TimeStamp)
```

The TM then informs the cluster controller that the putting phase of the transfer is complete and then that the requested transfer job is complete.

```
CC                                    TM
<<------------------------------------<<
TRJobComplete.Req(JobID=TMJob1,
TimeStamp,Status=OK)
```

8. PM1 JOB PROCESSING

The process module informs the cluster controller, now that is has received the material and having previously entered the setup state, that it is ready to initiate processing on the wafer. When actual processing is initiated, the PM informs the cluster controller of its current status.

```
CC                                   PM1
<<------------------------------------<<
PRJobProcessing.Req(JobID=PM1Job1,TimeSta
mp)
```

A data collection is enabled to allow parameters to be sent from the PM to the cluster controller while the wafer processing is taking place. A previously defined event is enabled and the report structure is established.

```
CC                                   PM1
>>------------------------------------>>
Set.Req(EREvent,EventID=Event1,
Enabled=TRUE)
CC                                   PM1
<<------------------------------------<<
Set.Rsp(EREvent,EventID=Event1,Status=OK)
```

Two variables for time and temperature will be reported based on Event1 occurring.

```
CC                                   PM1
<<------------------------------------<<
ERReport.Req(EventID=Event1,ReportID=Rpt1
, ValueList={time,temp})
```

When wafer processing is complete on the PM, the cluster controller is informed of the current status.

```
CC                                   PM1
<<------------------------------------<<
PRJobProcessingComplete.Req(JobID=PM1Job1
, TimeStamp,Status=OK)
```

Data collection is then disabled.

```
CC                                   PM1
>>------------------------------------>>
Set.Req(EREvent,EventID=Event1,
Enabled=FALSE)
CC                                   PM1
<<------------------------------------<<
Set.Rsp(EREvent,EventID=Event1,Status=OK)
```

9. PM1 to PM2 TRANSFER JOB SETUP

The CC informs the TM that a wafer transfer is required from the PM1 to PM2.

```
CC                                    TM
>>------------------------------------>>
TRJobCreate.Req(MID="Wafer1",
TRSourceAmID="PM1",
TRDestinationAmID="PM2")
```

The transport module assigns an identifier for this particular transfer job. The transfer job is then started by the transport module.

```
CC                                    TM
<<------------------------------------<<
TRJobCreate.Rsp(JobID=TMJob2,Status= OK)
CC                                    TM
<<------------------------------------<<
TRJobStarted.Req(JobID=TMJob2, TimeStamp)
```

10. CC TO PM1 SEND WAFER

The CC informs PM1 that a wafer is to be transferred to the TM.

```
CC                                    PM1
>>-------------------------------------->>
TRJobCreate.Req(MID="Wafer1",Port=
"Port1", Dir=Send)
CC                                    PM1
<<--------------------------------------<<
TRJobCreate.Rsp(JobID=IOJob4,Status=OK)
CC                                    PM1
<<--------------------------------------<<
TRJobStarted.Rsp(JobID=IOJob4, TimeStamp)
```

When the process module is ready, it directly communicates with the transport module to indicate its readiness for wafer handoff.

```
TM                                    PM1
<<--------------------------------------<<
HOReady.Req(PortID="Port1",AmID="CM",
HODir=GET,MID="Wafer1")
```

11. CC TO PM2 PROCESSING JOB SETUP

The CC informs PM2 that a wafer will be delivered (requires a TRJob) and is to be processed (requires that a process job be established). This involves creating the jobs and transferring the appropriate recipes to the process module as required. At some point in this process, the PM will inform the transport module that it is ready to receive the wafer with the isssuing of the HOReady Request.

```
CC                                    PM2
>>-------------------------------------->>
TRJobCreate.Req(MID="Wafer1",Port="Port1"
, Dir=Receive)
CC                                    PM2
<<--------------------------------------<<
TRJobCreate.Rsp(JobID=IOJob5,Status=OK)
CC                                    PM2
<<--------------------------------------<<
TRJobStarted.Req(JobID=IOJob5, TimeStamp)
CC                                    PM2
>>-------------------------------------->>
PRJobCreate.Req(RecID="PP2",MID="Wafer1",
ManualStart)
CC                                    PM2
<<--------------------------------------<<
PRJobCreate.Rsp(JobID=PM2Job1,Status=OK)
CC                                    PM2
<<--------------------------------------<<
PRJobSetup.Req(JobID=PM2Job1)
```

The process module requests that all sections (headers and the body) of the recipe be downloaded.

```
CC                                    PM2
<<--------------------------------------<<
RCRequest.Req(RecID="PP2",RCTransfer=all)
CC                                    PM2
>>-------------------------------------->>
RCRequest.Rsp(RecID="PP2",LinkedID=0,{Bod
y: "TEMP 250 TIME 40"})
CC                                    PM2
>>-------------------------------------->>
RCSend.Req(RecID="PP1",<Headers & Body>)
CC                                    PM2
<<--------------------------------------<<
RCSend.Rsp(RecID="PP1",Status=OK)
```

The process module then informs the transport module to proceed with the putting of the wafer.

```
TM                                   PM2
<<------------------------------------<<
HOReady.Req(Port=Port1,AmID="PM2",HODir=P
UT,MID="Wafer1")
```

12. PM1 TO TM WAFER HANDOFF

The PM1 transfers the wafer to the TM through the handoff sequence, and the TM will keep the cluster controller informed of its transfer progress. The transport module informs the process module that it is proceeding with the "getting" of the wafer.

```
PM                                    TM
<<------------------------------------<<
HOReady.Req(Port=Port1,AmID="PM1",HODir=G
ET,MID="Wafer1")
```

The process module informs the transport module to use the PICK sequence to get the wafer.

```
PM1                                   TM
>>------------------------------------>>
HOPick.Req(Port=Port1,AmID="PM1")
PM1                                   TM
<<------------------------------------<<
HOPick.Rsp (Port=Port1,AmID="PM1",
Status=OK)
```

When the getting of the wafer has been successful, the process module informs the transport module with a verify sequence.

```
PM1                                   TM
>>------------------------------------>>
HOVerify.Req(Port=Port1,AmID="PM1")
PM1                                   TM
<<------------------------------------<<
HOVerify.Rsp(Port=Port1,AmID="PM1",
Status=OK)
```

Process module #1 then informs the cluster controller of its current status.

```
CC                                   PM1
<<------------------------------------<<
TRJobComplete.Req(JobID=IOJob4,TimeStamp,
Status=OK)
```

The cluster controller is informed that the processing job performed in PM1 is complete.

```
CC                                   PM1
<<------------------------------------<<
PRJobComplete.Req(JobID=PM1Job1,
TimeStamp,Status=OK)
```

13. TM TO PM2 WAFER HANDOFF

The TM informs the cluster controller that it is proceeding with the transfer of the wafer. The TM informs the PM that it is proceeding with the putting of the wafer.

```
PM2                                   TM
<<------------------------------------<<
HOReady.Req(Port=Port1,AmID="PM2",HODir=P
UT,MID="Wafer1")
```

The process module informs the TM to use the EXTEND scenario as the put operation.

```
PM2                                   TM
>>------------------------------------>>
HOExtend.Req(Port=Port1,AmID="PM2")
PM2                                   TM
<<------------------------------------<<
HOExtend.Rsp(Port=Port1,AmID="PM2",
Status=OK)
```

During this particular handoff, an exception is generated stating that the pins in the process module failed to come up and receive the wafer from the end effector. The cluster controller is given three recovery options to select from in attempting to resume operation in the presence of this exception.

```
CC                                   PM2
<<------------------------------------<<
EXPost.Req(EXID=Error1, EXType=Error,
EXMessage="pins failed to raise",
EXRecoveryList=
{"Abort","Retry","Continue"}, TimeStamp)
```

The cluster controller selects the “continue” option.

```
CC                                     PM2
>>-------------------------------------->>
EXRecover.Req(EXID=Error1,
EXRecoverySelected= "Continue")
CC                                     PM2
<<--------------------------------------<<
EXRecover.Rsp(EXID=Error1, Status=OK)
CC                                     PM2
<<--------------------------------------<<
EXRecoveryComplete.Rsp(EXID=Error1,
Status=OK)
```

This action is successful, and the alarm (exception type) clears.

```
CC                                     PM2
<<--------------------------------------<<
EXCleared.Rsp(EXID=Error1,EXType=Error,
TimeStamp)
```

When the putting of the wafer is now successful, the PM commands the transport module to Retract.

```
PM2                                     TM
>>-------------------------------------->>
HORetract.Req(Port=Port1,AmID="PM2")
PM2                                     TM
<<--------------------------------------<<
HORetract.Rsp(Port=Port1,AmID="PM2",
Status=OK)
```

Upon successful completion of the handoff, a verify sequence is used to complete the transaction.

```
PM2                                     TM
>>-------------------------------------->>
HOVerify.Req(Port=Port1,AmID="PM2")
PM2                                     TM
<<--------------------------------------<<
HOVerify.Rsp(Port=Port1,AmID="PM2",
Status=OK)
CC                                     PM2
<<--------------------------------------<<
TRJobComplete.Req(JobID=IOJob5,TimeStamp,
Status=OK)
```

The TM then informs the cluster controller that the putting of material is complete and that the requested transfer job is complete.

```
CC                                      TM
<<--------------------------------------<<
TRJobComplete.Req(JobID=TMJob2,
TimeStamp,Status=OK)
```

The process module informs the cluster controller that now that is has received the material, it is ready to initiate processing on the wafer upon Start command.

```
CC                                     PM2
<<--------------------------------------<<
PRJobWaitingStart.Req(JobID=PM2Job1)
```

14. PM2 JOB PROCESSING

The cluster controller commands the PM to start processing the wafer.

```
CC                                     PM1
>>-------------------------------------->>
PRJobStart.Req(JobID=PM2Job1)
CC                                     PM1
<<--------------------------------------<<
PRJobStart.Rsp(JobID=PM2Job1,Status=OK)
```

When actual processing is initiated, the PM informs the cluster controller of its current status.

```
CC                                     PM2
<<--------------------------------------<<
PRJobProcessing.Req(JobID=PM2Job1,
TimeStamp)
```

A data collection is enabled to allow parameters to be sent from the PM to the cluster controller while the wafer processing is taking place. A previously defined event is enabled and the report structure is established.

```
CC                                     PM2
>>-------------------------------------->>
Set.Req(EREvent,EventID=Event1,
Enabled=TRUE)
CC                                     PM2
<<--------------------------------------<<
Set.Rsp(EREvent,EventID=Event1,Status=OK)
```

Two variables for time and temperature will be reported based on Event1 occurring.

```
CC                                     PM2
<<--------------------------------------<<
ERReport.Req(EventID=Event1,ReportID=Rpt1
, ValueList={time,temp})
```

When wafer processing is complete on the PM, the cluster controller is informed of the current status.

```
CC                                     PM2
<<--------------------------------------<<
PRJobProcessingComplete.Req(JobID=PM2Job1
, TimeStamp,Status=OK)
```