

SEMI E102-0600

PROVISIONAL SPECIFICATION FOR CIM FRAMEWORK MATERIAL TRANSPORT AND STORAGE COMPONENT

This provisional specification was technically approved by the Global Information and Control Committee and is the direct responsibility of the Japanese Information and Control Committee. Current edition approved by the Japanese Regional Standards Committee on March 14, 2000 and by the North American Regional Standards Committee on March 2, 2000. Initially available at www.semi.org March 2000; to be published June 2000.

1 Purpose

1.1 Wafer fabrication factories will require the baseline capabilities of stocker storage and interbay transport. In addition to these baseline capabilities, intrabay transport will be added as a result of ergonomic and safety requirements brought about by the increased size and weight of 300 mm wafer carriers. These stocker, interbay transport, and intrabay transport systems will be required to be fully integrated with each other and the factory Manufacturing Execution System (MES) in order to realize the full vision of cost effective automated material transport to and from production equipment.

1.2 A baseline requirement of Automated Material Handling System (AMHS) equipment is efficient integration with the factory MES. Therefore, the purpose of this specification is to enable cost effective integration of interoperable AMHS systems, as illustrated in Figure 1. Manufacturers require a SEMI standard which specifies the visible behavior of the AMHS Integration system and its interface to the factory MES. The purpose of this standard is to specify these interfaces and the interactions between MES and AMHS systems as a part of the CIM Framework. While the term AMHS is commonly used to refer to a wide range of equipment and systems that support automated material handling, the CIM Framework specifies a software component called "Material Transport and Storage Component (MTSC)" which represents the standard interface of the MES to the complete suite of AMHS capabilities.

2 Scope

2.1 This specification provides the common interfaces required by Manufacturing Execution Systems for runtime interactions between the factory (as represented by other CIM Framework components) and the Material Tracking and Storage Component. Some interfaces supporting configuration and tracking of the material transport and storage equipment are the responsibility of the Equipment Tracking and Maintenance component of the CIM Framework. These interfaces are complementary to the interfaces of

Transport Machines and Storage Machines presented in this specification.

2.2 The responsibilities of the Material Transport and Storage Component include interfaces that

- Support scheduling of material transport and processing by predicting time for material delivery to specific locations. NOTE: The interfaces for delivery time prediction are deferred as one of the deficiencies noted in this provisional specification.
- Execute and monitor transport jobs to move material to specific locations.
- Validate that the job can be done: the material is available, the destination is reachable and has available storage or loadport capacity, and there are sufficient material movement resources (cars, storage space, etc.) to implement the job.
- Request operations from AMHS equipment controllers to enact internal material movement and storage actions. Implementations of the Material Transport and Storage Component will use lower level standards such as IBSEM and StockerSEM for communication with equipment that performs the physical movement and storage actions.
- Collect and record data on transport job execution and history.
- Capture, record, interpret, and respond to equipment events and fault detection.
- Generate MES-level job status events and material location change events.
- Record and report on material locations and material transport histories for material in the system.
- Interact with machine and port interfaces for material hand-off handshake protocols.

2.3 Automated Material Handling Systems (AMHS's) are an important part of any semiconductor factory and have, typically, been implemented and integrated with a

Manufacturing Execution System (MES) as a separate logical software entity. An AMHS is made up of the AMHS Framework, the system controllers for AMHS equipment (the software) and the transport and storage machines (the hardware). In keeping with this tradition, the CIM Framework views the AMHS as a “black box” where the interface into that system is visible, but the inner workings and control of that system is the responsibility of the AMHS supplier and not in the domain of the MES. The CIM Framework specifies the

Material Transport and Storage Component (MTSC) as the MES level interface to AMHS capabilities.

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

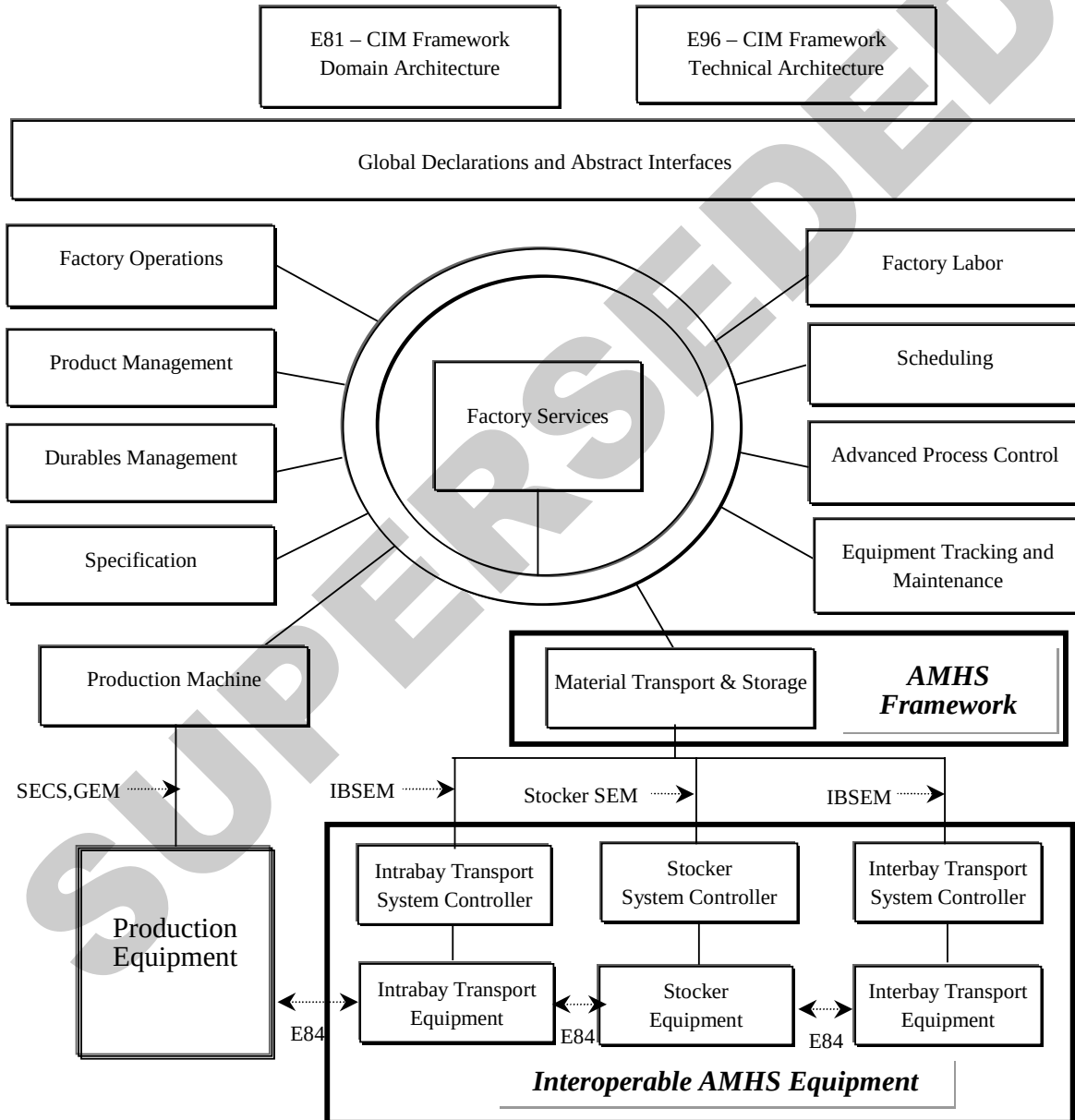


Figure 1
AMHS Environment

3 Limitations

3.1 The Material Transport and Storage Component provides AMHS Framework level interfaces to interoperable AMHS Equipment. This level of material transport is factory level movement, i.e., the transport of product and durables between machines, areas or material tracking locations. The Material Tracking and Storage Component does not provide interfaces that make material movement within machines visible.

3.2 *Provisional Status* — This specification is designated as provisional due to known areas that need to be completed. The following items summarize the deficiencies of the provisional specification to be addressed before a subsequent ballot to upgrade it to full standard status.

3.2.1 *Dependence on Global Declarations and Abstract Interfaces* — Components of this document reference global declarations and specialize abstract interfaces for material, resource and job concepts. The provisional specification for these declarations and interfaces was balloted and recommended by the SEMI Information and Control Committee in July, 1999. When this specification is published a revision will be undertaken to clearly define the relationship of this specification to those common elements of the CIM Framework.

3.2.2 *Dependence on Durables Management Component* — The Material Transport and Storage Component depends upon interfaces specified as part of the Durables Management Component of the CIM Framework. This document includes only the required interfaces from Durables Management. Those interfaces, found in Section 6.2.3, will be removed from this document when the Durables Management Component of the CIM Framework is adopted in a future ballot.

3.2.3 *Dependence on Machine Abstract Interface Group* — The Material Transport and Storage Component depends upon interfaces specified as part of the Machine Abstract Interface Group (Machine AIG) of the CIM Framework. This document includes only the required interfaces from the Machine AIG. Those interfaces, found in Section 6.2.4, will be removed from this document when the Machine AIG of the CIM Framework is adopted in a future ballot.

3.2.4 *Dependence on Factory Operations Component* — The Material Transport and Storage Component depends upon interfaces specified as part of the Factory Operations Component of the CIM Framework. This document includes only the required interfaces from the Factory Operations Component. Those interfaces, found in Section 6.2.5, will be removed from this

document when the Factory Operations Component of the CIM Framework is adopted in a future ballot.

3.2.5 *Deterministic Prediction of Transport Time* — The responsibility for providing estimates of predicted transport time of future TransportJobs in a deterministic way is beyond the scope of the Provisional Specification. This capability may be offered in future upgrades of this specification based on added experience with such predictions. It may be found that heuristic methods for transport time prediction are the most practical way to provide such estimates.

4 Referenced Standards

4.1 SEMI Documents

SEMI E81 — Provisional Specification for CIM Framework Domain Architecture

4.2 Other Documents

ISO/IEC International Standard 14750 (also ITU-T Recommendation X.920): Information Technology – Open Distributed Processing – Interface Definition Language¹

UML Notation Guide, Version 1.1, document number ad/97-08-05, Object Management Group²

5 Terminology

5.1 Abbreviations and Acronyms

5.1.1 *AMHS* — Automated Material Handling System

5.1.2 *MTSC* — Material Transport and Storage Component

6 Requirements

6.1 The following sections specify the interfaces that comprise the Material Transport and Storage Component of the CIM Framework. Additionally, the subsets of interfaces from Durables Management, Machine Abstract Interface Group, and Factory Operations required to support Material Transport and Storage are included here pending future ballots that will address the full scope of these other specifications.

6.1.1 Figure 2 illustrates the context of the Material Transport and Storage Component within the CIM Framework. The Factory Operation component requests the transport of material by the Material Transport Manager which provides services for creation

1 ISO Central Secretariat, 1, rue de Varembe, Case postale 56, CH-1211 Genève 20, Switzerland

2 UML Notation Guide v1.1 is available to the general public at <http://www.omg.org/cgi-bin/doclist.pl>, +1-508-820 4300, Object Management Group, Inc., Framingham Corporate Center, 492 Old Connecticut Path, Framingham, MA 01701.

of Transport Jobs. The Material Transport and Storage Component uses services provided by the Machine AIG to access interfaces of Machines and Ports that serve as the source or destination of Transport Jobs. The MES Factory defines areas within a factory which represent groups of machines. Finally, Durables Management provides the services needed to manage the material containers that are moved.

6.1.2 Figure 3 shows the Material Transport and Storage Component information model, including the inherited interfaces and the associations between interfaces. These interfaces are fully defined in Sections 6.3 and 6.4. The interfaces associated with the Material Transport and Storage are as follows:

1. TransportJobSupervisor – abstract interface,
2. MaterialTransportManager,
3. MaterialTransportController, and
4. TransportJob.

6.2 Required Subsets of Future CIM Framework Specifications

6.2.1 This specification depends on interfaces from Durables Management Component, the Machine Abstract Interface Group and the Factory Operations Component that will eventually be contained in future CIM Framework component specifications. They are included with this document to support a complete specification for AMHS support within the CIM Framework. When these items are balloted as separate documents, they will be removed from this specification and referenced in their full form.

6.2.2 Figure 4 illustrates the dependencies of the Material Transport and Storage Component.

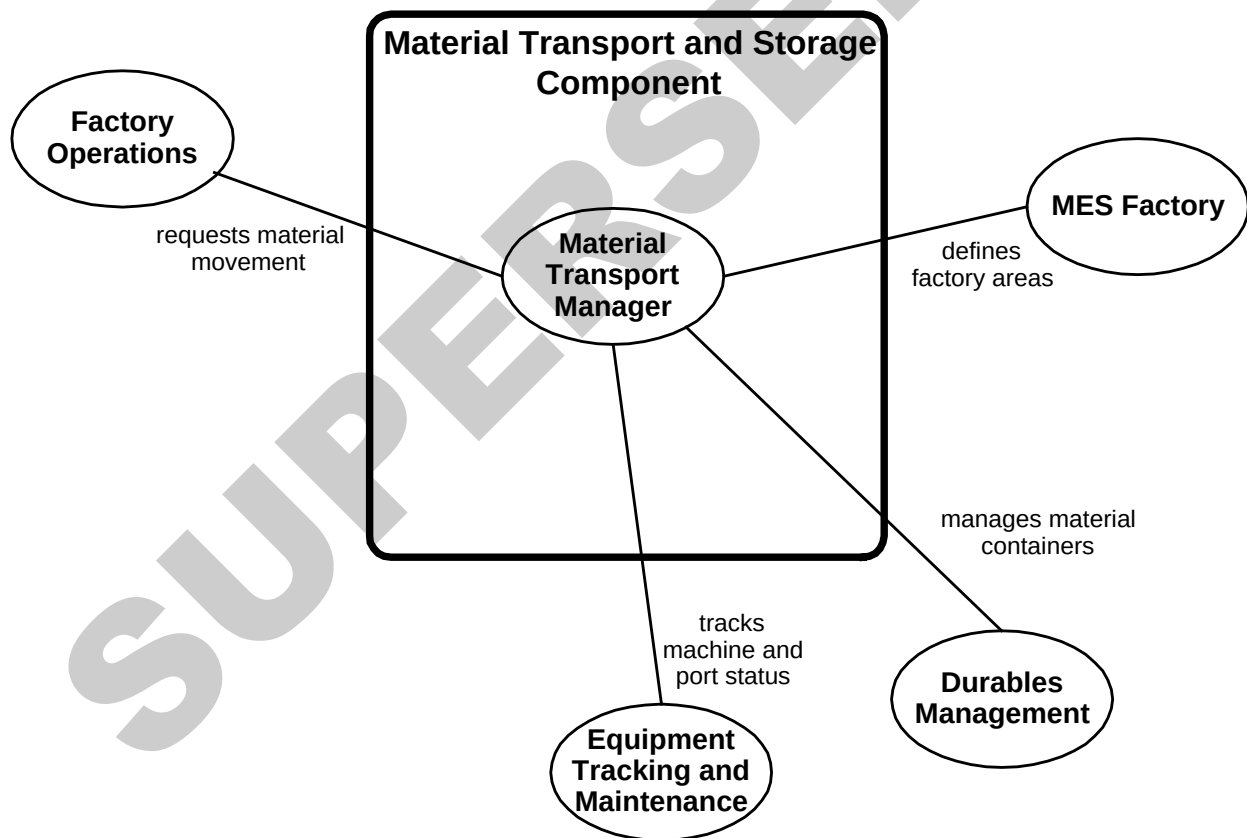


Figure 2
Material Transport and Storage Component Context

Material Transport and Storage Component

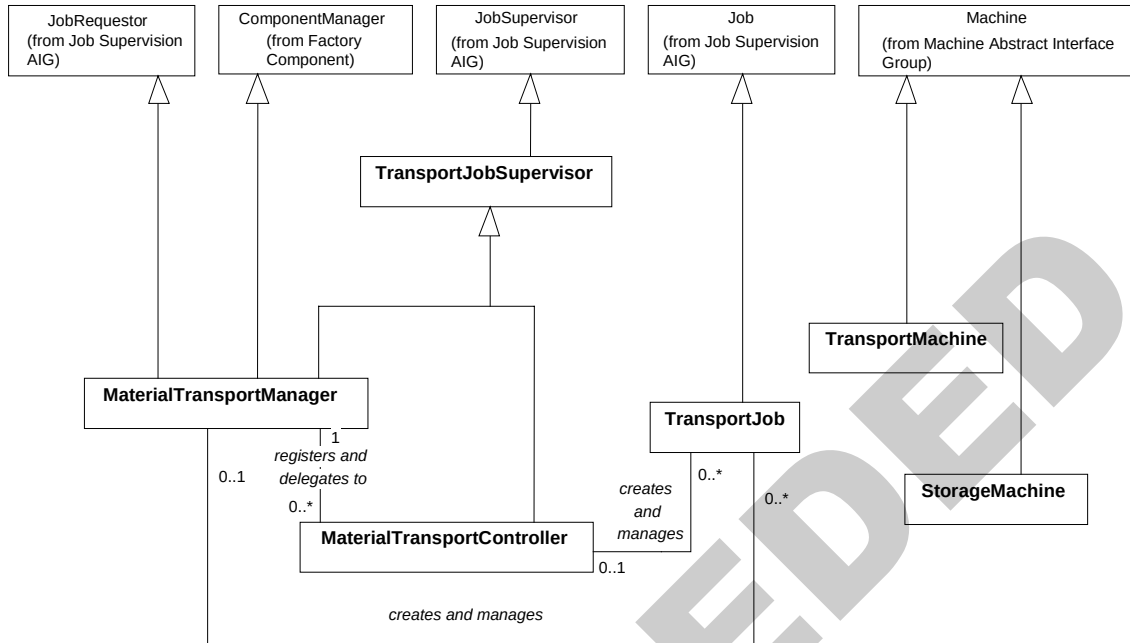


Figure 3
Material Transport and Storage Information Model

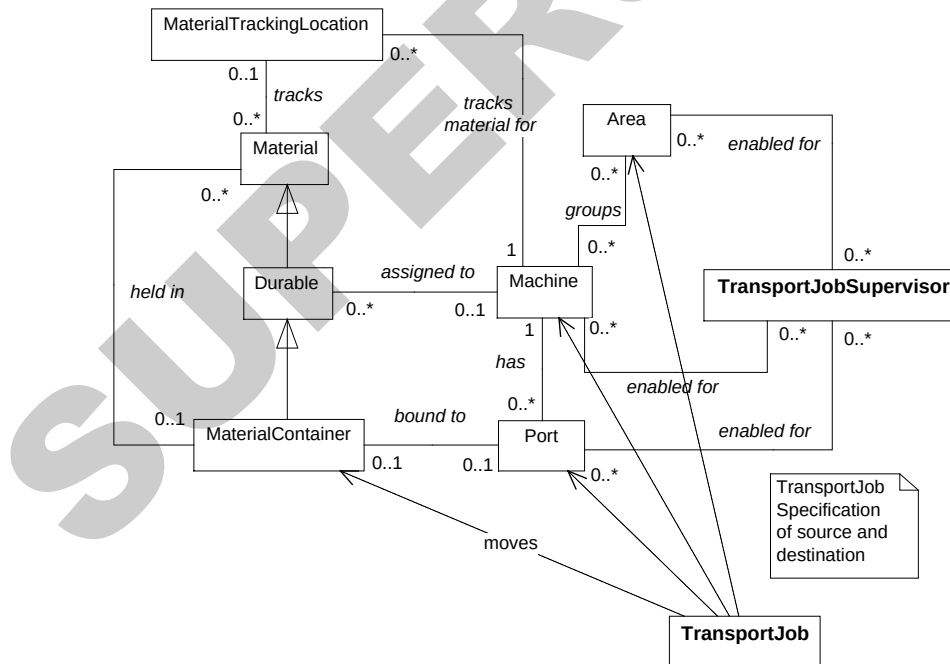


Figure 4
Material Transport and Storage Dependencies

6.2.3 *Durables Management Component (subset)* — This section includes only those interfaces and specific operations from the Durables Management Component that are referenced within this component specification. These interfaces are provided here for the referential integrity of this specification and are not included as part of this component. Further, this is not intended to be a comprehensive treatment of the required interfaces, but only the subset needed here for reference prior to the full specification of Durables Management being adopted through a subsequent ballot. When that occurs, this specification can be updated by a follow-on ballot to reference the interfaces directly rather than replicate them here.

6.2.3.1 *Durable Interface (subset)*

Module: DurablesManagement

Interface: Durable

Inherited Interface: Material

interface **Durable** : AbstractIF::Material {

Description: The Durable interface represents any Material in the Factory used to facilitate manufacturing but not normally consumed in the process. It is capable of relocation within the Factory and requires dynamic tracking. This includes containers used to transport the Material, fixtures (attachments for holding material in a fixed position), and tools (such as reticles, load boards, workholders) used by equipment or personnel in the manufacturing process. Grouping of Durables (such as all reticles that are usable for a given process) can be achieved by creating MaterialGroups for the specific categorizations required.

Exceptions:

Published Events:

/* Notifies subscribers of a change in the MaterialTrackingLocation that is tracking the Durable. */

DurableLocationChangedEvent

Provided Services:

/* Set and get the location of the durable. If the MaterialTrackingLocation is set to NULL, the durable is not in any MaterialTrackingLocation and may be under manual control within the factory. */

```
void setMaterialTrackingLocation (
    in MachineAIG::MaterialTrackingLocation
    aMaterialTrackingLocation)
    raises (Global::FrameworkErrorSignal,
           MachineAIG::MaterialTrackingLocation::MaterialTrackingLocationFullSignal);
```

```
MachineAIG::MaterialTrackingLocation getMaterialTrackingLocation ( )
    raises (Global::FrameworkErrorSignal);
```

/* Set and get the Unit that determines which MaterialTrackingLocation this Durable will be tracked through for a given Machine. The Unit is a string value that maps to the Unit attribute of the MaterialTrackingLocation. */

```
void setUnit(
    in string aUnit)
    raises (Global::FrameworkErrorSignal,
           MachineAIG::MaterialTrackingLocation::InvalidUnitSignal);
```

```
string getUnit( )
    raises (Global::FrameworkErrorSignal);
```

}; // **Durable**

Dynamic Model: None

6.2.3.2 *MaterialContainer Interface (subset)*

Module: DurablesManagement

Interface: MaterialContainer

Inherited Interface: Durable

interface **MaterialContainer** : Durable {

Description: A MaterialContainer interface represents any receptacle for holding Material for transport, processing or storage. Examples of MaterialContainers are shipping boxes, tubes, standard mechanical interface (SMIF) pods, etc.

Exceptions: None.

Published Events:

/* Notifies subscribers of a state change in the MaterialContainer. */

MaterialContainerStateChangeEvent

Provided Services:

/* The following operations are provided to trigger MaterialContainer state transitions. */

```
void makeManualControl ( )
    raises (Global::FrameworkErrorSignal,
           Global::InvalidStateTransitionSignal);
```

```
void makeStored ( )
    raises (Global::FrameworkErrorSignal,
           Global::InvalidStateTransitionSignal);
```

```
void makeInTransit ( )
    raises (Global::FrameworkErrorSignal,
           Global::InvalidStateTransitionSignal);
```

```
void makeProcessing ( )
    raises (Global::FrameworkErrorSignal,
           Global::InvalidStateTransitionSignal);
```

```
}; // MaterialContainer
```

```
typedef sequence <MaterialContainer> MaterialContainerSequence;
```

Dynamic Model:

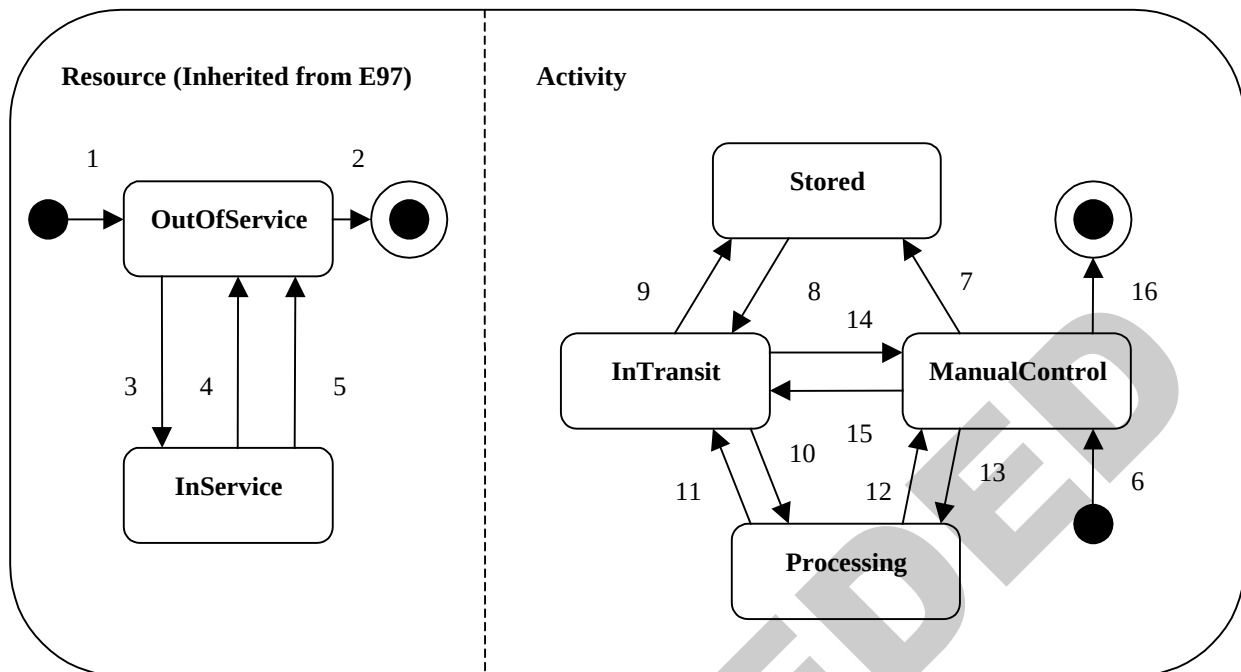


Figure 5
MaterialContainer Dynamic Model

Table 1 MaterialContainer State Definitions

State	State Definition	Example
InService	Inherited from Resource State Model	
OutOfService	Inherited from Resource State Model	
InTransit	Container is en route as the subject of a Transport Job.	Container is on an OHT transport vehicle.
Stored	Container is stored in a Storage Machine and not subject to a Transport Job.	Container is on a stocker shelf.
ManualControl	Container is outside of the control of the Material Transport and Storage Component.	Container is on a PGV.
Processing	Container is at process equipment.	Container is on a lithography tool.

Table 2 MaterialContainer State Transitions

#	Current State	Trigger	Next State	Action	Comment
1–5	NOTE: Transitions 1–5 are defined in the dynamic model inherited from Resource.				
6	N/A	makeManualControl()	ManualControl	MaterialContainer instance is created.	Initial state
7	ManualControl	makeStored()	Stored	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container is placed on input port of a Storage Machine without a TransportJob assigned to it.
8	Stored	makeInTransit()	InTransit	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has started to move. TransportJob state transitions to Executing.
9	InTransit	makeStored()	Stored	DurableLocation ChangedEvent Published	Container stored. TransportJob state

#	Current State	Trigger	Next State	Action	Comment
				by the instance of MaterialContainer	transitions to Completed.
10	InTransit	makeProcessing()	Processing	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has reached destination at process equipment. TransportJob state transitions to Completed.
11	Processing	makeInTransit()	InTransit	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has left process equipment. TransportJob state transitions to Executing.
12	Processing	makeManual Control()	ManualControl	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has left control of process equipment (not via TransportJob).
13	ManualControl	makeProcessing()	Processing	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has entered control of process equipment (not via TransportJob).
14	InTransit	makeManual Control()	ManualControl	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has left the control of the TransportJob via a manual output port. TransportJob state transitions to Aborted.
15	ManualControl	makeInTransit()	InTransit	DurableLocation ChangedEvent Published by the instance of MaterialContainer	Container has entered the AMHS via a manual input port. TransportJob state transitions to Executing.
16	ManualControl	none	N/A	none	MaterialContainer instance has been removed.

6.2.4 Machine Abstract Interface Group (subset)

This section includes only those interfaces and specific operations from the Machine Abstract Interface Group that are referenced within this component specification. These interfaces are provided here for referential integrity of this specification and are not intended to remain as part of this component. Further, this is not intended to be a comprehensive treatment of the required interfaces, but only the subset needed for reference until the full specification is adopted through a subsequent ballot. When that occurs, this specification can be updated to reference the interfaces directly rather than replicate them here.

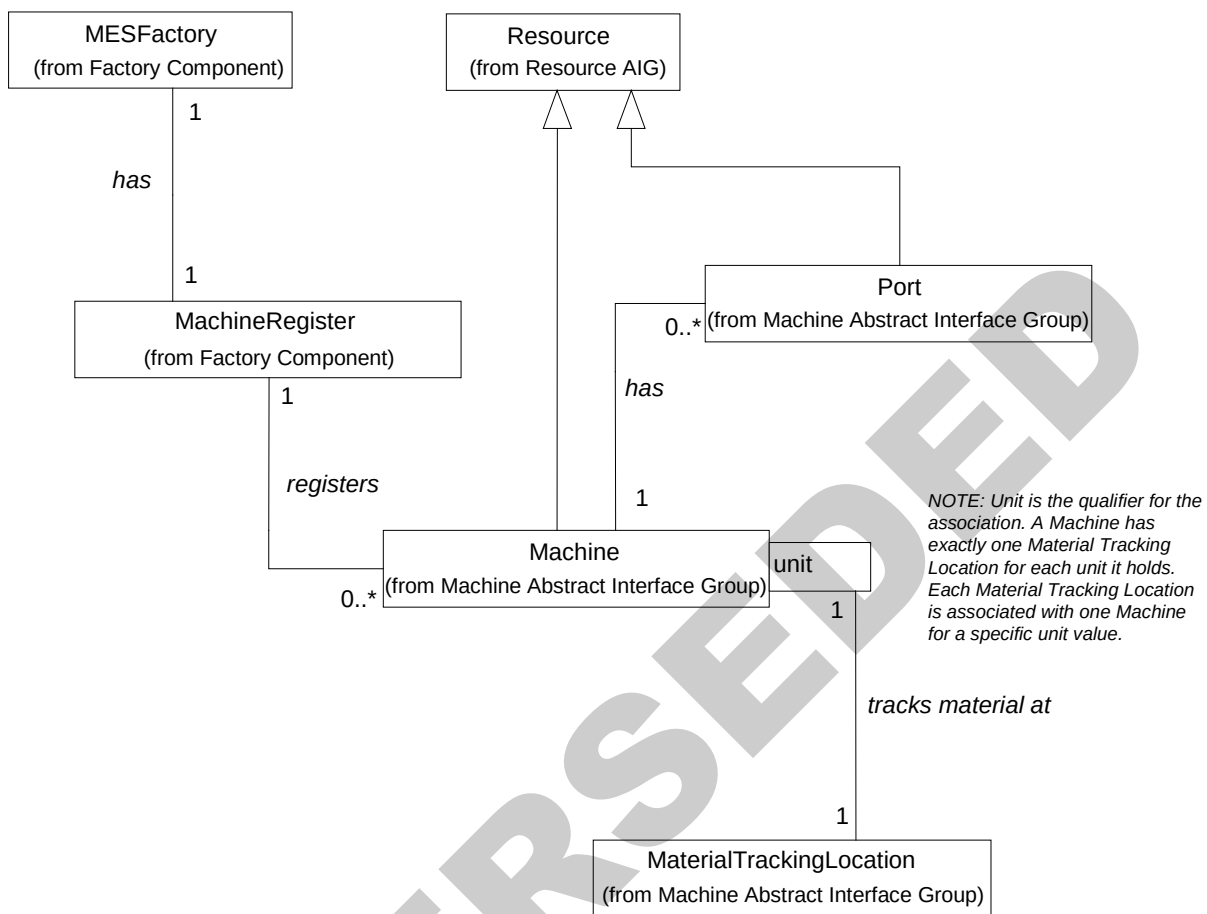


Figure 6
Machine Abstract Interface Group (subset) Information Model

6.2.4.1 Machine Interface (subset)

Module: MachineAIG

Interface: Machine

Inherited Interface: Resource

```
interface Machine : AbstractIF::Resource {
```

Description: This abstract interface representing the concept of a piece of equipment in the factory. A Machine establishes the identity of the physical equipment existing in the factory for reference within an MES context. The Machine also maintains one MaterialTrackingLocation for each Unit defined to classify the Material processed, used, or stored by the Machine.

Type Definitions:

```
typedef sequence <Machine> MachineSequence;
```

```
typedef sequence <MaterialTrackingLocation> MaterialTrackingLocationSequence;
```

```

typedef sequence <Port> PortSequence;

enum MachineType {
    FixedBufferProductionMachine,
    InternalBufferProductionMachine,
    StorageMachine,
    TransportMachine };

Exceptions:

/* Port indicated for receiving material was invalid. */

exception InvalidPortSignal { };

/* Material specified for unloading was invalid. */

exception InvalidMaterialSignal { };

Published Events:

/* An E10 State change has occurred in the Machine that the Factory needs to know. */

MachineResourceE10StateChangedEvent

Provided Services:

/* Set the MachineType for this machine. */

void setMachineType (
    in MachineType aMachineType )
    raises (Global::FrameworkErrorSignal);

/* Return the MachineType for this machine. */

MachineType getMachineType ( )
    raises (Global::FrameworkErrorSignal);

/* Set up a MaterialTrackingLocation for this Machine to track a particular material Unit characterizing its contents.
A Machine may have more than one MaterialTrackingLocation as long as each is tracking a different Unit. The
value of Unit is used with corresponding attributes of the material to allocate the tracked material to the correct
MaterialTrackingLocation for a Machine. */

void setMaterialTrackingLocation (
    in MaterialTrackingLocation aMaterialTrackingLocation,
    in string aUnitToTrack)
    raises (Global::FrameworkErrorSignal);

/* Get all MaterialTrackingLocations for this Machine. There is one and only one MaterialTrackingLocation per
Unit used in the Machine. */

MaterialTrackingLocationSequence getAllMaterialTrackingLocations ( )
    raises (Global::FrameworkErrorSignal);

/* Get the MaterialTrackingLocation for the Unit specified. */

MaterialTrackingLocation getMaterialTrackingLocation (
    in string aUnit)
    raises (Global::FrameworkErrorSignal);

```

/* Request a Machine to reserve a port to load material. This may involve communication between the Port and the Machine Objects, but these need not be public (or may involve communication between the Machine and the Equipment). If requestedPort parameter left null, machine chooses the requestedPort. Returns Null if Machine cannot or refuses to comply. */

```
Port reservePortForTransferTo (
    in DurablesManagement::MaterialContainer aContainer,
    in Port requestedPort )
    raises (Global::FrameworkErrorSignal,
           InvalidPortSignal);
```

/* Request a Machine to reserve a port to load two or more MaterialContainers. All MaterialContainers in the sequence must be loaded before the Machine can accept another port reservation request. The MaterialContainers specified in the MaterialContainerSequence may be loaded in any order. If requestedPort parameter left null, machine chooses the requestedPort. Returns Null if Machine cannot or refuses to comply. */

```
Port reservePortForBatchTransferTo (
    in DurablesManagement::MaterialContainerSequence aContainerGroup,
    in Port requestedPort )
    raises (Global::FrameworkErrorSignal,
           InvalidPortSignal);
```

/* Request a Machine to cancel a reservation previously established for a transfer to a port. */

```
void cancelPortReservationForTransferTo (
    in DurablesManagement::MaterialContainer aContainer,
    in Port requestedPort )
    raises (Global::FrameworkErrorSignal,
           InvalidPortSignal);
```

/* Inform a Machine that a transfer to or from a port has been completed. This may involve communication between the Port and the Machine Objects, but these need not be public (or may involve communication between the Machine and the Equipment). */

```
void transferComplete (in DurablesManagement::MaterialContainer aContainer,
    in Port reservedPort )
    raises (Global::FrameworkErrorSignal,
           InvalidPortSignal);
```

/* Request a Machine to prepare to unload material. Returns the Port on which the material will be unloaded or Null if the Machine cannot or refuses to comply. */

```
Port containerOut (
    in DurablesManagement::MaterialContainer aContainer,
    in Port requestedPort )
    raises (Global::FrameworkErrorSignal,
           InvalidMaterialSignal);
```

/* Returns a list of all Ports for the machine. */

```
PortSequence allPorts ( )
    raises (Global::FrameworkErrorSignal);
```

/* Returns the list of all Areas that this Machine is in. */

```
AreaSequence allAreas( )
    raises (Global::FrameworkErrorSignal);
```

```
}; // Machine
```

6.2.4.2 *MaterialTrackingLocation Interface*

Module: MachineAIG
Interface: MaterialTrackingLocation
Inherited Interface: OwnedEntity

```
interface MaterialTrackingLocation : AbstractIF::OwnedEntity {
```

Description: A MaterialTrackingLocation is a place where Material may be held. The “type” of the location is based on the classification Unit of Material that the location can hold (e.g., locations to hold wafers have a “Wafer” Unit and can only hold this Material type). Each machine has one MaterialTrackingLocation for each material Unit. The MaterialTrackingLocation also maintains a capacity.

Exceptions:

```
/* Material specified in the call is not a valid Material for this MaterialTrackingLocation. */  
exception InvalidMaterialSignal { };  
/* Intent to assign more Material to the MaterialTrackingLocation failed because it is full. */  
exception MaterialTrackingLocationFullSignal { };  
/* Unit value is not valid for the MaterialTrackingLocation. */  
exception InvalidUnitSignal { };
```

Published Events:

```
/* A Material Tracking Location has reached its capacity. */  
MaterialTrackingLocationFullEvent  
/* A Material Tracking Location that was previously Full now has available capacity. */  
MaterialTrackingLocationNotFullEvent
```

Provided Services:

```
/* Return the Material at the MaterialTrackingLocation, return nil if the location is empty. */  
  
AbstractIF::MaterialSequence allMaterialHeld ( )  
    raises (Global::FrameworkErrorSignal);  
/* Get and set the unique identifier for the MaterialTrackingLocation. */  
  
string getIdentifier ( )  
    raises (Global::FrameworkErrorSignal);  
  
void setIdentifier (  
    in string identifier)  
    raises (Global::FrameworkErrorSignal,  
           Global::SetValueOutOfRangeSignal,  
           Global::DuplicateIdentifierSignal);  
  
/* Return the Machine that holds this MaterialTrackingLocation. */  
  
Machine getMachine ( )  
    raises (Global::FrameworkErrorSignal);
```

/* Set the material Unit that can be held in the MaterialTrackingLocation. Example: If the MaterialTrackingLocation is to hold wafers, the Unit could be “wafer”. If it was a particular size of Wafer the Unit could be “200mmWafer.” */

```
void setUnit (
    in string aUnit)
    raises (Global::FrameworkErrorSignal,
           InvalidUnitSignal);
```

/* Get the Unit for the MaterialTrackingLocation. */

```
string getUnit ( )
    raises (Global::FrameworkErrorSignal);
```

/* Add a Material to the MaterialTrackingLocation. */

```
void addMaterialToTrack (
    in AbstractIF::Material aMaterial )
    raises (Global::FrameworkErrorSignal,
           InvalidMaterialSignal,
           MaterialTrackingLocationFullSignal);
```

/* Remove a Material from the MaterialTrackingLocation. */

```
void removeMaterial (
    in AbstractIF::Material aMaterial )
    raises (Global::FrameworkErrorSignal,
           InvalidMaterialSignal);
```

/* Reserve capacity of the MaterialTrackingLocation. If accepted, this portion of the MaterialTrackingLocation’s capacity will be kept for the designated material. If the capacity is reserved the operation returns True. */

```
boolean reserveCapacity (
    in AbstractIF::Material capacityToReserve)
    raises (InvalidUnitSignal,
           InvalidMaterialSignal,
           MaterialTrackingLocationFullSignal,
           Global::FrameworkErrorSignal);
```

/* Release reserved capacity of the MaterialTrackingLocation. If accepted, this portion of the MaterialTrackingLocation’s capacity will be released for use by other material. */

```
boolean releaseCapacity (
    in AbstractIF::Material capacityToRelease)
    raises (Global::FrameworkErrorSignal);
```

/* Get and set the Maximum Capacity for the MaterialTrackingLocation. */

```
void setMaximumCapacity (
    in long maximum)
    raises (Global::FrameworkErrorSignal,
           Global::SetValueOutOfRangeSignal);
```

```
long getMaximumCapacity ( )
    raises (Global::FrameworkErrorSignal);
```

/* Get the Available Capacity for the MaterialTrackingLocation. Available Capacity is the difference between Maximum Capacity and the count of the sequence returned by AllMaterialHeld. */

```
long getAvailableCapacity ( )
    raises (Global::FrameworkErrorSignal);
```

/* Get the Reserved Capacity for the MaterialTrackingLocation. Reserved Capacity is the portion of the AvailableCapacity that has been set aside for known material that has not yet arrived. */

```
long getReservedCapacity ( )
    raises (Global::FrameworkErrorSignal);
```

/* This is a query for the availability status of the MaterialTrackingLocation. Available means the MaterialTrackingLocation has available capacity in which more material could go. */

```
boolean hasAvailableCapacity ( )
    raises (Global::FrameworkErrorSignal);
```

```
}; // MaterialTrackingLocation
```

6.2.4.3 Port Interface (subset)

Module: MachineAIG

Interface: Port

Inherited Interface: Resource

```
interface Port : AbstractIF::Resource {
```

Description: The Port represents the point at which a “change of ownership” occurs during a material transfer. Each Port has at least one associated Machine and the Port may be thought of as an “access point” to the Machine.

Exceptions: None.

Events: None.

Provided Services:

/* Returns the machine to which this port belongs. */

```
Machine getMachine ( )
    raises (Global::FrameworkErrorSignal);
```

```
}; // Port
```

6.2.5 *Factory Operations Component (subset)* — This section includes only those interfaces and specific operations from the Factory Component that are referenced within this component specification. These interfaces are provided here for referential integrity of this specification and are not intended to remain as part of this component. Further, this is not intended to be a comprehensive treatment of the required interfaces, but only the subset needed for reference until the full specification is adopted through a subsequent ballot for the full Factory Operations component. When that occurs, this specification can be updated to reference the interfaces directly rather than replicate them here.

6.2.5.1 MachineRegister Interface

Module: FactoryOperations

Interface: MachineRegister

Inherited Interface: Resource

```
interface MachineRegister : AbstractIF::Resource {
```

Description: The MachineRegister maintains a list of known machines for the factory and supplies related information on demand. There is only one MachineRegister per factory. NOTE: this interface may eventually be replaced by the trader service.

Exceptions:

```

/* An attempt was made to locate an unknown Machine. */

exception MachineNotFoundSignal {string machineName;};

exception MachineRemovalFailedSignal { };

Published Events:

/* Event indicating that a Machine has been added or removed from the Factory. */

MachineListChangedEvent

Provided Services:

/* Add a Machine to the set of machine(s) managed by the MachineRegister. */

void addMachine (in Machine aMachine)
    raises (Global::FrameworkErrorSignal);

/* Remove a Machine from the set of machine(s) managed by the MachineRegister. */

void removeMachine (in Machine aMachine)
    raises (Global::FrameworkErrorSignal,
           MachineRemovalFailedSignal);

/* Return a sequence of all machines managed by the MachineRegister. */

MachineSequence allMachines ( )
    raises (Global::FrameworkErrorSignal);

/* Return the Machine corresponding to the given name. */

Machine findMachineNamed (in string identifier)
    raises (Global::FrameworkErrorSignal,
           MachineNotFoundSignal) ;

Contracted Services:      None.
Dynamic Model:            None.

}; // MachineRegister

```

6.3 Material Transport Abstract Interfaces — The Material Transport Abstract Interfaces are defined to perform material transport at the factory level. The TransportJobSupervisor interface must be specialized by the users of the Material Transport Manager Interface. These interfaces provide a way to track the job through the factory.

6.3.1 TransportJobSupervisor Interface

Module: MaterialTransport
Interface: TransportJobSupervisor
Inherited Interface: JobSupervisor

```
interface TransportJobSupervisor : AbstractIF::JobSupervisor {
```

Description: TransportJobSupervisor is an abstract interface which provides the operations needed by clients to request, track, and control material transport within the domain of the specific TransportJobSupervisor. Any entity which is responsible for moving material will inherit and implement TransportJobSupervisor.

Type Definitions:

```
typedef sequence <Global::Properties> JobSpecSequence;
```

Exceptions:

exception **UnsupportedDestinationTypeSignal** { };

Published Events: None.

Provided Services:

The Creation of TransportJobs is accomplished by use of the requestJob operation from the inherited JobSupervisor interface. The “JobType” property specified in the inherited interface should be set to “TransportJobType” for all TransportJob requests. The additional Job Specification properties required for the creation of Transport jobs are defined in Table 3. The types of machines specified in the TransportJob are AMHS Storage Machines, Fixed Buffer Production Equipment, or Internal Buffer Production Equipment. Each machine type represents a separate case used in defining the Job Specification for a Transport Job, and these cases are also defined in Table 3.

For transportJobs, the inherited ABORTING and ABORTED states may not always result in an immediate termination of the job and associated activities. For example, transportJobs may involve autonomous transport equipment that operate without persistent communication links to the TransportJob and TransportJobSupervisor objects. In these cases there may be a time delay before the results of a transportJob makeAborted request can be enacted to allow the transport equipment to reach a known location. In other cases a movement activity must reach a stable condition before it can be interrupted. The location of the MaterialContainer associated with the aborted TransportJob is the nearest location where movement can be safely and reliably interrupted.

By contrast, if a transportJob is transitioned to the inherited STOPPING state, the location of the MaterialContainer associated with the stopped TransportJob must be a legal Source for issuing a new TransportJob.

Table 3 Additional Required Properties for Transport Jobs

<i>Name</i>	<i>Value Type</i>	<i>Description</i>
“Container”	MaterialContainer	Container being transported.
“SourceMachine”	Machine	The source production machine for the transport job. AMHS Storage Equipment: SourceMachine is Optional. Fixed Buffer Production Equipment: SourceMachine is Required. Internal Buffer Production Equipment: SourceMachine is Required.
“SourcePort”	Port	The source production machine port for the transport job. AMHS Storage Equipment: SourcePort is Optional. Fixed Buffer Production Equipment: SourcePort is Required. Internal Buffer Production Equipment: SourcePort is optional. Port may be obtained dynamically with the containerOut operation or it may be predefined in the TransportJob specification.
“DestinationArea”	Area	The destination area for the transport job. DestinationArea shall contain only StorageMachines. The implementation shall determine the specific machine to which the MaterialContainer is delivered. AMHS Storage Equipment: DestinationArea is optional. Fixed Buffer Production Equipment: DestinationArea is not allowed. Internal Buffer Production Equipment: DestinationArea is not allowed.
“DestinationMachine”	Machine	The destination machine for the transport job. AMHS Storage Equipment: DestinationMachine is required if DestinationArea is not specified. Fixed Buffer Production Equipment: DestinationMachine is required. Internal Buffer Production Equipment: DestinationMachine is required.

Name	Value Type	Description
"DestinationPort"	Port	The destination port for the transport job. AMHS Storage Equipment: DestinationPort is required for Manual output port destinations. Fixed Buffer Production Equipment: DestinationPort is required. Internal Buffer Production Equipment: DestinationPort is NULL.

/* The requestBatchTransportJob operation will provide two or more job specifications for transport jobs that share a dependency on completion. All job specifications must have the same priority. In the normal case, the TransportJobSupervisor will create simple transport jobs for each job specification and return the jobs as a sequence. Subsequently, if one of the simple jobs cannot complete, the other subordinate jobs must be reassessed by the JobRequestor to determine the appropriate corrective action. In this case, the TransportJobSupervisor will notify the JobRequestor of the job failure with the informJobTerminated message and pause the other jobs of the batch-job to allow the JobRequestor to determine its response. Implementations may implement the TransportJobSupervisor with more complex default actions in response to a partial failure of a batch-transport-job, but these more complex scenarios are not part of the standard behavior for this operation. */

```
AbstractIF::JobSequence requestBatchTransportJob (
    in JobSpecSequence jobSpecs,
    in AbstractIF::JobRequestor aJobRequestor)
    raises (Global::FrameworkErrorSignal,
           AbstractIF::JobSupervisor::JobRejectedSignal);
```

Contracted Services: None

Dynamic Model: None

```
}; // TransportJobSupervisor
```

6.4 Material Transport and Storage Component

6.4.1 This component is concerned only with factory-wide material transport and not the movement of material within a piece of equipment. Also note that material can be either product, durables, or consumables. The CIM Framework recognizes the potential for multiple AMHS's within a single factory (e.g., one supplier of interbay movement and another supplier of intrabay movement). The MES, however, wants to only have to issue a move request ("move this material from here to there") without regard to whether that transport spans one or more AMHS's. Therefore, the component supporting material transport requires a single entry point for initiating material transport which will be through the Material Transport Manager (MTM).

6.4.2 The Material Transport Manager must be able manage transport jobs carried out by many different AMHS's. To accomplish this, a separate interface for a Material Transport Controller (MTC) is specified. In the simplest implementations, a single MTM may manage all TransportJobs directly. In this case no separate Material Transport Controller implementation is needed. If there is a need for delegation to one or more Material Transport Controllers, the Material Transport Manager can register any conformant MTC implementation which supports the specified interface. The low-level interaction of the material movement controllers with the physical equipment (transport machines, storage machines and ports) is encapsulated within the scope of this component. Note that the component manager does not provide lifecycle services for its managed material movement controllers, but instead uses the registration interfaces.

6.4.3 MaterialTransportManager Interface

Module: MaterialTransport

Interface: MaterialTransportManager

Inherited Interface: ComponentManager, TransportJobSupervisor, JobRequestor

```
interface MaterialTransportManager : FactoryOperations::ComponentManager,
    TransportJobSupervisor, AbstractIF::JobRequestor {
```