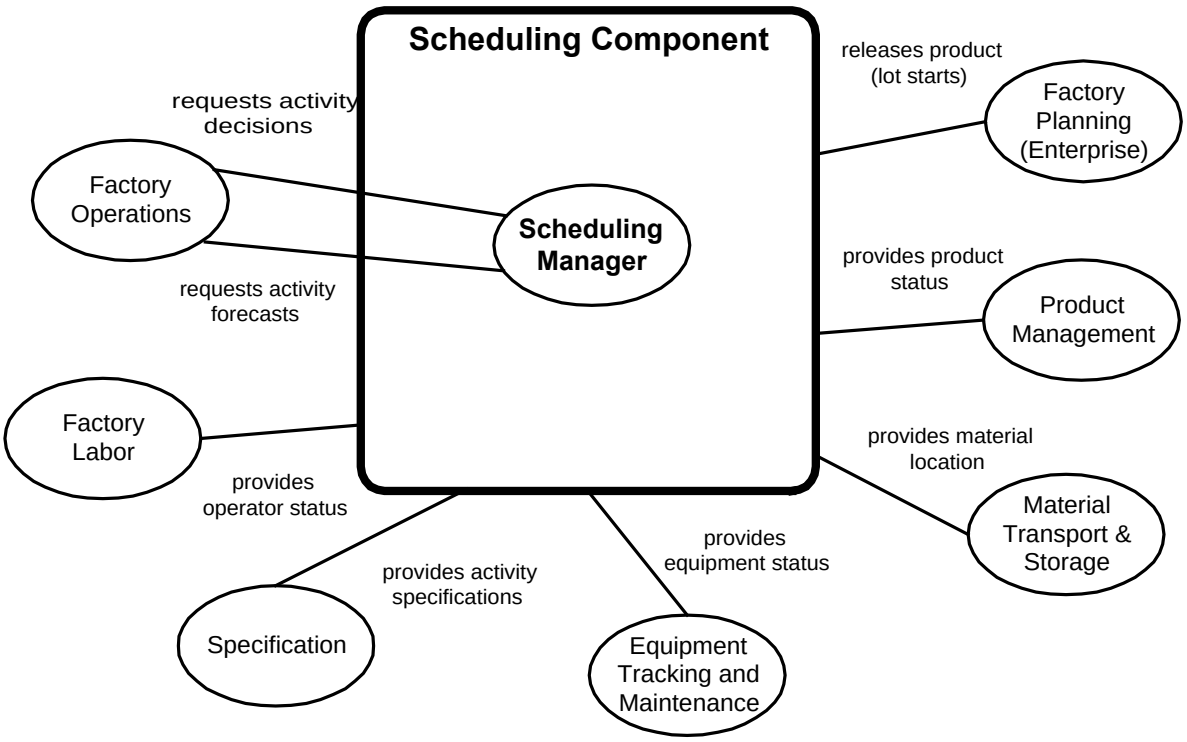


Figure 1
Scheduling Component Interactions

2.6 This specification does not describe the definition of the policies used by the Scheduling Component to produce activity lists. This is left to component implementors.

2.7 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 limitations prior to use.

3 Limitations

3.1 *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.

3.1.1 *Dependence on Factory Operations Component* — The Scheduling Component provides specifications of activities that are executed by the Factory Operations Component in the form of production jobs, transport jobs and maintenance jobs. The specification of Factory Operations may require changes to the activity specification properties of the Scheduling Component. When the Specification for CIM Framework Factory Operations Component is complete, this dependency can be resolved.

3.1.2 *Specification of Event Subscriptions* — The Scheduling Component maintains a current status of the factory resources and material by receiving event notifications of occurrences that indicate changes in other components. Events are specified with the component responsible for publishing the event notifications. Until all of the required event formats are defined as part of the publishing component specification, the inputs to the Scheduling Component will be defined only in general terms. When event specifications are complete, these event subscriptions can be specified.

3.1.3 *Initialization* — The Scheduling Component does not currently provide interfaces that allow the representation of factory conditions to be initialized for the start-up of dispatching and scheduling. The addition of initialization interfaces will be added or deemed out of scope prior to upgrade to full standard status.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E81 — Provisional Specification for CIM Framework Domain Architecture

SEMI E97 — Provisional Specification for CIM Framework Global Declarations and Abstract Interfaces

SEMI E102 — Provisional Specification for CIM Framework Material Transport and Storage Component

4.2 *Other Documents*

UML Notation Guide, Version 1.1, document number ad/97-08-05, Object Management Group.[1]

ISO/IEC International Standard 14750 (also ITU-T Recommendation X.920): Information Technology – Open Distributed Processing – Interface Definition Language [2]

5 Terminology

5.1 *Definitions*

5.1.1 *activity* — work performed as part of the manufacturing operations of a factory. Activities may be specified formally by a predefined type of job specification (for example, Production Job, Transport Job or PM Job), or they may be represented by identifying the minimal set of resources and material needed to allow subsequent completion of the job specification. An activity is the result of dispatching or scheduling.

5.1.2 *dispatching* — generation of a decision or option for the next activity involving a particular factory resource or material. The dispatch result is determined by evaluating the current state of the factory, the priorities and requirements for the activities, and the relationship of the activities to one another. Dispatching returns only the immediately applicable part of a schedule.

5.1.3 *factory planning* — recommendation of lot starts for a particular production facility over an extended period of time. The factory plan is determined by predicting future changes in factory state and available capacity as lots progress through production. This prediction is used to determine the optimum sequence of lot starts to best achieve the production goals of the facility. Factory planning is typically the responsibility of enterprise systems.

5.1.4 *scheduling* — generation of a forecast of future time sequenced activities involving factory resources or material. The schedule is based on the current state of the factory, the priorities and requirements for the activities, the relationship of the activities to one another and knowledge of factory level goals and capacity. Scheduling covers activities projected to occur over a longer future time interval than dispatching.

6 Requirements

6.1 *Scheduling Component* — The Scheduling Component provides a single interface called the SchedulingManager with operations for both dispatching and scheduling.

6.1.1 Any resource or material within the factory may have a next activity defined for it. The SchedulingManager interface is responsible for identifying activities that answer questions such as:

- What's next for this Machine?
- Where next for this Lot?
- What's next for this Operator?

6.1.2 The SchedulingManager interface is also responsible for generating a time-sequenced forecast of future activities for a resource or for material within the factory. This forecast is based on projections from the current factory state and may be updated as actual activities are dispatched and executed.

6.1.3 The Scheduling Component Information Model is shown in Figure 2.

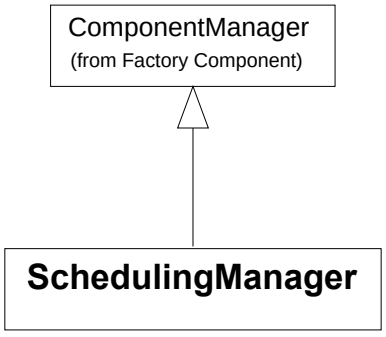


Figure 2
Dispatching Component Information Model

1 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.

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

6.1.4 *Scheduling Component Declarations*

/* The ActivityType enumerated value will determine which kind of activity is represented and the appropriate properties present in the activitySpec. */

```
enum ActivityType {
      ProductionActivity,
      TransportActivity,
      MaintenanceActivity,
      DoNothing };
```

/* ActivityOption represents a single activity that is eligible to be dispatched for execution. The prioritySequenceNumber represents the priority ordering of a list of ActivityOptions in the case where a manual process includes selection of one ActivityOption from the list. The deadline is a timestamp that is used to convey start time constraints for activities. It may also be used in a DoNothing activity to specify the timeout for a retry of the dispatch request. */

```
struct ActivityOption {
      ActivityType activityType;
      short prioritySequenceNumber;
      Global::TimeStamp deadline;
      AbstractIF::ResourceSequence resources;
      AbstractIF::MaterialGroupSequence materialGroups;
      DurablesManagement::DurableSequence durables;
      Global::Properties activitySpec;
}; // ActivityOption
```

/* ActivityForecast represents a single Activity that may be dispatched and executed in the future. An ActivityForecast includes information that is equivalent to an ActivityOption, but with added start and finish time projections. The timeSequenceNumber represents the time ordering of activities consistent with the projectedStart and projectedFinish. */

```
struct ActivityForecast {
      ActivityType activityType;
      short timeSequenceNumber;
      AbstractIF::ResourceSequence resources;
      AbstractIF::MaterialGroupSequence materialGroups;
      DurablesManagement::DurableSequence durables;
      Global::TimeStamp projectedStart;
      Global::TimeStamp projectedFinish;
      Global::Properties activitySpec;
}; // ActivityForecast
```

/* An ActivityList contains one or more ActivityOptions for a Resource, MaterialGroup or Durable which are all eligible for immediate execution. A list of one ActivityOption represents a decision from the SchedulingComponent. A list of more than one ActivityOption represents recommendations from the SchedulingComponent which require some external interaction for a decision to be reached. */

```
typedef sequence <ActivityOption> ActivityList;
```

/* A ForecastList contains one or more ActivityForecasts for a Resource or MaterialGroup which are not yet dispatched for execution. */

```
typedef sequence <ActivityForecast> ForecastList;
```

6.1.5 *Use of ActivitySpec Properties*— The ActivitySpec part of the ActivityOption and ActivityForecast are provided to allow implementations to extend the specifications of activities for specific implementations. The names and value types of properties included in these ActivitySpec structure are not part of the standard.

6.1.6 *SchedulingManager Interface*

Module: Scheduling

Interface: SchedulingManager

Inherited Interface: ComponentManager

```
interface SchedulingManager : FactoryOperations::ComponentManager {
```

Description: The SchedulingManager provides the concrete interfaces that support the registration and control of the Scheduling Component and enable access to the Dispatcher and Scheduler interfaces.

Exceptions:

/* An attempt was made to request an activity for a resource that is not found within the factory data available to this SchedulingManager. */

```
exception ResourceNotFound {
     AbstractIF::Resource requestedObject;};
```

/* An attempt was made to request an activity for a material group that is not found within the factory data available to this SchedulingManager. */

```
exception MaterialGroupNotFound {
     AbstractIF::MaterialGroup requestedObject;};
```

/* An attempt was made to request an activity for a durable that is not found within the factory data available to this SchedulingManager. */

```
exception DurableNotFound {
     DurablesManagement::Durable requestedObject;};
```

Published Events:.

/* The ActivityListAvailable event is published by the Scheduling Component to notify event subscribers that a new ActivityList containing changes is available. The generation of ActivityListAvailable events is controlled as an implementation specific configuration choice. */

`ActivityListAvailable`

/* The ForecastListAvailable event is published by the Scheduling Component to notify event subscribers that a new Forecast is available. ForecastList generation is potentially a compute-intensive operation which may benefit from event notification when an updated forecast is available. The generation of ForecastListAvailable events is controlled as an implementation specific configuration choice. */

`ForecastListAvailable`

Provided Services:

/* Return an activity list for this resource. The activity list will contain the number of existing activities requested in listCount in maximum. If “listCount” is blank, the activity list shall contain all existing activities. A decision will be represented by an activity list with only one activity option. The ActivityOptions returned by this operation may be ProductionActivity, MaintenanceActivity or DoNothing ActivityTypes. Although Resources are typically immobile (such a Equipment), some Resources may also be subject to TransportActivities. */

```
ActivityList whatNextForResourceByCount (
       in AbstractIF::Resource aResource)
       in short listCount)
       raises (Global::FrameworkErrorSignal,
                    ResourceNotFound);
```

/* Return an activity list for this material group. The activity list will contain the number of existing activities requested in listCount in maximum. If "listCount" is blank, the activity list shall contain all existing activities. A decision will be represented by an activity list with only one activity option. The ActivityOptions returned by this operation may be ProductionActivity, TransportActivity or DoNothing ActivityTypes. */

```
ActivityList whatNextForMaterialGroupByCount (
     in AbstractIF::MaterialGroup aMaterialGroup,
     in short listCount)
     raises (Global::FrameworkErrorSignal,
          MaterialGroupNotFound);
```

/* Return an activity list for this durable. The activity list will contain the number of existing activities requested in listCount in maximum. If "listCount" is blank, the activity list shall contain all existing activities. A decision will be represented by an activity list with only one activity option. The ActivityOptions returned by this operation may be ProductionActivity, MaintenanceActivity, TransportActivity or DoNothing ActivityTypes. */

```
ActivityList whatNextForDurableByCount (
      in DurablesManagement::Durable aDurable)
      in short listCount)
      raises (Global::FrameworkErrorSignal,
            DurableNotFound);
```

/* Return an activity list containing the next TransportActivity for this material group. A decision will be represented by an activity list with only one activity option. The ActivityOptions returned by this operation may be TransportActivity or DoNothing ActivityTypes. This operation is defined to support a factory operating with a "push" model that pushes material to the next storage or production machine. */

```
ActivityList whereNextForMaterialGroup (in AbstractIF::MaterialGroup aMaterialGroup)
     raises (Global::FrameworkErrorSignal,
          MaterialGroupNotFound);
```

/* Return an activity list containing the next TransportActivity for this durable. A decision will be represented by an activity list with only one activity option. The ActivityOptions returned by this operation may be TransportActivity or DoNothing ActivityTypes. This operation is defined to support a factory operating with a "push" model that pushes material to the next storage or production machine. */

```
ActivityList whereNextForDurable (in DurablesManagement::Durable aDurable)
     raises (Global::FrameworkErrorSignal,
          DurableNotFound);
```

/* Return a forecast list containing future activities for this resource. The forecast list will contain all activities projected to start before the lookAheadTime, but which have not yet been started. The forecast list may include a mix of different activity types. */

```
ForecastList forecastForResourceByTime (
     in AbstractIF::Resource aResource,
     in Global::TimeStamp lookAheadTime)
     raises (Global::FrameworkErrorSignal,
          ResourceNotFound);
```

/* Return a forecast list containing future activities for this material group. The forecast list will contain all activities projected to start before the lookAheadTime, but which have not yet been started. The forecast list may include a mix of different activity types. */

```
ForecastList forecastForMaterialGroupByTime (
     in AbstractIF::MaterialGroup aMaterialGroup,
     in Global::TimeStamp lookAheadTime)
     raises (Global::FrameworkErrorSignal,
          MaterialGroupNotFound);
```

/* Return a forecast list containing future activities for this durable. The forecast list will contain all activities projected to start before the lookAheadTime, but which have not yet been started. The forecast list may include a mix of different activity types. */

```
ForecastList forecastForDurableByTime (
     in DurablesManagement::Durable aDurable,
     in Global::TimeStamp lookAheadTime)
     raises (Global::FrameworkErrorSignal,
          DurableNotFound);
```

/* Return a forecast list containing future activities for this resource. The forecast list will contain the number of upcoming activities requested in lookAheadCount. The forecast list may include a mix of different activity types. */

```
ForecastList forecastForResourceByCount (
     in AbstractIF::Resource aResource,
     in short lookAheadCount)
     raises (Global::FrameworkErrorSignal,
          ResourceNotFound);
```

/* Return a forecast list containing future activities for this material group. The forecast list will contain the number of upcoming activities requested in lookAheadCount. The forecast list may include a mix of different activity types. */

```
ForecastList forecastForMaterialGroupByCount (
     in AbstractIF::MaterialGroup aMaterialGroup,
     in short lookAheadCount)
     raises (Global::FrameworkErrorSignal,
          MaterialGroupNotFound);
```

/* Return a forecast list containing future activities for this durable. The forecast list will contain the number of upcoming activities requested in lookAheadCount. The forecast list may include a mix of different activity types. */

```
ForecastList forecastForDurableByCount (
     in DurablesManagement::Durable aDurable,
     in short lookAheadCount)
     raises (Global::FrameworkErrorSignal,
          DurableNotFound);
```

Contracted Services: None.

Dynamic Model: Inherited from ComponentManager.

```
}; // SchedulingManager
```

APPENDIX 1
COMPLETE LISTING OF SCHEDULING COMPONENT IDL

NOTE: The material in this appendix is an official part of SEMI E105 and was approved by full letter ballot procedures on August 28, 2000 by North American Regional Standards Committee.

```
module CIMFW {

#include <Global.idl>
#include <AbstractIF.idl>
#include <FactoryOperations.idl>
#include <DurablesManagement.idl>
#ifndef _CIMFW_SCHEDULING_
#define _CIMFW_SCHEDULING_

module Scheduling {

     enum ActivityType {
          ProductionActivity,
          TransportActivity,
          MaintenanceActivity,
          DoNothing };

     struct ActivityOption {
          ActivityType activityType;
          short prioritySequenceNumber;
          Global::TimeStamp deadline;
          AbstractIF::ResourceSequence resources;
          AbstractIF::MaterialGroupSequence materialGroups;
          DurablesManagement::DurableSequence durables;
          Global::Properties activitySpec;
     }; // ActivityOption

     struct ActivityForecast {
          ActivityType activityType;
          short timeSequenceNumber;
          AbstractIF::ResourceSequence resources;
          AbstractIF::MaterialGroupSequence materialGroups;
          DurablesManagement::DurableSequence durables;
          Global::TimeStamp projectedStart;
          Global::TimeStamp projectedFinish;
          Global::Properties activitySpec;
     }; // ActivityForecast

     typedef sequence <ActivityOption> ActivityList;

     typedef sequence <ActivityForecast> ForecastList;

interface SchedulingManager : FactoryOperations::ComponentManager {

     exception ResourceNotFound {
          AbstractIF::Resource requestedObject;};

     exception MaterialGroupNotFound {
          AbstractIF::MaterialGroup requestedObject;};

     exception DurableNotFound {
          DurablesManagement::Durable requestedObject;};
```

```
ActivityList whatNextForResourceByCount (
      in AbstractIF::Resource aResource)
      in short listCount)
      raises (Global::FrameworkErrorSignal,
            ResourceNotFound);

ActivityList whatNextForMaterialGroupByCount (
      in AbstractIF::MaterialGroup aMaterialGroup,
      in short listCount)
      raises (Global::FrameworkErrorSignal,
            MaterialGroupNotFound);

ActivityList whatNextForDurableByCount (
      in DurablesManagement::Durable aDurable)
      in short listCount)
      raises (Global::FrameworkErrorSignal,
            DurableNotFound);

ActivityList whereNextForMaterialGroup (in AbstractIF::MaterialGroup aMaterialGroup)
      raises (Global::FrameworkErrorSignal,
            MaterialGroupNotFound);

ActivityList whereNextForDurable (in DurablesManagement::Durable aDurable)
      raises (Global::FrameworkErrorSignal,
            DurableNotFound);

ForecastList forecastForResourceByTime (
      in AbstractIF::Resource aResource,
      in Global::TimeStamp lookAheadTime)
      raises (Global::FrameworkErrorSignal,
            ResourceNotFound);

ForecastList forecastForMaterialGroupByTime (
      in AbstractIF::MaterialGroup aMaterialGroup,
      in Global::TimeStamp lookAheadTime)
      raises (Global::FrameworkErrorSignal,
            MaterialGroupNotFound);

ForecastList forecastForDurableByTime (
      in DurablesManagement::Durable aDurable,
      in Global::TimeStamp lookAheadTime)
      raises (Global::FrameworkErrorSignal,
            DurableNotFound);

ForecastList forecastForResourceByCount (
      in AbstractIF::Resource aResource,
      in short lookAheadCount)
      raises (Global::FrameworkErrorSignal,
            ResourceNotFound);

ForecastList forecastForMaterialGroupByCount (
      in AbstractIF::MaterialGroup aMaterialGroup,
      in short lookAheadCount)
      raises (Global::FrameworkErrorSignal,
            MaterialGroupNotFound);

ForecastList forecastForDurableByCount (
      in DurablesManagement::Durable aDurable,
      in short lookAheadCount)
      raises (Global::FrameworkErrorSignal,
            DurableNotFound);

}; // SchedulingManager
}; // module Scheduling
```

```
#endif // _CIMFW_SCHEDULING_
}; // mdoule CIMFW
```

NOTICE: SEMI makes no warranties or representations as to the suitability of the standard set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

RELATED INFORMATION 1
IMPLEMENTATION GUIDANCE FOR INPUT OF FACTORY STATE INFORMATION TO THE SCHEDULING COMPONENT

NOTE: This related information is not an official part of SEMI E105. This related information was approved for publication by full letter ballot procedures on August 28, 2000.

NOTE: This related information is included with the Scheduling Component specification to aid the readers in understanding the intent and use of the standard.

R1-1 The Scheduling Component has two types of interfaces:

A. Scheduling and Dispatching Service Interface - this is the interface for services provided by the Scheduling Component in response to requests from the Factory Operations Component or some other client. It also includes interfaces to support asynchronous actions for an active scheduler or dispatcher which provides the same information content on an event driven basis rather than requiring a client request.

   This is included within the scope of the current standard.

B. Scheduling and Dispatching Factory Input Interface - this is the interface used by other components to provide updated information on factory state used by the Scheduling Component in making scheduling and dispatching decisions. The updates are typically provided by other CIM Framework components to the Scheduling Component asynchronously.

   This is NOT included within the scope of the current standard.

R1-2 The goal for CIM Framework standards is to eventually standardize both of these types of interface. However, at this time there are two major issues that prevent inclusion of the Factory Input Interface to the Scheduling Component.

1. There is inadequate consensus on the technical maturity of the currently available asynchronous input mechanisms. While the CIM Framework specifies events for delivery via an Event Broker mechanism, it is not assumed that the quality of service for events will include guaranteed delivery and performance adequate for keeping a Scheduling Component current with real-time factory state changes. For a more complete discussion of the CIM Framework technical architecture issues, see SEMI E96.
2. Some key components of the CIM Framework needed to provide factory state to the Scheduling Component have not yet been approved as SEMI standards.

R1-3 The following approach is suggested as a possible way to address Scheduling Component Input in the future:

1. Identify the most important inputs required by the Scheduling Component in a follow-on ballot for the Scheduling Component to be used during the interim period until the full component specifications containing those events are approved as SEMI standards.
2. Suppliers who implement the Scheduling Component should take advantage of existing standard interfaces defined in adopted CIM framework standards and use this experience as expert input on future updates to this standard.
3. Encourage technical studies to provide added understanding of the tradeoffs and limitations that impact standardization of this interface in the future.

RELATED INFORMATION 2
SCENARIOS FOR SCHEDULING AND DISPATCHING.

NOTE: This related information is not an official part of SEMI E105. This related information was approved for publication by full letter ballot procedures on August 28, 2000.

These scenarios show possible implementations of the SEMI Scheduling Component. The information shown here is intended to be a guide rather than describe prescriptive implementation rules. Some of the message sequences and implied functionality included in these scenarios will vary across different implementations.

This section focuses primarily on the interface between the Scheduling Component and factory operations component. The messaging between the lower level components is for informational purposes, and may not actually reflect what is developed in the future when standards for those components have been completed.

R2-1 *Scenario 1* — Simulator Based Scheduler/Dispatcher, Stocker is Not Managed by Scheduling Component

R2-1.1 Lot 1 is scheduled for equipment 2 at 10:00. Lot is held in stocker until ready to start on equipment 2. Factory operations makes the carrier delivery decisions.

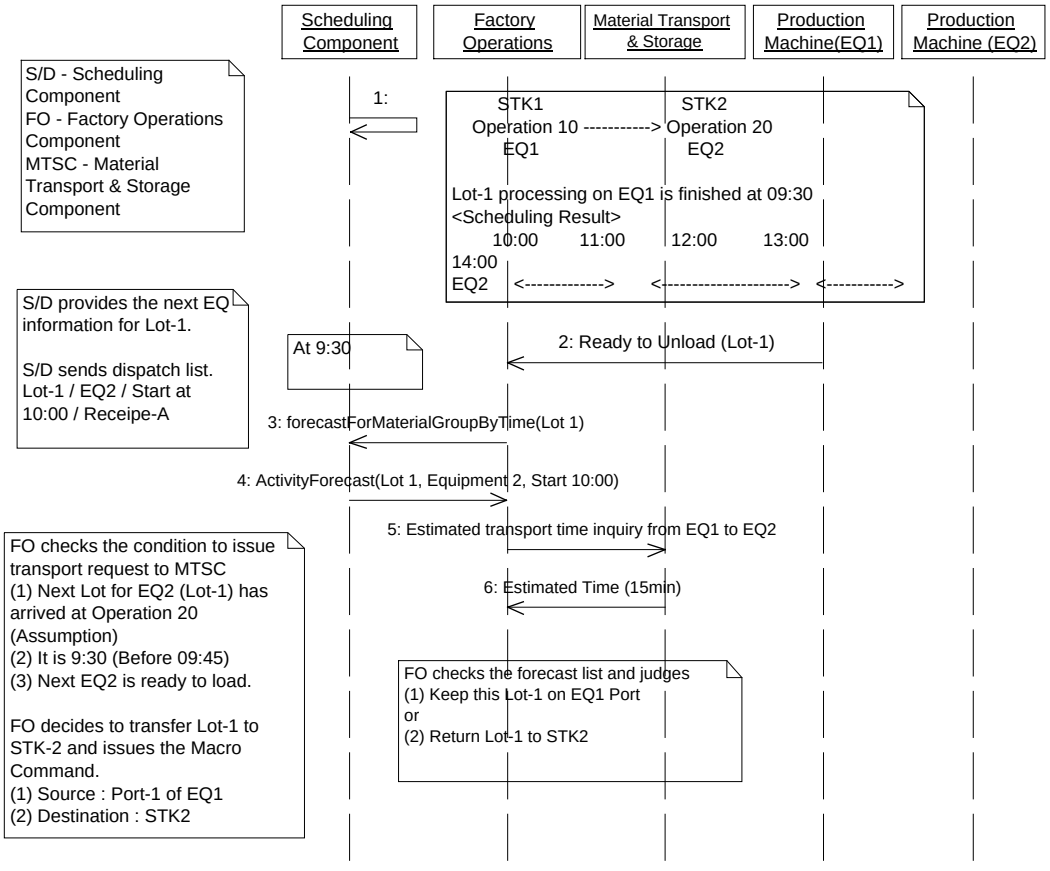


Figure R2-1
Scenario Case 1

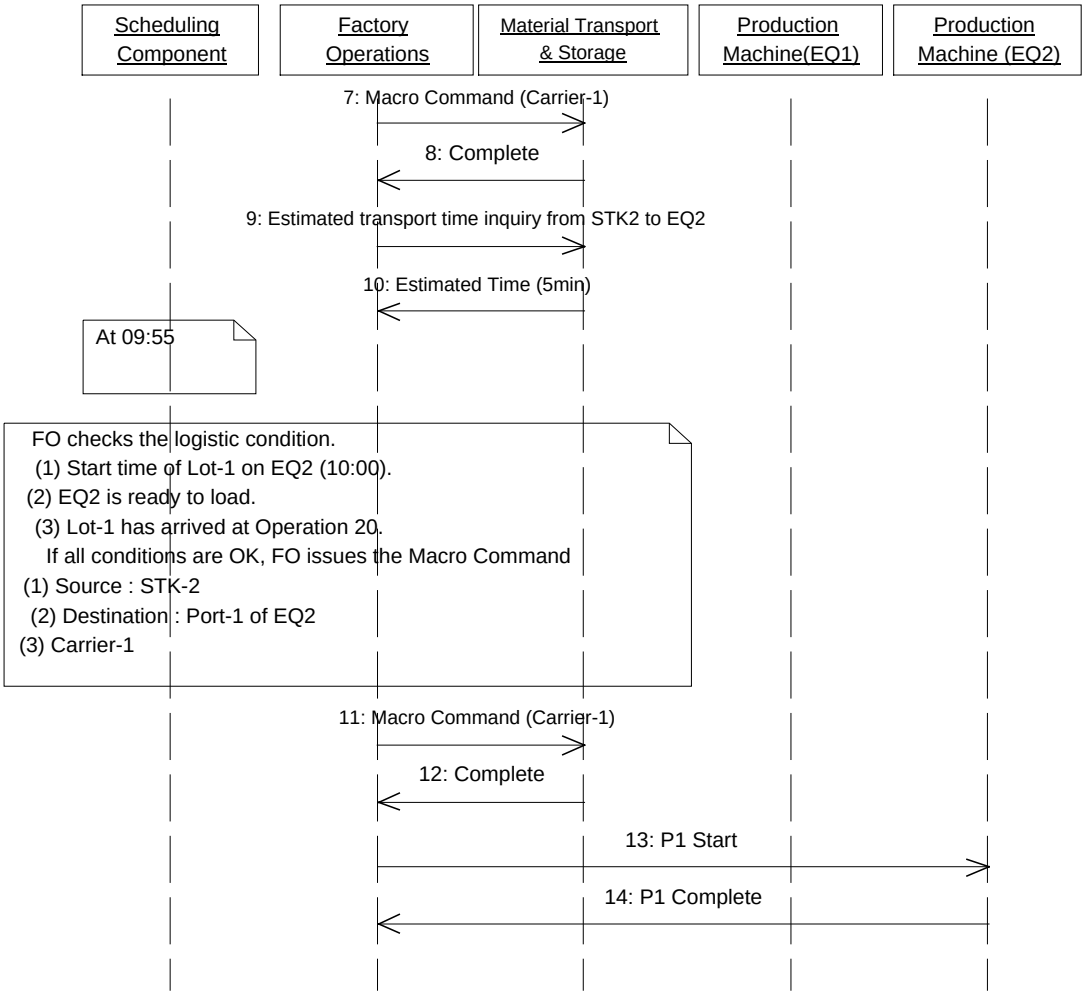


Figure R2-2
Scenario Case 1 Continued

R2-2 *Scenario 2* — Simulator Based Scheduler/Dispatcher, Stocker is Not Managed by Scheduling Component

R2-2.1 Lot 1 is scheduled for equipment 2. Lot is transported directly from Eqp1 to Eqp 2. Factory Operations makes the carrier delivery decisions.

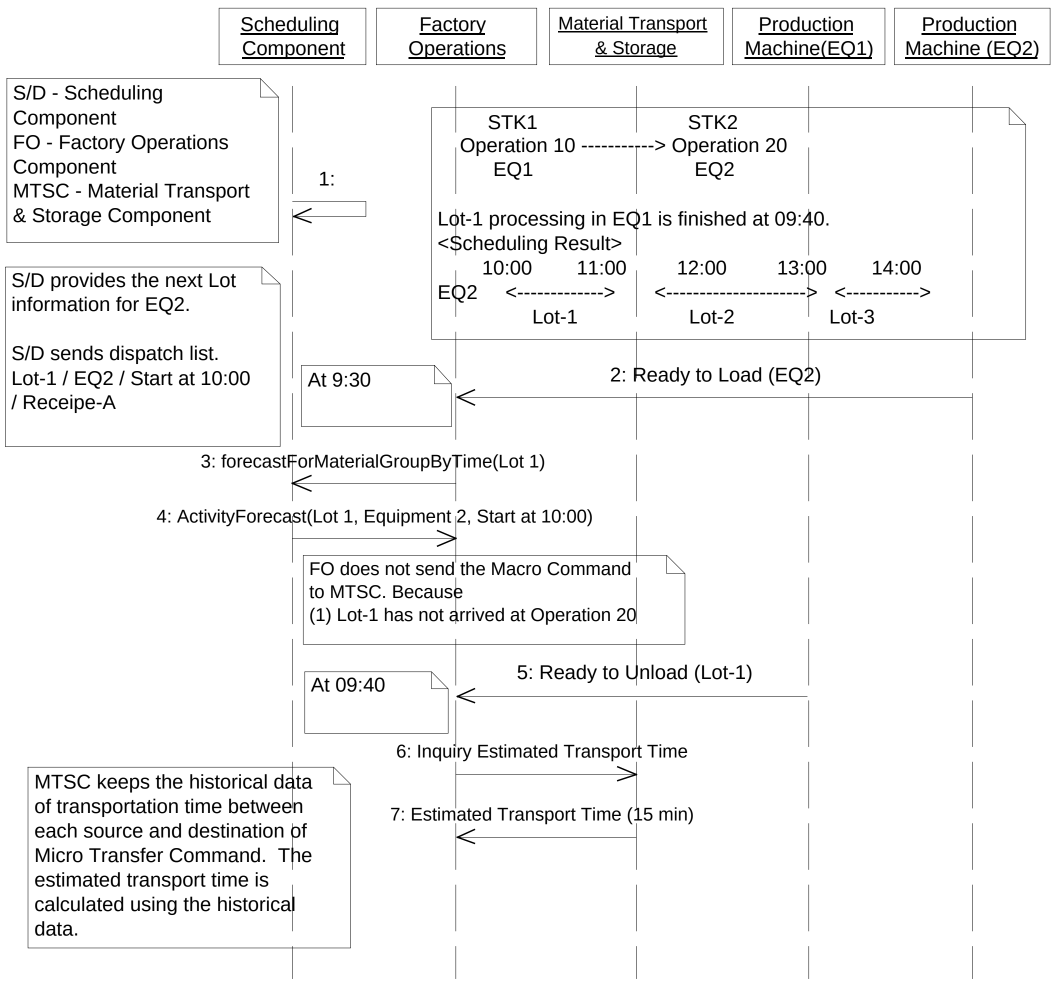


Figure R2-3
Scenario Case 2

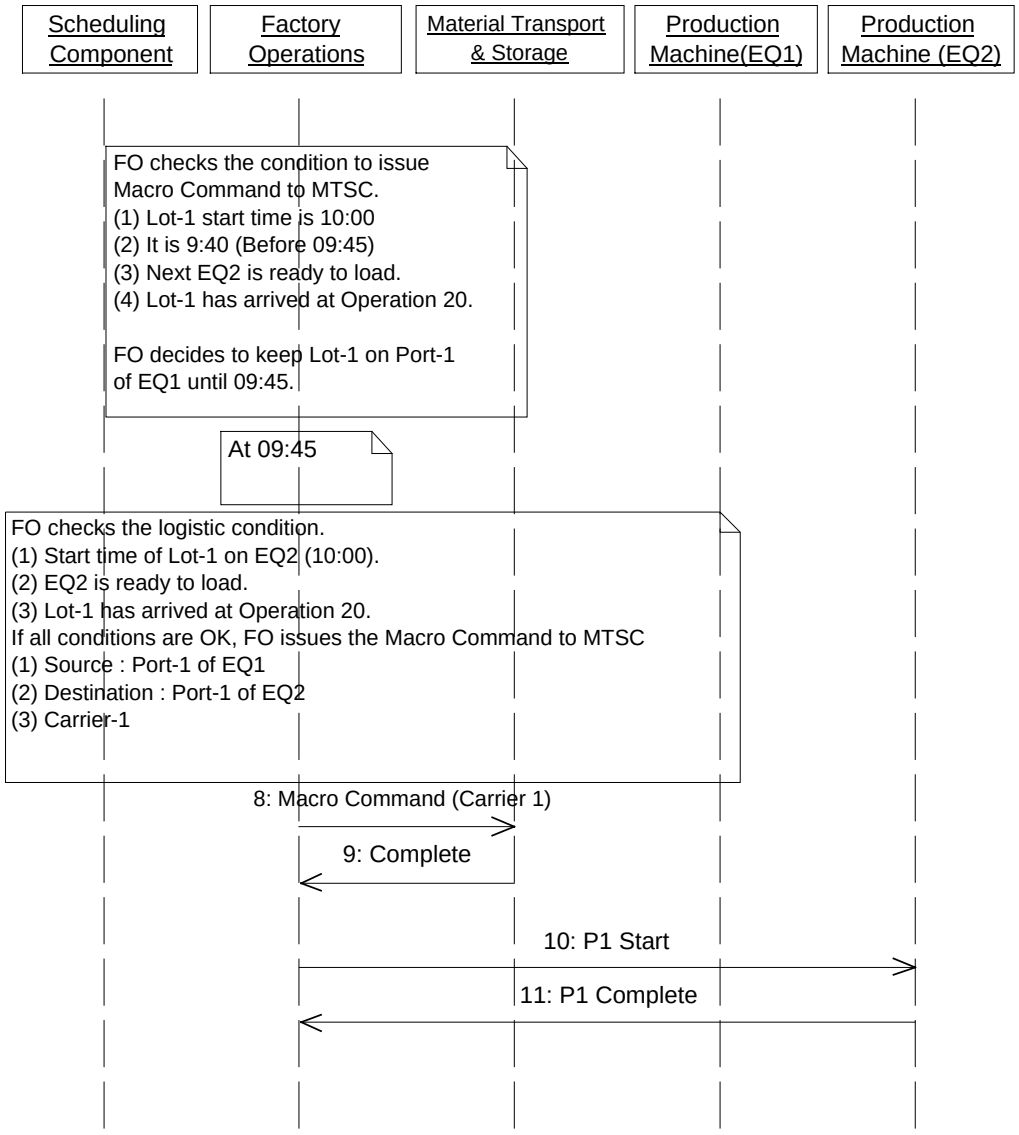
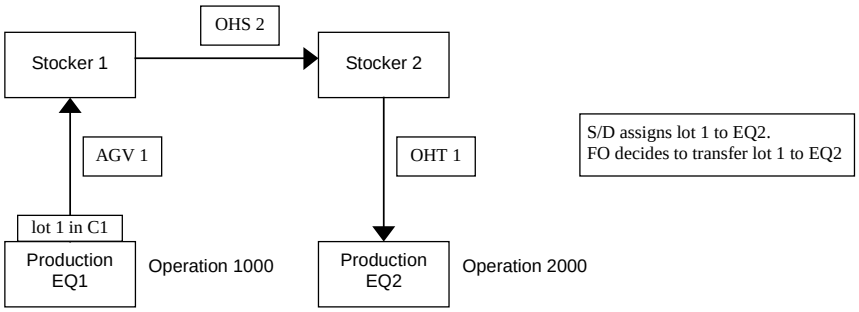
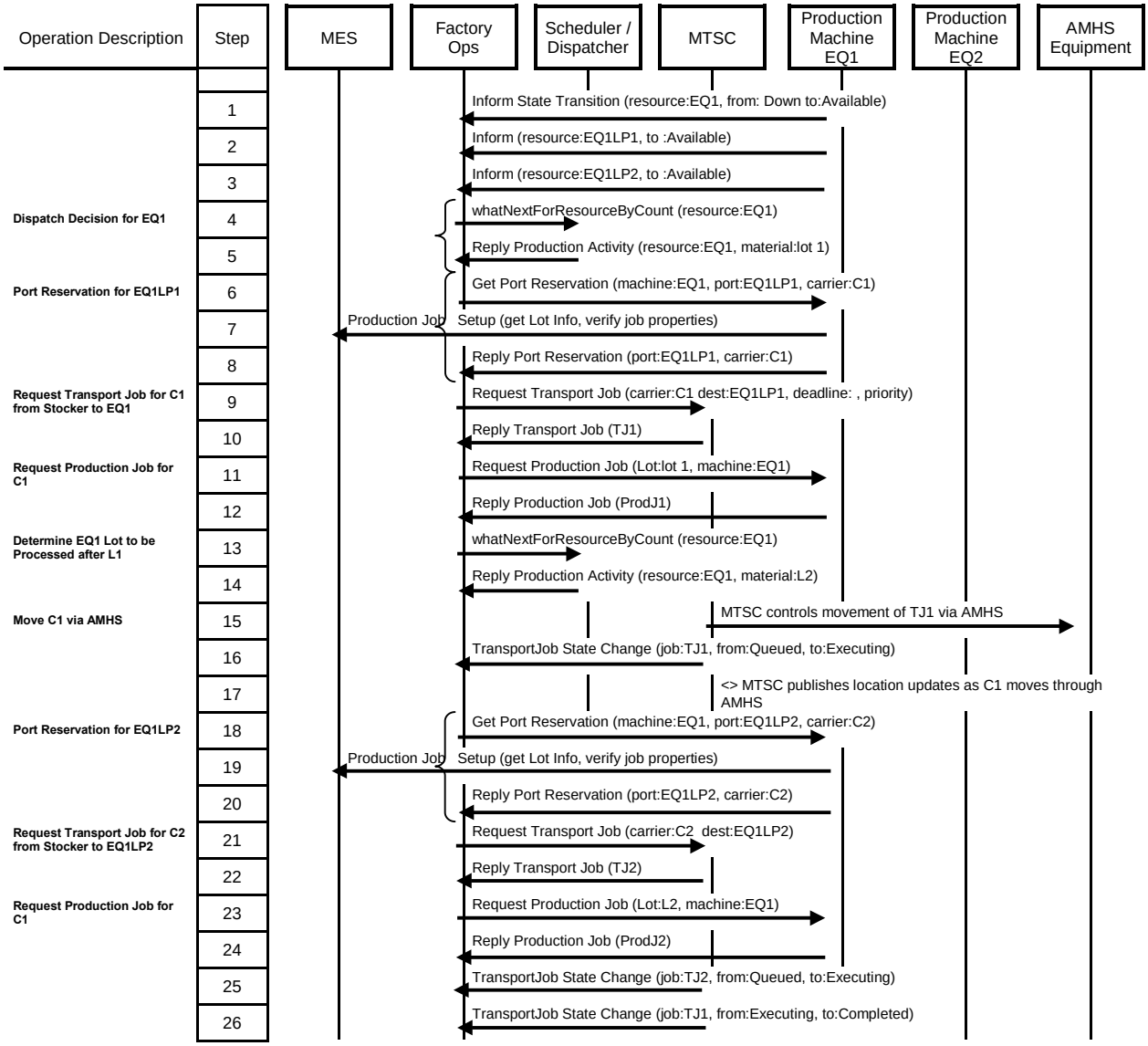


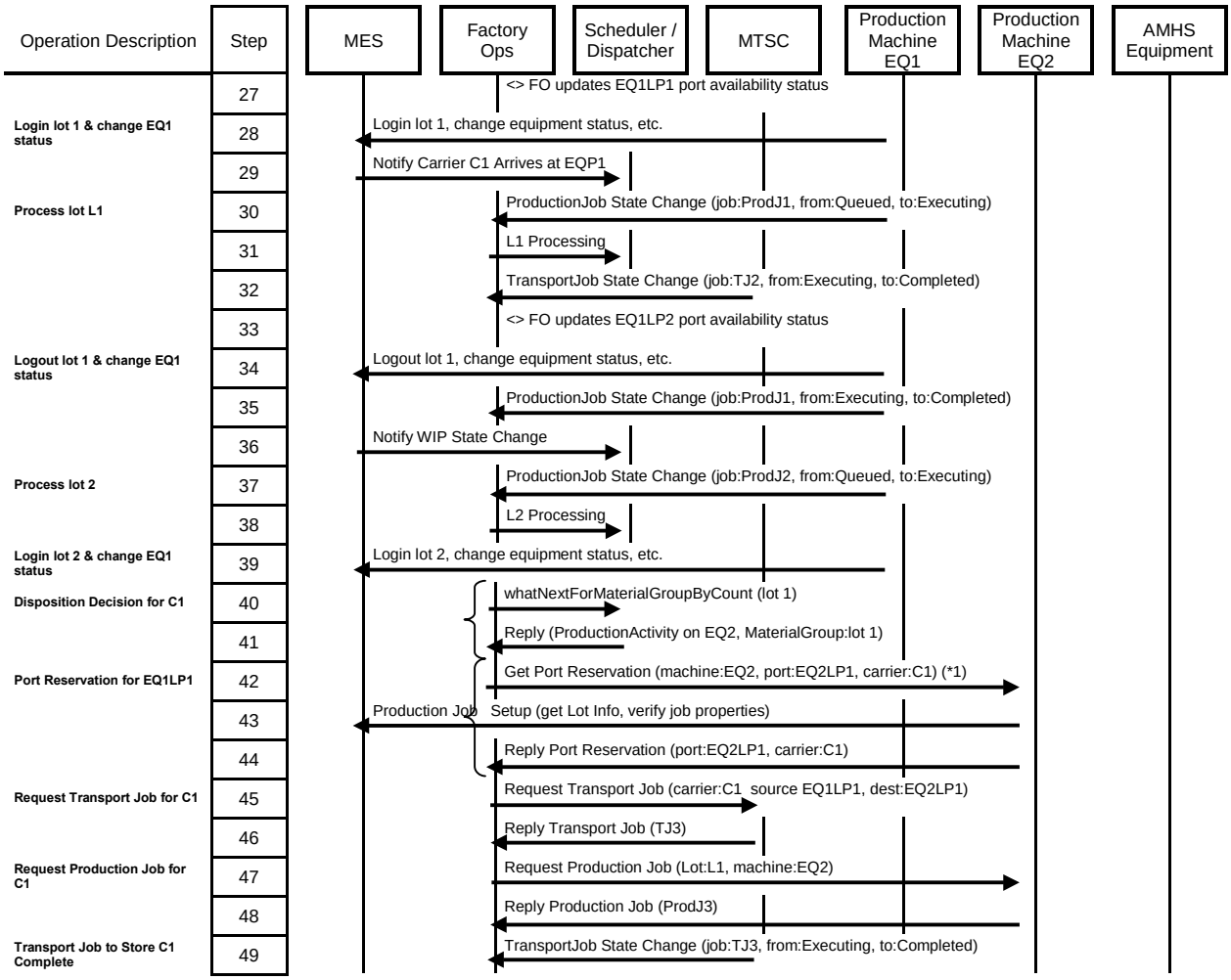
Figure R2-4
Scenario Case 2 Continued

R2-3 *Scenario 3* — Dispatch List Based Scheduler/Dispatcher, Stocker is Not managed by Scheduling Component

R2-3.1 Dispatch list based. EQ1->EQ2. After lot 1 processing on EQ1 is finished, Scheduler assigns lot 1 to EQ2. FO checks the EQ2 port status and finds the available port. FO makes the logistic decision to transfer lot 1 from EQ1 to EQ2 directly. Scheduler does not assign lots to Stocker. In this scenario 3, the direct transfer "from EQ1 to EQ2" is described.



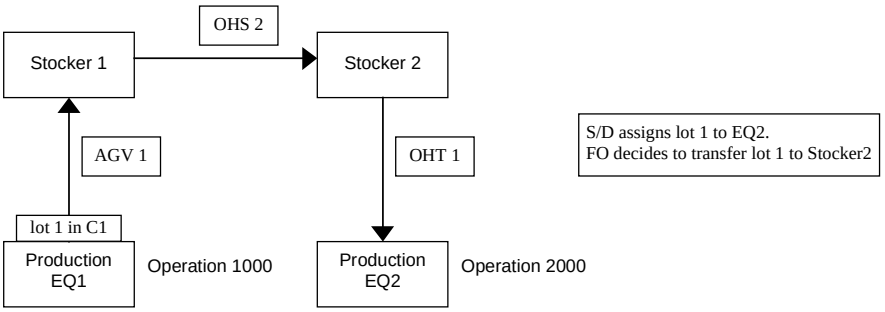


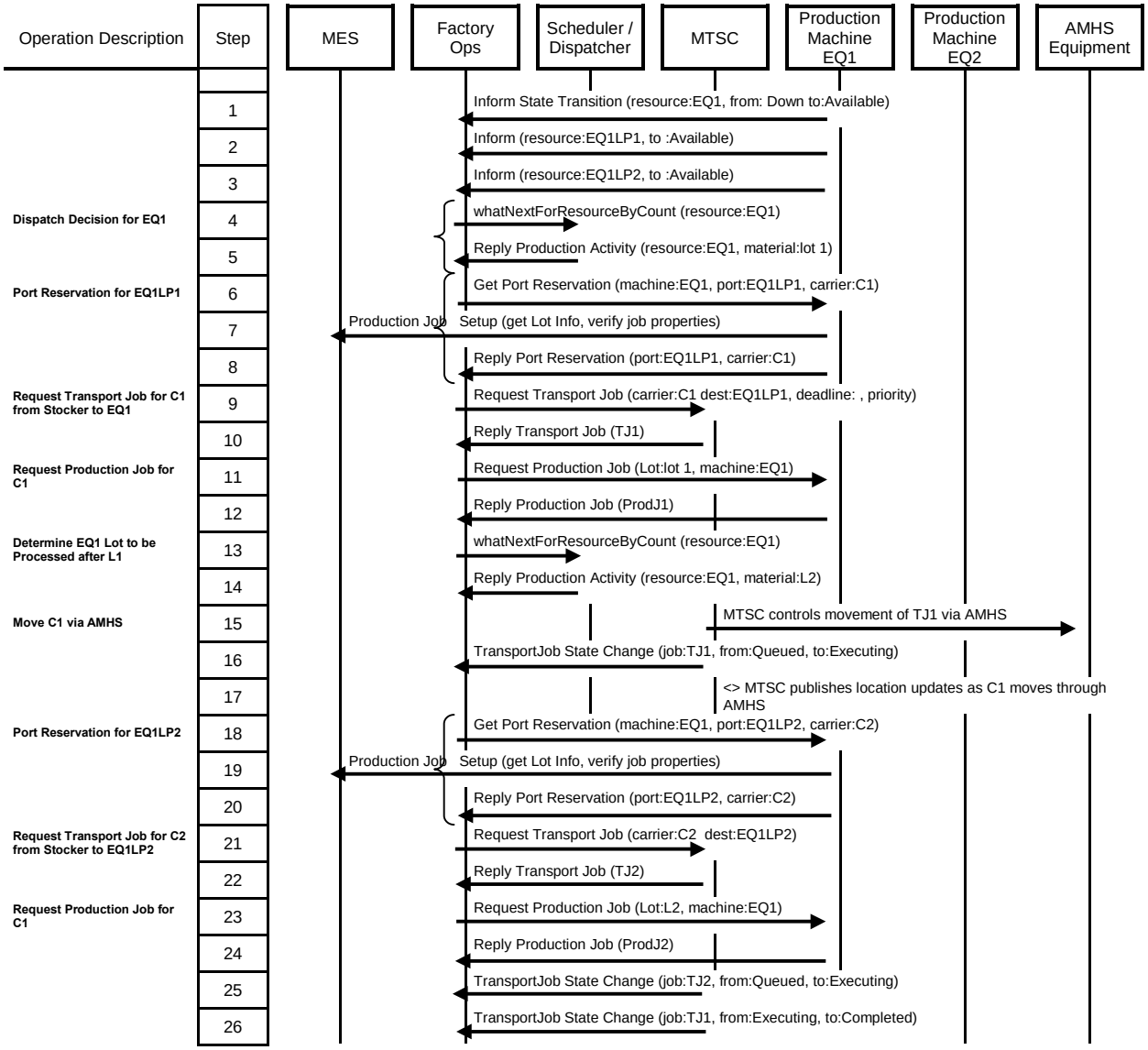


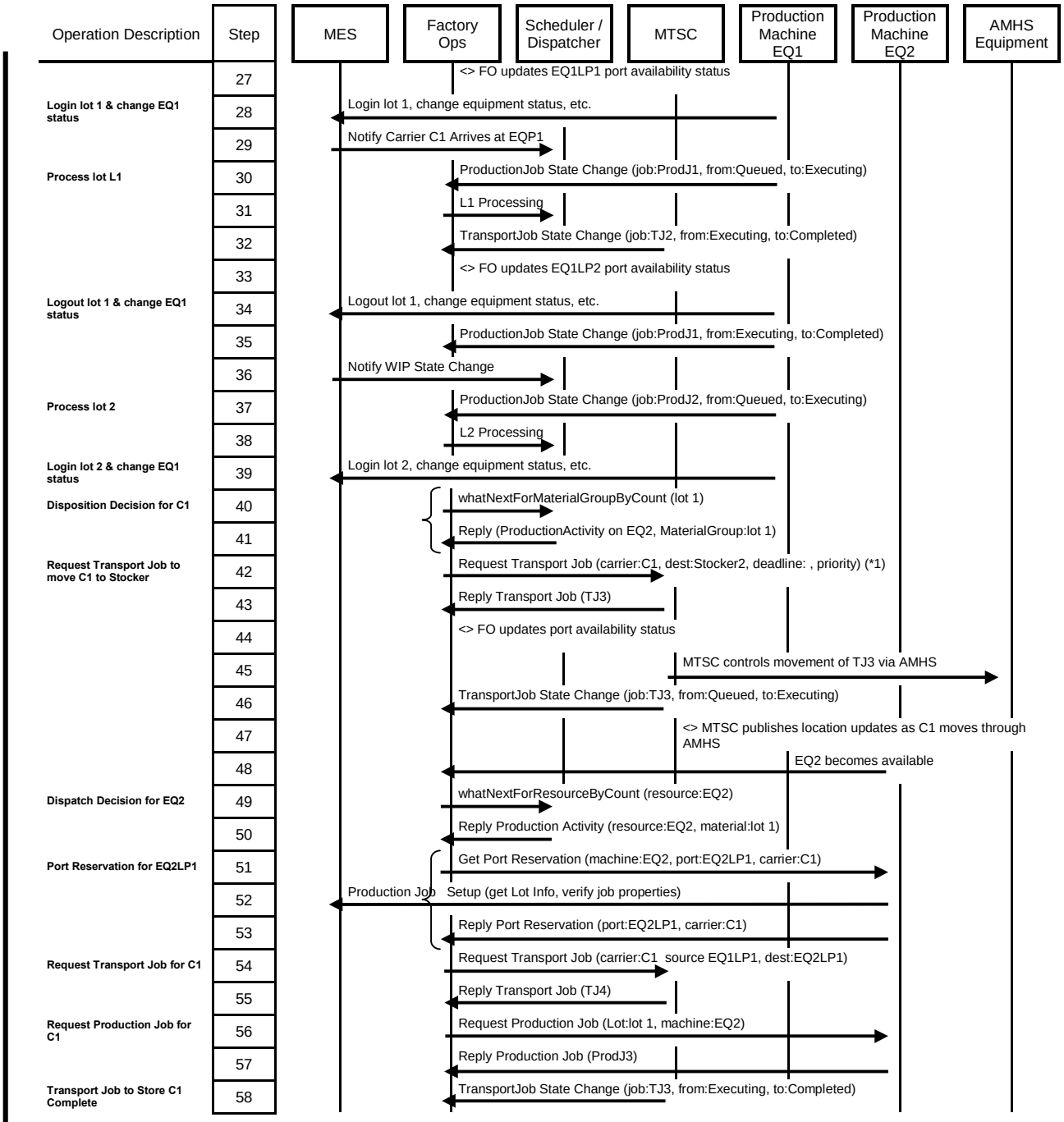
(*1) FO checks port status on tool. Finds port available. Delivers lot directly to EQ2.

R2-4 *Scenario 4* — Dispatch List Based Scheduler/Dispatcher, Stocker is Not managed by Scheduling Component

R2-4.1 Dispatch list based. EQ1->Stocker->EQ2. After lot 1 processing on EQ1 is finished, Scheduler assigns lot 1 to EQ2. FO checks the EQ2 port status and finds No available port. FO makes the logistic decision to transfer lot 1 from "EQ1 to Stocker 2" instead of "from EQ1 to EQ2". Scheduler does not assign lots to Stocker. In this scenario 4, the separate transfers "from EQ1 to Stocker 2" and "from Stocker 2 to EQ2" are described.



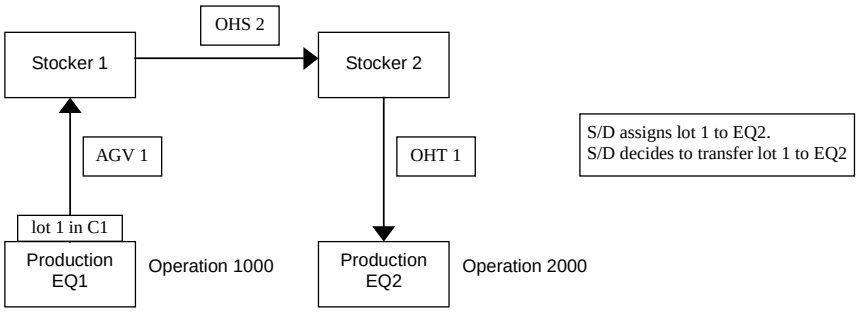


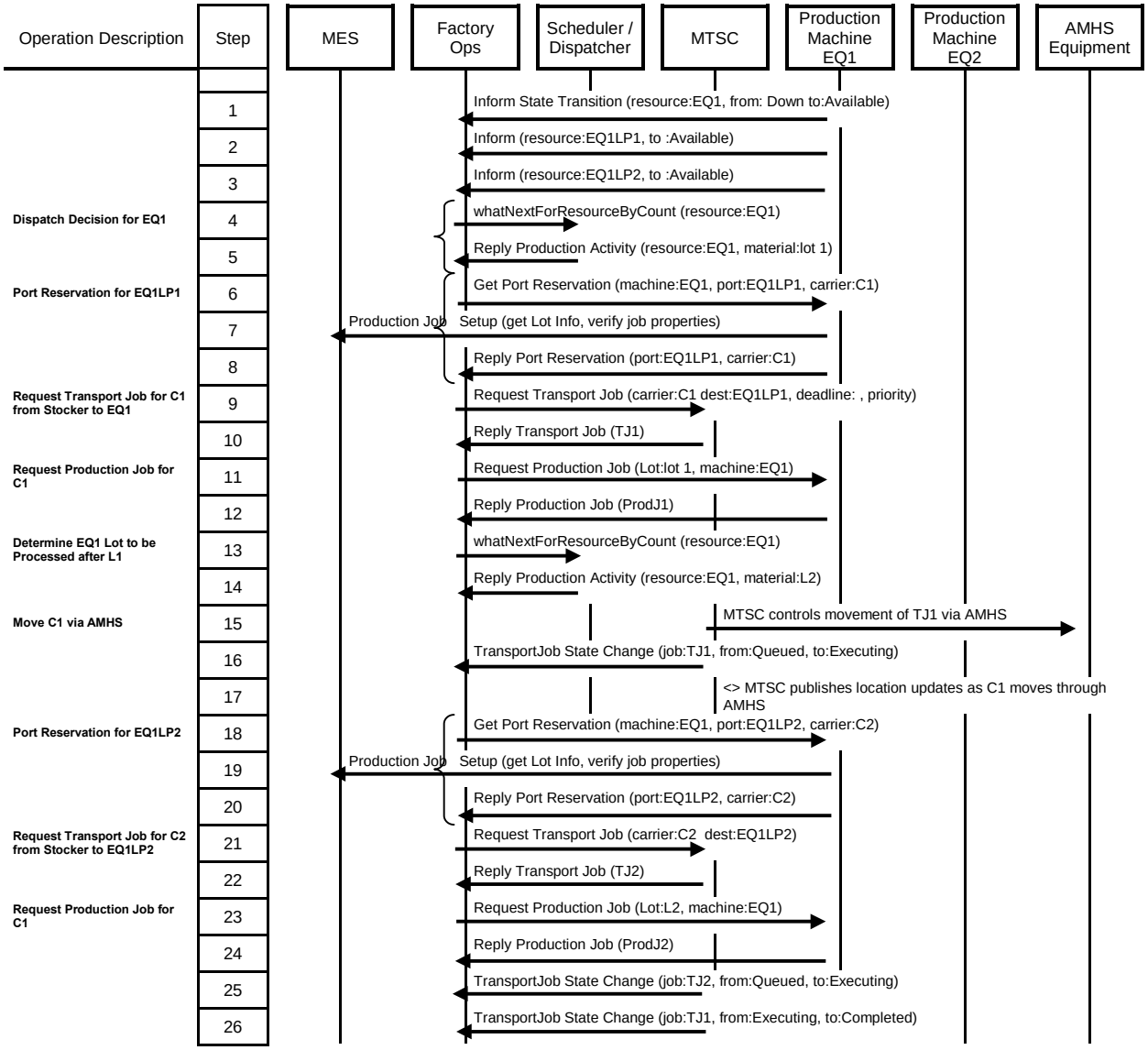


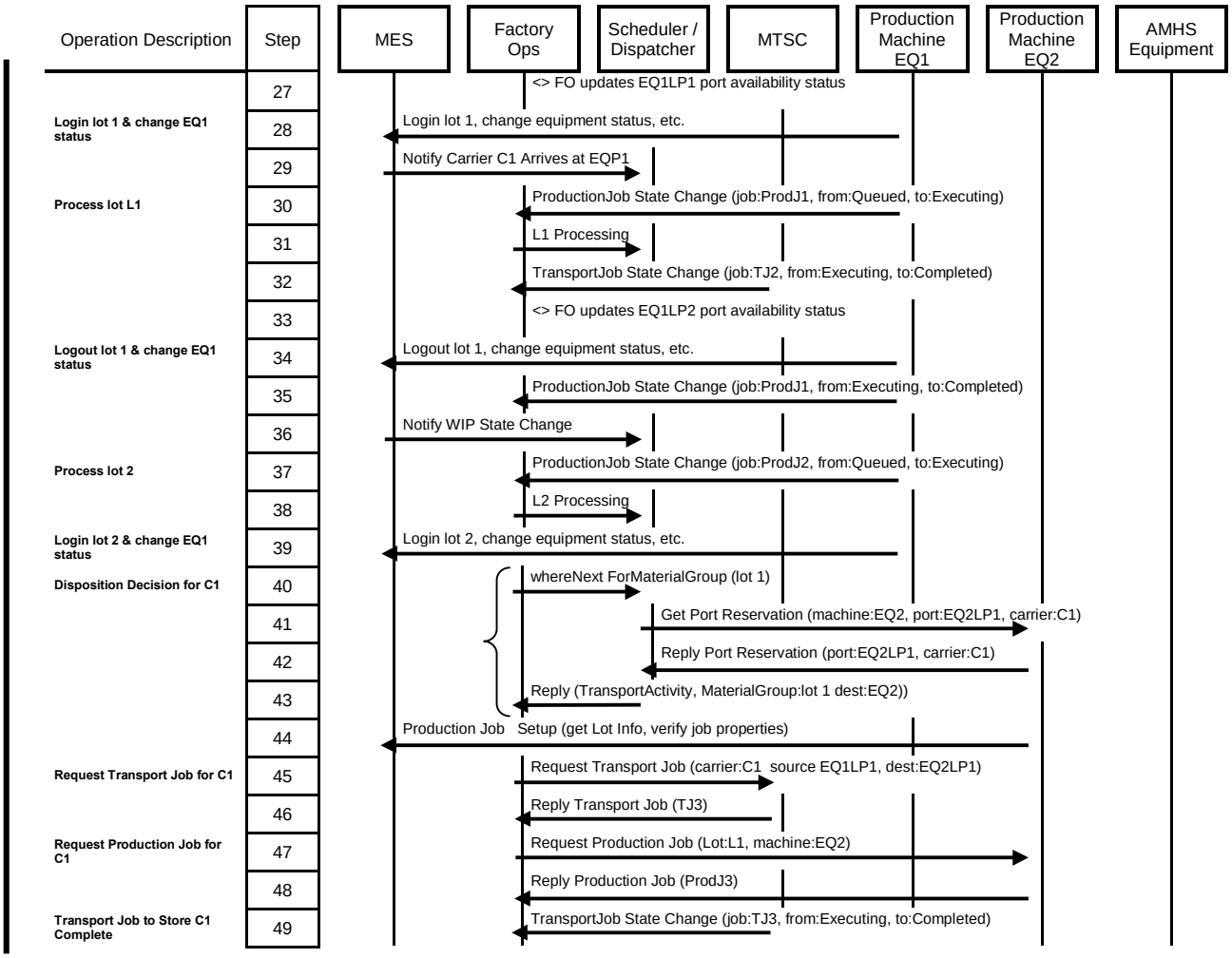
(*1) FO checks port status on tool. Finds no ports available. Delivers lot to stocker close to tool.

R2-5 *Scenario 5* — Dispatch List Based Scheduler/Dispatcher, Stocker is managed by Scheduling Component

R2-5.1 Dispatch list based. EQ1->EQ2. After lot 1 processing on EQ1 is finished, Scheduler assigns lot 1 to EQ2. Scheduler additionally checks the EQ2 port status and finds the available port. Scheduler makes the logistic decision to transfer lot 1 from EQ1 to EQ2 directly. Scheduler assigns lots to Stocker when the next assigned equipment is not available. In this scenario 5, the direct transfer "from EQ1 to EQ2" is described.

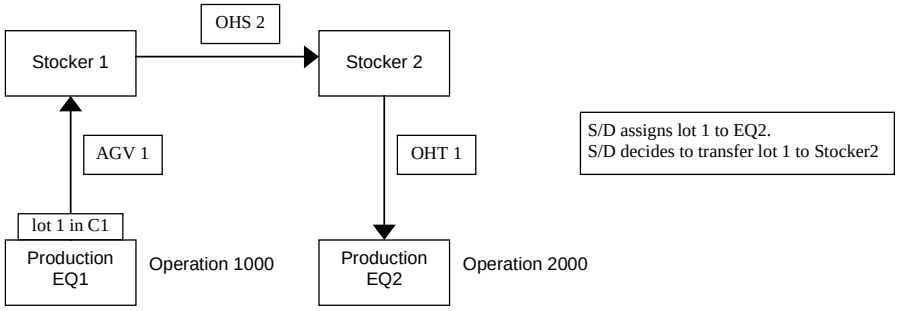


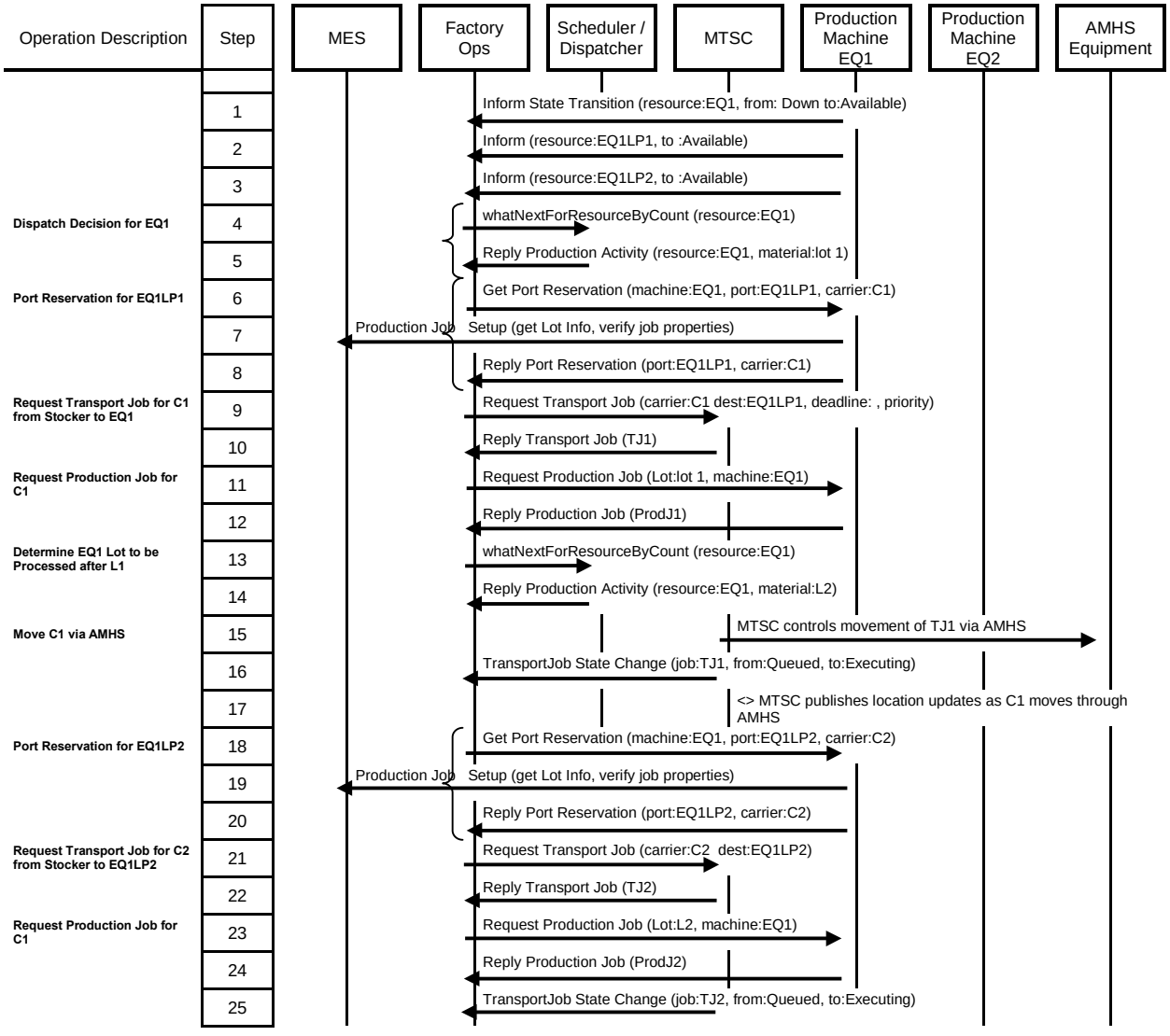


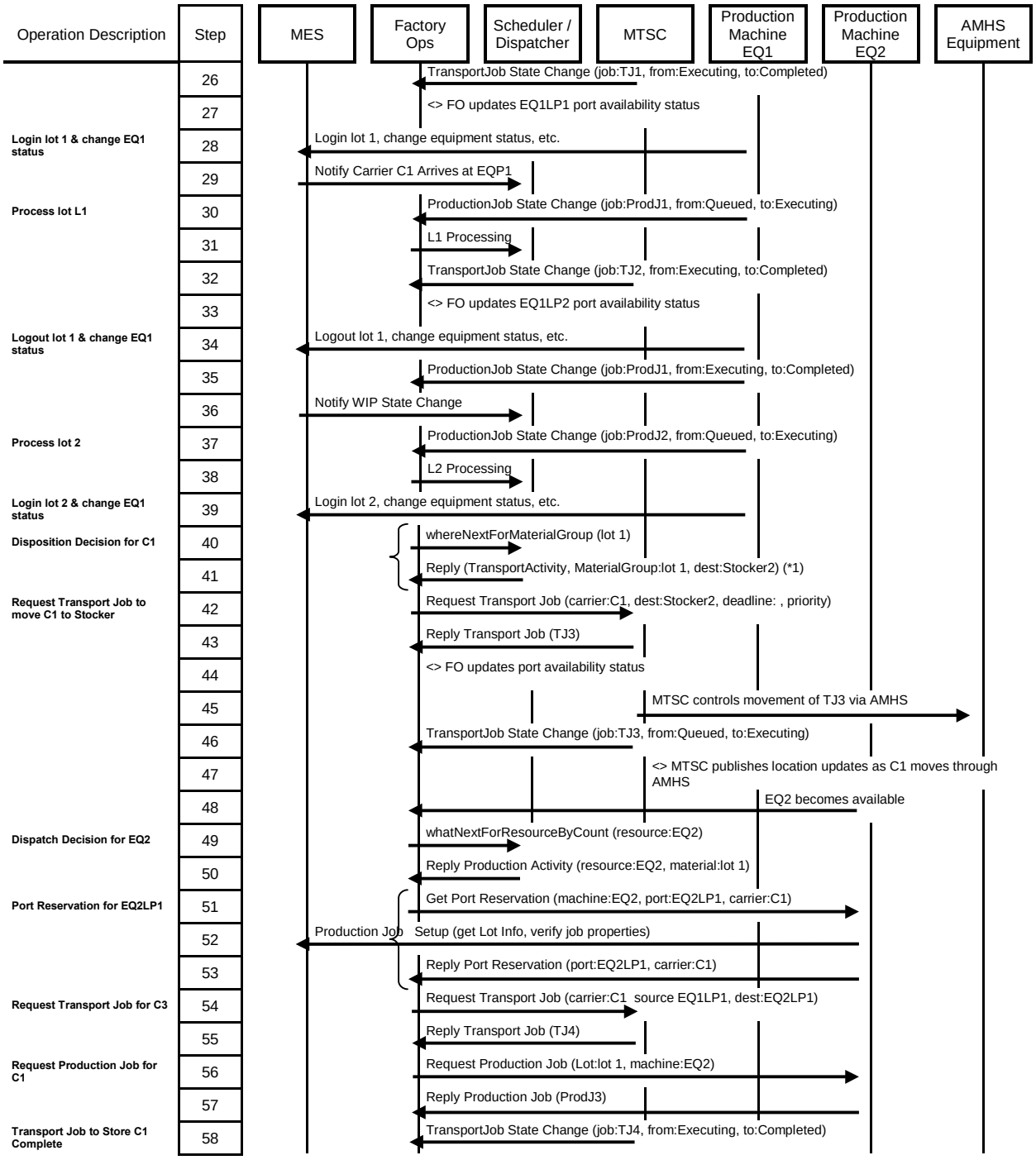


R2-6 *Scenario 6* — Dispatch List Based Scheduler/Dispatcher, Stocker is managed by Scheduling Component

R2-6.1 Dispatch list based. EQ1->Stocker->EQ2. After lot 1 processing on EQ1 is finished, Scheduler assigns lot 1 to EQ2. FO checks the EQ2 port status and finds the available port. Scheduler makes the logistic decision to transfer lot 1 "from EQ1 to Stocker 2" instead of "from EQ1 to EQ2". Scheduler assigns lots to Stocker when the next equipment is not available. In this scenario 6, the separate transfers "from EQ1 to Stocker 2" and "from Stocker 2 to EQ2" are described.







(*1) Scheduler/Dispatcher checks the next equipment EQ2 status and port availability. EQ2 Status: Processing, Available Ports: No vacant port. Scheduler/Dispatcher decides to transfer lot 1 to Stocker 2.

RELATED INFORMATION 3
ASSUMED CASE OF SCHEDULER/DISPATCHER

NOTE: This related information is not an official part of SEMI E105. This related information was approved for publication by full letter ballot procedures on February 1, 2001.

The assumed two cases of scheduler/dispatcher were used to have an analysis and to make scenarios when the interfaces defined in SEMI E105 ware developed. This related information describes the assumed two case of scheduler/dispatcher to make easier for the standard user to understand SEMI E105. SEMI E105 does not limit the case of Scheduler/Dispatcher to these two assumed scheduler/dispatcher.

R3-1 *Dispatch List Based Scheduler/Dispatcher*

R3-1.1 *Basic Functions*

Dispatch List Based Scheduler/Dispatcher has a waiting lots list in each operation and has a capability to prioritize the existing lots.

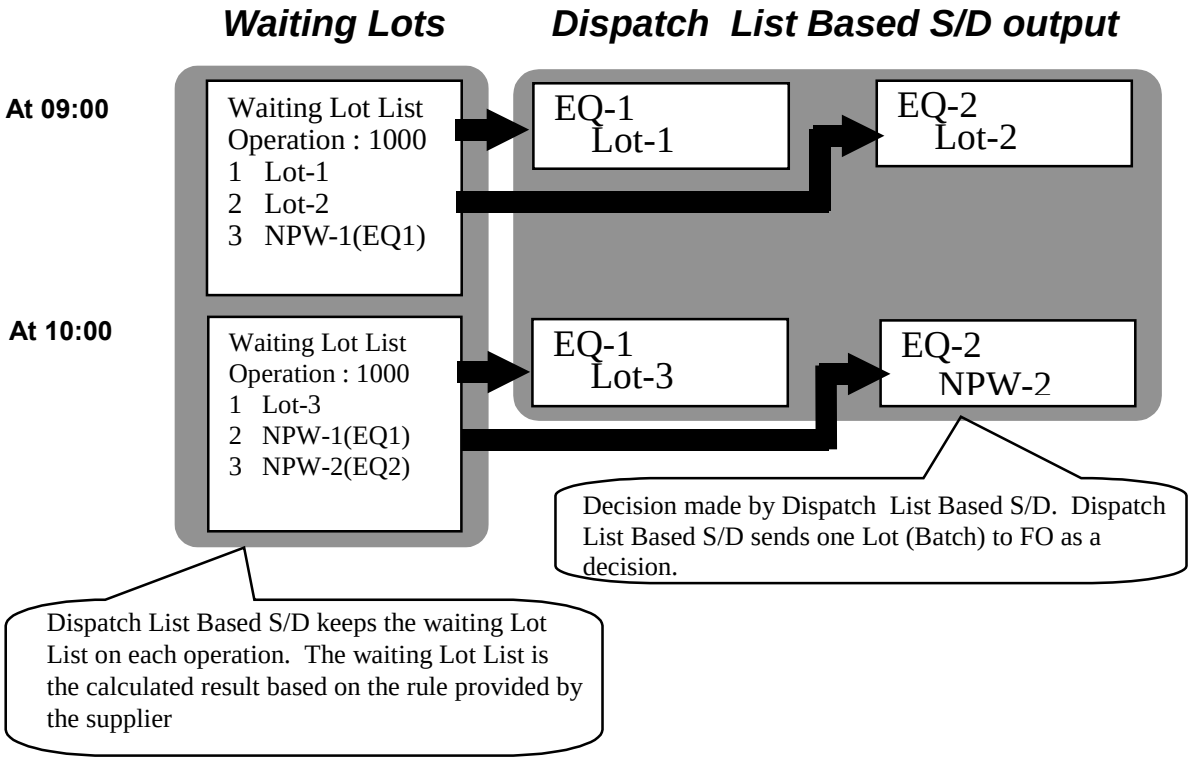


Figure R3-1

R3-1.2 *Case 1* – Factory Operation decides storage machine (Refer to Scenarios R2-3 and R2-4)

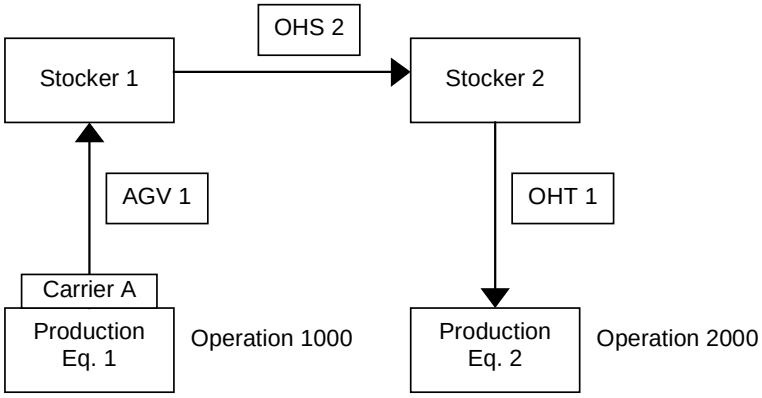


Figure R3-2

1. Carrier A processing on Eq.1 (Operation 1000) is finished.
2. Next operation for Carrier A is “2000” and the production equipment in operation “2000”is Eq.2 only.
3. Factory Operation makes the logistic decision .

(1) Transfer Carrier A from Eq.1 to Eq.2

or

(2) Transfer Carrier A from Eq.1 to Stocker 2.

4. Factory Operation uses the following interfaces defined in section 6.1.6. (*1)

- whatNextForResourceByCount
- whatNextForMaterialGroupByCount
- whatNextForDurableByCount

5. Factory Operation dose not use the following interfaces defined in section 6.1.6. (*1)

- whereNextForMaterialGroup
- whereNextForDurable

6. Scheduler / Dispatcher shows the waiting Lot list in operation “2000” (Eq.2) to Factory Operation

R3-1.2 *Case 2* – Scheduler / Dispatcher decides storage machine (Refer to Scenarios R2-5 and R2-6)

1. Carrier A processing on Eq.1 (Operation 1000) is finished.
2. Next operation for Carrier A is “2000” and the production equipment in operation “2000”is Eq.2 only.
3. Scheduler / Dispatcher checks the port status of Eq.2 and makes the logistic decision .

(1) Transfer Carrier A from Eq.1 to Eq.2

or

(2) Transfer Carrier A from Eq.1 to Stocker 2.

4. Factory Operation uses the following interfaces defined in section 6.1.6. (*1)

- whatNextForResourceByCount
- whereNextForMaterialGroup
- whereNextForDurable

5. Factory Operation dose not use the following interfaces defined in section 6.1.6. (*1)

- whatNextForMaterialGroupByCount
- whatNextForDurableByCount

R3-2 *Simulator Based Scheduler/Dispatcher (Refer to Scenarios R2-1 and R2-2)*

R3-2.1 *Basic Functions*

Simulator Based Scheduler / Dispatcher has a forecast list for each production equipment and provides the lot list with forecast time stamp to Factory Operation.

Factory Operation uses "forecastFor…" interfaces defined in section 6.1.6. (*1)

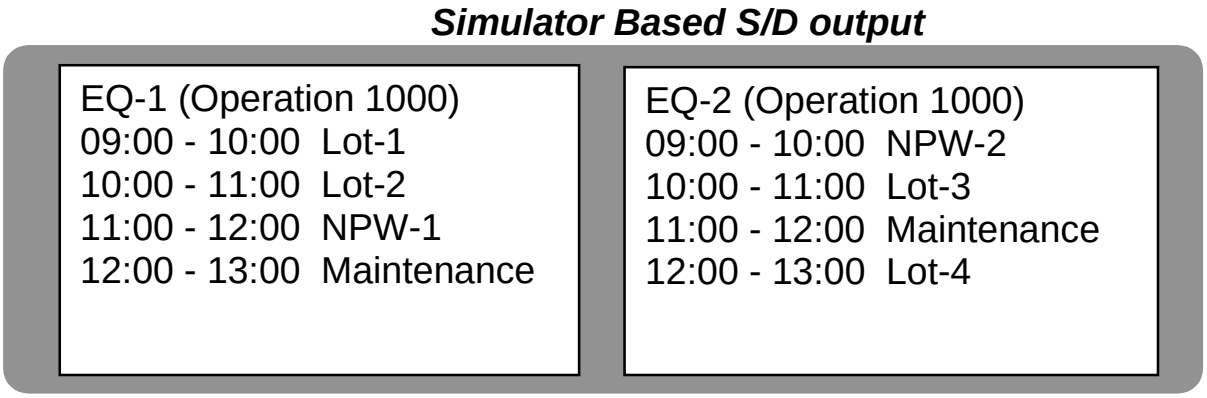


Figure R3-3

Remarks

In (*1), the conformance of each interfaces to each case of Scheduler/Dispatcher are described. These are only for reference and are not the requirement to the Scheduler/Dispatcher product. For example, Simulator Based Scheduler / Dispatcher may support "whatNextFor…" interfaces.

SEMI E107-1102
SPECIFICATION OF ELECTRIC FAILURE LINK DATA FORMAT FOR YIELD MANAGEMENT SYSTEM

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the Japanese Information & Control Committee. Current edition approved by the Japanese Regional Standards Committee on July 19, 2002. Initially available at www.semi.org October 2002; to be published November 2002. Originally published March 2001; previously published November 2001.

1 Purpose

1.1 The objective of this document is standardization of the specific data format passed from the test equipment to the Yield Management System. The Yield Management System is a kind of data server for detail test data and geometrical defect data of patterns on a wafer as described in the Terminology section of this document.

1.2 This document assumes a Yield Management System in which test equipment electrical failure information is managed and analyzed in an integrated manner. Examples of test equipment failure information include bit map data, bin data, and inpsection information obtained by devices such as wafer inspection equipment and review tools. Standardization of the data file format helps to reduce the development burden on customers and related vendors.

2 Scope

2.1 This document specifies the data file format for transferring from test equipment to a Yield Management System.

2.2 This document is an extension of the general map data item standard, i.e. SEMI G81, and the general map data format document, currently under development. This document does not redefine the general specification.

2.3 The scope of this document is just defining data items and their formats. Data file creation methods, data creating environments and file naming conventions are outside of the scope of this document. Also, communication protocols to transfer the data are beyond the scope of this document.

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

3 Limitations

3.1 This standard defines extensional data for YMS Link Data map for generic test data map standardized by SEMI G81 and G85, data format of each item for YMS Link Data is defined independently as extra definition here.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E30.1 — Inspection and Review Specific Equipment Model (ISEM)

SEMI G81 — Specification for Map Data Items

SEMI G85 — Specification for Map Data Format

NOTE 1: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 *Defintions*

5.1.1 *analog data* — One of three categories of data: measured values such as voltage or current obtained when test equipment measures a semiconductor device's electrical characteristics, parameter values which are test parameters when measuring, and limit values which are decision parameters if test results are pass or fail.

5.1.2 *bin data* — same as Category below.

5.1.3 *category* — data indicating the type of electric failure or rank of characteristics of die tested by the test equipment. Used in the same manner as Bin Data in this standard.

5.1.4 *cell-logical-address* — gives the electrical location of a memory cell in a die.

5.1.5 *cell-logical-io-number* — number identifying the IO data which can be simultaneously electrically accessed within a memory device.

5.1.6 *cell-physical-address* — gives location of a memory cell in a die on two-dimensional plane.

5.1.7 *die* — a unit equivalent to one die on a wafer. Also known as Chip.

5.1.8 *electrical failure information* — failure information generated by test equipment; e.g. Bit Map Data, Bin Data and Analog Data.

5.1.9 *error class* — specifies the type of electrical failure configuration in a memory cell group. Bit-fail, line-fail and block-fail are examples.

5.1.10 *fail bit map data* — data representing memory cell electrical failure information according to its location information, in units of the die or wafer.

5.1.11 *inspection information* — inspection results for a wafer, indicating defect location and defect details obtained as the result of inspection used in wafer fabrication and the inspection process, such as appearance inspection, contaminant inspection, etc.

5.1.12 *test equipment* — equipment which tests the electrical characteristics and functions of semiconductor devices.

5.1.13 *test program name* — name of program used on test equipment when testing a die electrically.

5.1.14 *yield management system (YMS)* — a system that stores visual inspection data of geometrical defects on wafer and electrical test data of each die. The system processes those data statistically to discover correlation between them. Correlation leads to discovery the cause of failures and understanding of production situation in order to help improve yield. Defect data is standardized in SEMI E30.1 (ISEM).

6 Requirements

6.1 *Map Data Item* — All map data items appear in map data except those standardized in this document shall be specified in the common map data item document, SEMI G81.

6.2 *Map Data Format* — All map data formats appear in map data except those standardized in this document shall be specified in the common map data format document, SEMI G85.

7 General

7.1 *General File Format*

NOTE 2: A standard for Data representation is currently under development.

8 Description

8.1 The fail-bit map data item specifications defined in this document are shown in Table 2. The columns in this table are described in the following sections.

8.1.1 *Item Name Column* — The name by which the data item is referenced.

8.1.2 *Data Type Column* — The type of the data as defined in Table 1.

Table 1 Data Types and Sizes

Type	*Definition*
String	A string of ASCII characters from zero to "Size" characters in length.
Integer	A string of numeric characters (0..9) that represent an integer value.
Float	A string representing a floating point value in exponent format.

8.1.3 *Size Column* — See SEMI G81.

8.1.4 *Description Column* — See SEMI G81.

8.1.5 *Coordinate System Conventions* — See SEMI G81.

Table 2 Data Items for YMS Link

Item Name	*Data Type*	*Size*	*Description*
Comment	String	256	Comment Area. [Comment] = 'Engineering Test'
TestProgramName	String	256	Name of test program when test equipment obtains test data. [TestProgramName] = 'pro123a'
TestEquipmentId	String	256	Test equipment identification code. [TestEquipmentId] = 'tester1'

Item Name	*Data Type*	*Size*	*Description*
DieCoordinate	String	13	The die coordinates' origin location and coordinate axis direction for expressing location coordinates within a die. See Figure 2 and Table 7. [DieCoordinate] = 'TopLeftXY' [DieCoordinate] = 'BottomLeftXY' [DieCoordinate] = 'TopLeftYX' [DieCoordinate] = 'BottomLeftYX' [DieCoordinate] = 'TopRightXY' [DieCoordinate] = 'BottomRightXY' [DieCoordinate] = 'TopRightYX' [DieCoordinate] = 'BottomRightYX'
ErrorClassList	String	---	Error type category list, per error class definition. See Figure 2.
ErrorClassNumber	String	3	Error class number from 0 to 255.
ErrorQuality	String	16	Describes the quality of the specified [ErrorClassNumber]. This may be "bit-fail" or "line-fail" or some other value defined by an application.
ErrorDescription	String	256	A description of the specified [ErrorClassNumber], e.g. "100MHz".
FailBitDataTemplate	String	---	Defines the data items and order for expressing the fail bit error class data analyzed within a die by die address and coordinate values within the die. See Figure 2. Defines the items and order for outputting the following fail bit data. See Table 3.
FailBitData	String	---	The list of data expressing the fail bit error class. Data analyzed within a die in accordance with the fail bit data definition. Expresses it as die address and coordinates within a die. See Figure 2.
AnalogDataTemplate	String	---	Defines analog data items and order. These are measured values such as voltage, current, etc. on measuring semiconductor device's electrical characteristics by the test equipment, setting values which are test parameters when measuring, and limit values which are decision parameters if test results are pass or fail. See Table 4.
Invalid	String	14	Defined to indicate invalid data for Analog Data field.
AnalogData	String	---	List of analog data, per analog data definition. These are the semiconductor device's electrical characteristics measured by the test equipment.
ErrorTotal	Integer	10	Total number of electrical errors in the tested/analyzed substrate, by error class type.
TestDie	Integer	10	Total number of dies which were tested/analyzed on the substrate.

Table 3 Data Items for FailBitDataTemplate

Item Name	*Data Type*	*Size*	*Description*
Seq	Integer	10	Sequential number assigned in ascending order, starting with 1.
XAddress	Integer	5	X address of defective die with fail bit data.
YAddress	Integer	5	Y address of defective die with fail bit data.
XMin	Float	14	X minimum coordinate value of fail bit data's error range. Unit 'um'.
XMax	Float	14	X maximum coordinate value of fail bit data's error range. Unit 'um'.
YMin	Float	14	Y minimum coordinate value of fail bit data's error range. Unit 'um'.
YMax	Float	14	Y maximum coordinate value of fail bit data's error range. Unit 'um'.
XCenter	Float	14	X-direction center point coordinate value of fail bit data's error range. Unit 'um'.
YCenter	Float	14	Y-direction center point coordinate value of fail bit data's error range. Unit 'um'.
XSize	Float	14	X-direction size of fail bit data's error range. Unit 'um'.

Item Name	*Data Type*	*Size*	*Description*
YSize	Float	14	Y-direction size of fail bit data's error range. Unit 'um'.
Area	Float	14	Area of fail bit data's error range. Unit 'um'.
DefectCate	String	16	Category name of fail bit data.
XMinPhysic	Integer	10	X minimum cell physical address of fail bit data's error range.
XMaxPhysic	Integer	10	X maximum cell physical address of fail bit data's error range.
YMinPhysic	Integer	10	Y minimum cell physical address of fail bit data's error range.
YMaxPhysic	Integer	10	Y maximum cell physical address of fail bit data's error range.
IoNumber	Integer	10	Cell logical IO number of fail bit data's error range.
XMinLogic	Integer	10	X minimum cell logical address of fail bit data's error range.
XMaxLogic	Integer	10	X maximum cell logical address of fail bit data's error range.
YMinLogic	Integer	10	Y minimum cell logical address of fail bit data's error range.
YMaxLogic	Integer	10	Y maximum cell logical address of fail bit data's error range.

Table 4 Data Items for AnalogDataTemplate

Item Name	*Data Type*	*Size*	*Description*
XAddress	Integer	5	X address of die with analog data.
YAddress	Integer	5	Y address of die with analog data.
TestNumber	Integer	7	Test number corresponding to analog data.
Note	String	256	Note
Measure	Float	14	Analog data measured value.
MeasureUnit	String	4	Unit of above. See Tables 5 and 6.
Std	Float	14	Standard deviation, etc., for measuring analog data.
Parameter	Float	14	Analog data measurement parameter value.
ParameterUnit	String	4	Unit of above. See Tables 5 and 6.
UpperLimitData	Float	14	Upper limit of first type of parameter for judging analog data.
LowerLimitData	Float	14	Lower limit of first type of parameter for judging analog data.
TestResult	String	256	Analog data PASS/FAIL or decision result such as category type number, etc.

Table 5 Unit Table

Unit Code	*Description*
s	Seconds
Hz	Hertz
A	Amperes
V	Volts
Ohm	Ohms
W	Watts
F	Farad
H	Henry
%	Percent
dB	Decibel
lsb	Least Significant Bit
rad	Radian
deg	Degree

Table 6 Prefix Symbol Table

Prefix Symbol	*Multiplicative*
T	10^{12}
G	10^{9}
M	10^{6}
k	10^{3}
m	10^{-3}
u (micro)	10^{-6}
n	10^{-9}
p	10^{-12}

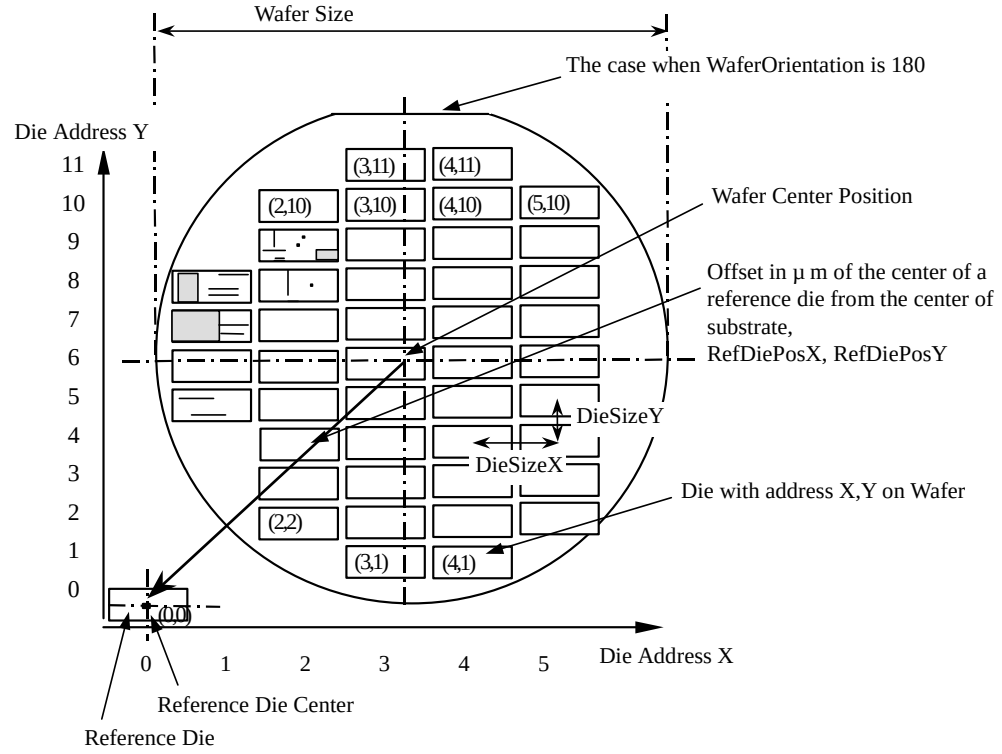
Example:

[MeasureUnit] = 'V' or 'mV'

[ParameterUnit] = 'A' or 'uA'

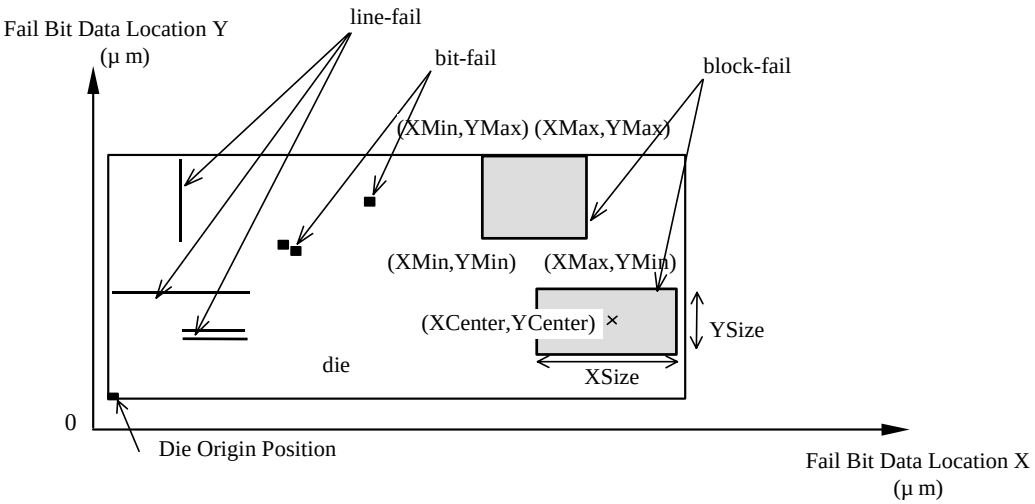
The units above are a subset of those in Section 9 ("Units of Measure") of SEMI E5 (SECS-II).

8.1.6 *Map Data Format of YMS Link* — This section is reserved for data format of above items.



(The case of BottomLeft)

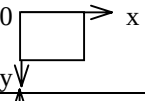
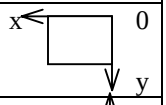
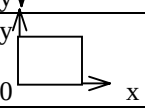
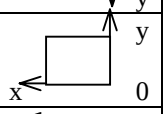
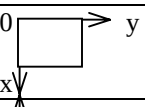
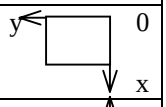
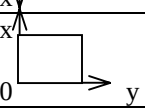
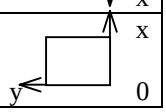
Figure 1
Wafer Coordinates



(The case of BottomLeftXY)

Figure 2
Die Coordinates

Table 7 Die Coordinates

Coordinate Information	*Meaning*	*Coordinate Information*	*Meaning*
TopLeftXY		TopRightXY	
BottomLeftXY		BottomRightXY	
TopLeftYX		TopRightYX	
BottomLeftYX		BottomRightYX	

9 Map Data Format for YMS Link Data

9.1 Because this standard defines extensional data for YMS Link Data map for generic test data map standardized by SEMI G81 and G85, data format of each item for YMS Link Data is defined independently as extra definition here.

9.2 It is very appropriate for this document to define such format with XML Schema, for SEMI G85 defines the generic test data map in XML. List 1 below defines YMS Link Data map format.

List 1. YMS Link Data Map Format

```
<xsd:schema xmlns:xsd=" http://www.w3.org/2001/XMLSchema"
xmlns:mdf=" http://www.semi.org/ATE" targetNamespace=" http://www.semi.org/YMSLDF" >

<xsd:import namespace=" http://www.semi.org/ATE" />
<! -- Commonized Part may be imported from schemaLocation " somehere/mdfSchema.xsd" -->

<xsd:complexType name=" ymsldfMap" >
 <xsd:extension base=" Map" >
  <xsd:all>
   <xsd:element name=" Comment" type=" string256Type" use=" optional" />
   <xsd:element name=" TestProgramName" type=" string256Type" use=" optional" />
   <xsd:element name=" TestEquipmentId" type=" string256Type" use=" optional" />
   <xsd:element name=" DieCoordinate" type=" dieCoordinateType" use=" optional" />
   <xsd:element name=" ErrorClassList" type=" errorClassListType" use=" optional" />
   <xsd:element name=" FailBitDataTemplate" type=" failBitDataTemplateType" use=" optional" />
   <xsd:element name=" FailBitData" type=" failBitDataType" use=" optional" />
   <xsd:element name=" AnalogDataTemplate" type=" analogDataTemplateType" use=" optional" />
   <xsd:element name=" Invalid" type=" string14Type" use=" optional" />
   <xsd:element name=" AnalogData" type=" failBitDataType" use=" optional" />
   <xsd:element name=" ErrorTotal" type=" unsignedInt10Type" use=" optional" />
   <xsd:element name=" TestDie" type=" unsignedInt10" use=" optional" />
  </xsd:all>
 </xsd:extension>
</xsd:complexType>

<! -- EnumSchemas -->
<xsd:simpleType name=" dieCoordinateType" >
 <xsd:restriction base=" xsd:string" >
```

```
    <xsd:enumeration value=" TopLeftXY" />
    <xsd:enumeration value=" BottomLeftXY" />
    <xsd:enumeration value=" TopLeftYX" />
    <xsd:enumeration value=" BottomLeftYX" />
    <xsd:enumeration value=" TopRightXY" />
    <xsd:enumeration value=" BottomRightXY" />
    <xsd:enumeration value=" TopRightYX" />
    <xsd:enumeration value=" BottomRightYX" />
  </xsd:restriction>
</xsd:simpleType>


<! -- Complicated Lists -->
<xsd:complexType name=" errorClassListType" >
  <xsd:element name=" ErrorClass"  type=" errorClassType"  minOccurs=" 1"  maxOccurs=" 255" />
</xsd:complexType>

<xsd:complexType name=" errorClassType" >
  <xsd:sequence>
    <xsd:element name=" ErrorClassNumber"  type=" charIntegerType" />
    <xsd:element name=" ErrorQuality"  type=" string16Type"  use=" optional" />
    <xsd:element name=" ErrorDescription"  type=" string256Type"  use=" optional" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name=" failBitDataType" >
  <xsd:sequence>
    <xsd:element name=" failBitDataUnit"  type=" failBitDataUnitType"  minOccurs=" 1"  maxOccurs=" unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name=" failBitDataUnitType" >
  <xsd:list itemType=" failBitDataParticleType"  />
</xsd:simpleType>

<xsd:simpleType name=" failBitDataParticleType" >
  <xsd:union memberTypes=" Seq XAddress YAddress XMin XMax YMin YMax XCenter YCenter XSize YSize
Area DefectCate XMinPhysic XMaxPhysic YMinPhysic YMaxPhysic IoNumber XMinLogic XMaxLogic
YMinLogic YMaxLogic"  />
</xsd:simpleType>

<xsd:simpleType name=" Seq"  type=" unsignedInt10Type" />
<xsd:simpleType name=" XAddress"  type=" int5Type"  />
<xsd:simpleType name=" YAddress"  type=" int5Type"  />
<xsd:simpleType name=" XMin"  type=" float14Type"  />
<xsd:simpleType name=" XMax"  type=" float14Type"  />
<xsd:simpleType name=" YMin"  type=" float14Type"  />
<xsd:simpleType name=" YMax"  type=" float14Type"  />
<xsd:simpleType name=" XCenter"  type=" float14Type"  />
<xsd:simpleType name=" YCenter"  type=" float14Type"  />
<xsd:simpleType name=" XSize"  type=" float14Type"  />
<xsd:simpleType name=" YSize"  type=" float14Type"  />
<xsd:simpleType name=" Area"  type=" float14Type"  />
<xsd:simpleType name=" DefectCate"  type=" string16Type"  />
```

```
<xsd:simpleType name=" XMinPhysic"  type=" unsignedInt10Type"  />
<xsd:simpleType name=" XMaxPhysic"  type=" unsignedInt10Type"  />
<xsd:simpleType name=" YMinPhysic"  type=" unsignedInt10Type"  />
<xsd:simpleType name=" YMaxPhysic"  type=" unsignedInt10Type"  />
<xsd:simpleType name=" IoNumber"type=" unsignedInt10Type"  />
<xsd:simpleType name=" XMinLogic"  type=" unsignedInt10Type"  />
<xsd:simpleType name=" XMaxLogic"type=" unsignedInt10Type"  />
<xsd:simpleType name=" YMinLogic"  type=" unsignedInt10Type"  />
<xsd:simpleType name=" YMaxLogic"type=" unsignedInt10Type"  />

<xsd:complexType name=" analogDataType" >
 <xsd:sequence>
  <xsd:element name=" analogDataUnit"  type=" analogDataUnitType"  minOccurs=" 1"  maxOccurs=" unbounded" />
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name=" analogDataUnitType" >
 <xsd:list itemType=" analogDataParticleType"  />
</xsd:simpleType>

<xsd:simpleType name=" analogDataParticleType" >
 <xsd:union memberTypes=" XAddress YAddress TestNumber Note Measure MeasureUnit Std Parameter
ParameterUnit UpperLimitData LowerLimitData TestResult"  />
</xsd:simpleType>

<xsd:simpleType name=" TestNumber"  type=" long7Type"  />
<xsd:simpleType name=" Note"  type=" string256Type"  />
<xsd:simpleType name=" Mesure"  type=" float14Type"  />
<xsd:simpleType name=" MeasureUnit"  type=" analogUnitType"  />
<xsd:simpleType name=" Std"  type=" float14Type"  />
<xsd:simpleType name=" Parameter"  type=" float14Type"  />
<xsd:simpleType name=" ParameterUnit"  type=" analogUnitType"  />
<xsd:simpleType name=" UpperLimit"  type=" float14Type"  />
<xsd:simpleType name=" LowerLimit"  type=" float14Type"  />
<xsd:simpleType name=" TestResult"  type=" string256Type"  />


<xsd:simpleType name=" analogUnitType" >
 <xsd:restriction base=" xsd:string" >
  <maxLength Value=" 4"  fixed=" true" />
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" string14Type" >
 <xsd:restriction base=" xsd:string" >
  <maxLength Value=" 14"  fixed=" true" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" string16Type" >
 <xsd:restriction base=" xsd:string" >
  <maxLength Value=" 16"  fixed=" true" />
 </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name=" string256Type" >
  <xsd:restriction base=" xsd:string" >
    <maxLength Value=" 256"  fixed=" true" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" charIntegerType" >
  <xsd:restriction base=" xsd:unsignedShort" >
    <xsd:minInclusive value=" 0" />
    <xsd:maxInclusive value=" 255" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" unsignedInt10Type" >
  <xsd:restriction base=" xsd:unsignedLong" >
    <xsd:totalDigits value=" 10" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" int5Type" >
  <xsd:restriction base=" xsd:int" >
    <xsd:totalDigits value=" 5" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" long7Type" >
  <xsd:restriction base=" xsd:long" >
    <xsd:totalDigits value=" 7" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name=" float14Type" >
  <xsd:restriction base=" xsd:float" >
    <xsd:pattern value=" [1-9].\ d{ 9} (+| -)\ d{ 2} " />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name=" failBitDataTemplateType" >
  <xsd:list itemType=" failBitDataItemNameType" />
</xsd:complexType>

<xsd:simpleType name=" failBitDataItemNameType" >
  <xsd:restriction base=" xsd:string" >
    <xsd:enumeration value=" Seq" />
    <xsd:enumeration value=" XAddress" />
    <xsd:enumeration value=" YAddress" />
    <xsd:enumeration value=" XMin" />
    <xsd:enumeration value=" XMax" />
    <xsd:enumeration value=" YMin" />
    <xsd:enumeration value=" YMax" />
    <xsd:enumeration value=" XCenter" />
    <xsd:enumeration value=" YCenter" />
    <xsd:enumeration value=" XSize" />
```

```
    <xsd:enumeration value=" YSize" />
    <xsd:enumeration value=" Area" />
    <xsd:enumeration value=" DefectCate" />
    <xsd:enumeration value=" XMinPhysic" />
    <xsd:enumeration value=" XMaxPhysic" />
    <xsd:enumeration value=" YMinPhysic" />
    <xsd:enumeration value=" YMaxPhysic" />
    <xsd:enumeration value=" IoNumber" />
    <xsd:enumeration value=" XMinLogic" />
    <xsd:enumeration value=" XMaxLogic" />
    <xsd:enumeration value=" YMinLogic" />
    <xsd:enumeration value=" YMaxLogic" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name=" analogDataTemplateType" >
  <xsd:list itemType=" analogDataItemNameType" />
</xsd:complexType>

<xsd:simpleType name=" analogDataItemNameType" >
  <xsd:restriction base=" xsd:string" >
    <xsd:enumeration value=" XAddress" />
    <xsd:enumeration value=" YAddress" />
    <xsd:enumeration value=" TestNumber" />
    <xsd:enumeration value=" Note" />
    <xsd:enumeration value=" Measure" />
    <xsd:enumeration value=" MeasureUnit" />
    <xsd:enumeration value=" Std" />
    <xsd:enumeration value=" Parameter" />
    <xsd:enumeration value=" ParameterUnit" />
    <xsd:enumeration value=" UpperLimitData" />
    <xsd:enumeration value=" LowerLimitData" />
    <xsd:enumeration value=" TestResult" />
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

RELATED INFORMATION 1

NOTE: This related information is not an official part of SEMI E107, but was approved for publication by full letter ballot procedure on July 19, 2002.

R1-1 Example of YMS Link Data

```
<? xml version=”1.0”? >
<ymsldfMap xmlns:semi=”http://www.semi.org”  WaferId=“wafer1234”  FormatRevision=“2.00"  >
 <Device  ProductID=”product1234” LotId=”lot1234” Orientation=”180”
  DeviceSizeX=” 4.8987000000e+ 03” DeviceSizeY=” 8.7395000000e+ 03”
  Rows=”4” Columns=”1” BinType=”Decimal” NullBin=”255” CreateDate=” 03-18-2000 15:07:50”
  Status=” prerepair_map”>
  <ReferenceDevice RefDevicePosX=”-2030.55” RefDevicePosY=”1427.98”/>
  <Bin BinCode=”001” BinCount=”35” BinQuality=”Pass”/>
  <Bin BinCode=”002” BinCount=”4” BinQuality=”Fail”/>
  <Bin BinCode=”003” BinCount=”1” BinQuality=”Fail”/>
  <Data>
   <Row><! [CDATA[255255255255255255255255255]]></Row>
   <Row><! [CDATA[2552552552552552550010020030 02]]></Row>
   <Row><! [CDATA[255255255255255255255255255]]></Row>
  </Data>
 </Device>
 <Comment>Engineering Test</Comment>
 <TestProgramName>pro123a</TestProgramName>
 <TestEquipmentId>tester1</TestEquipmentId>
 <DieCoordinate>BottomLeftXY</DieCoordinate>
 <ErrorClassList>
  <ErrorClass>
   <ErrorClassName>0</ErrorClassName>
   <ErrorClassQuality>bit-fail</ErrorClassQuality>
  </ErrorClass>
  <ErrorClass>
   <ErrorClassName>1</ErrorClassName>
   <ErrorClassQuality>line-fail</ErrorClassQuality>
  </ErrorClass>
  <ErrorClass>
   <ErrorClassName>2</ErrorClassName>
   <ErrorClassQuality>block-fail</ErrorClassQuality>
  </ErrorClass>
 </ErrorClassList>
 <FailBitDataTemplate>Seq XAddress YAddress XMin XMax YMin YMax XCenter YCenter XSize YSize Area
DefectCate</FailBitDataTemplate>
 <FailBitData>
  <FailBitDataUnit>1 8 18 0.12345000e+ 03 0.91234000e+ 03 1.06367000e+ 03 7.27391000e+ 03 0.51789500e+ 03
4.16879000e+ 03 0.78889000e+ 03 6.21024000e+ 03 8.89919623e+ 06 block-fail</FailBitDataUnit>
  <FailBitDataUnit>2 10 18 3.03120000e+ 03 3.03210000e+ 03 1.58330000e+ 03 1.58580000e+ 03 3.03165000e+ 03
1.58455000e+ 03 9.00000000e-01 2.50000000e+ 00 2.25000000e+ 00 bit-fail</FailBitDataUnit>
 </FailBitData>
 <AnalogDataTemplate>XAddress YAddress TestNumber Note Measure MeasureUnit TestNumber Comment
Measure MeasureUnit TestResult TestNumber Comment Measure MeasureUnit TestResult</AnalogDataTemplate>
 <Invalid>99999</Invalid>
 <AnalogData>
```

<AnalogDataUnit>8 18 100 CONTACT –0.600 V 200 IDD 125.0 mA PASS 300 IDDS 100.0 uA PASS</AnalogDataUnit>
<AnalogDataUnit>9 18 100 CONTACT –0.567 V 200 IDD 120.2 mA PASS 300 IDDS 321.0 uA FAIL</AnalogDataUnit>
<AnalogDataUnit>10 18 100 CONTACT –0.612 V 200 IDD 123.4 mA PASS 300 IDDS 99.4 uA PASS</AnalogDataUnit>
<AnalogDataUnit>11 18 100 CONTACT 99999 V 200 IDD 99999 mA PASS 300 IDDS 99.4 uA PASS</AnalogDataUnit>
</AnalogData>
<ErrorTotal>234</ErrorTotal>
<TestDie>3</TestDie>
<Map>

R1-2 Example Datafile

```
Map = {
FormatRevision = “2.00"
CreateDate = “03-18-2000 15:07:50”
ProductId = " product1234"
LotId = " lot1234"
WaferId = “wafer1234”
Status = “prerepair_map”
Comment = " Engineering Test"
TestProgramName = " pro123a"
TestEquipmentId = “tester1”
WaferOrientation = 180
WaferSize = 200
DieSizeX = 4.8987000000e+ 03
DieSizeY = 8.7395000000e+ 03
ReferenceDieList = {
        PosX = -2030.55 PosY = 1427.98
}
WaferCoordinate = “TopLeft”
DieCoordinate = “BottomLeftXY”
MapType = “Die”
BinType = “Decimal”
BinList = {
        Code = 001 Count = 35 Quality = “Pass”
        Code = 002 Count = 4 Quality = “Fail”
        Code = 003 Count = 1 Quality = “Fail”
}
Data = {
        007 018 001
        008 018 002
        009 018 003
        010 018 002
}
ErrorClassList = {
        Class = “0” Quality = " bit-fail"
        Class = “1” Quality = " line-fail"
        Class = “2” Quality = " block-fail"
}
FailBitDataTemplate = Seq XAddress YAddress XMin XMax YMin YMax XCenter YCenter XSize YSize Area DefectCate
FailBitData = {
```

```
        1 8 18 0.12345000e+ 03 0.91234000e+ 03 1.06367000e+ 03 7.27391000e+ 03
        0.51789500e+ 03 4.16879000e+ 03 0.78889000e+ 03 6.21024000e+ 03    4.89919623e+ 06 " block-fail"
        2 10 18 3.03120000e+ 03 3.03210000e+ 03 1.58330000e+ 03 1.58580000e+ 03
        3.03165000e+ 03 1.58455000e+ 03 9.00000000e-01 2.50000000e+ 00    2.25000000e+ 00 " bit-fail"
}
AnalogDataTemplate = XAddress YAddress TestNumber Note Measure
MeasureUnit TestNumber Comment Measure MeasureUnit TestResult
TestNumber Comment Measure MeasureUnit TestResult
Invalid = “99999”
AnalogData = {
        8 18 100 “CONTACT” –0.600 “V” 200 “IDD” 125.0 “mA” “PASS” 300 “IDDS”          100.0 “µ A”
“PASS”
        9 18 100 “CONTACT” –0.567 “V” 200 “IDD” 120.2 ”mA” “PASS” 300 “IDDS”          321.0 “µ A”
“FAIL”
        10 18 100 “CONTACT” –0.612 “V” 200 “IDD” 123.4 “mA” “PASS” 300 “IDDS”         99.4 “µ A”
“PASS”
        11 18 100 “CONTACT”  99999  “V” 200 “IDD”  99999  “mA” “PASS” 300 “IDDS”      99.4 “µ A”
“PASS”
}
ErrorTotal = 234
TestDie = 3
}
```

Table R1-1 Items and Attributes Used Yield Management specific part

Name	*Element or Attribute*	*Define In*	*Mandatory and Optional*	*Relation Items*
Comment	Attribute	Yms Map Data Items	Optional	---
TestProgramName	Attribute	Yms Map Data Items	Optional	---
TestEquipmentId	Attribute	Yms Map Data Items	Optional	---
DieCoordinate	Attribute	Yms Map Data Items	Optional	WaferLocation
ErrorClassList	Element	Yms Map Data Format	Optional	---
ErrorClassNumber	Attribute	Yms Map Data Items	Optional	ErrorClassList
ErrorQuality	Attribute	Yms Map Data Items	Optional	ErrorClassList
ErrorDescription	Attribute	Yms Map Data Items	Optional	ErrorClassList
FailBitDataTemplate	Element	Yms Map Data Format	Optional	---
FailBitData	Element	Yms Map Data Format	Optional	FailBitDataTemplate
AnalogDataTemplate	Element	Yms Map Data Format	Optional	---
Invalid	Attribute	Yms Map Data Items	Optional	---
AnalogData	Element	Yms Map Data Format	Optional	AnalogDataTemplate
ErrorTotal	Attribute	Yms Map Data Items	Optional	ErrorClassList
TestDie	Attribute	Yms Map Data Items	Optional	---

Table R1-2 Attributes Used for FailBitData

Item Name	*Item or Attribute*	*Define In*	*Mandatory and Optional*	*Relation Items*
Seq	Attribute	Map Data Items	Mandatory	---
XAddress	Attribute	Map Data Items	Mandatory	WaferLocation
YAddress	Attribute	Map Data Items	Mandatory	WaferLocation
XMin	Attribute	Map Data Items	Mandatory	DieCoordinate
XMax	Attribute	Map Data Items	Mandatory	DieCoordinate
YMin	Attribute	Map Data Items	Mandatory	DieCoordinate
YMax	Attribute	Map Data Items	Mandatory	DieCoordinate
XCenter	Attribute	Map Data Items	Mandatory	DieCoordinate
YCenter	Attribute	Map Data Items	Mandatory	DieCoordinate
XSize	Attribute	Map Data Items	Mandatory	DieCoordinate
YSize	Attribute	Map Data Items	Mandatory	DieCoordinate
Area	Attribute	Map Data Items	Mandatory	DieCoordinate
DefectCate	Attribute	Map Data Items	Mandatory	ErrorClassList
XMinPhysic	Attribute	Map Data Items	Mandatory	DieCoordinate
XMaxPhysic	Attribute	Map Data Items	Mandatory	DieCoordinate
YMinPhysic	Attribute	Map Data Items	Mandatory	DieCoordinate
YMaxPhysic	Attribute	Map Data Items	Mandatory	DieCoordinate
IoNumber	Attribute	Map Data Items	Mandatory	DieCoordinate
XMinLogic	Attribute	Map Data Items	Mandatory	DieCoordinate
XMaxLogic	Attribute	Map Data Items	Mandatory	DieCoordinate
YMinLogic	Attribute	Map Data Items	Mandatory	DieCoordinate
YMaxLogic	Attribute	Map Data Items	Mandatory	DieCoordinate

Table R1-3 Attributes Used for AnalogData

Item Name	*Item or Attribute*	*Define In*	*Mandatory and Optional*	*Relation Items*
XAddress	Attribute	Map Data Items	Mandatory	WaferLocation
YAddress	Attribute	Map Data Items	Mandatory	WaferLocation
TestNumber	Attribute	Map Data Items	Mandatory	---
Note	Attribute	Map Data Items	Optional	---
Measure	Attribute	Map Data Items	Optional	---
MeasureUnit	Attribute	Map Data Items	Optional	---
Std	Attribute	Map Data Items	Optional	---
Parameter	Attribute	Map Data Items	Optional	---
ParameterUnit	Attribute	Map Data Items	Optional	---
UpperLimitData	Attribute	Map Data Items	Optional	---
LowerLimitData	Attribute	Map Data Items	Optional	---
TestResult	Attribute	Map Data Items	Optional	---

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacture' s instructions, product labels, product data sheets, and other relevant literature, respecting any materials or equipment mentioned herein. These standards are subject to change without notice.

By publications of this standard, Semiconductor Equipment and Materials International (SEMI) takes no position respecting the validity of any patent rights or copyrights asserted in connection with any items mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights are entirely their own responsibility.

SEMI E109-0305
SPECIFICATION FOR RETICLE AND POD MANAGEMENT (RPMS)

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on December 10, 2004. Initially available at www.semi.org February 2005; to be published March 2005. Originally published July 2001; previously published November 2004.

1 Purpose

1.1 This document provides standardized behavior for lithography, reticle inspection, and bare reticle stocker equipment. It also provides for standardized communication with lithography, reticle inspection, and bare reticle stocker equipment. This includes the coordination, execution, and completion of automated and manual reticle pod transfers to and from the equipment, transfer of reticles to and from the reticle pods, movement of the reticles within the equipment, identification and verification of both reticle pods and reticles, inspection and qualification of reticles, and other relevant information such as tracking reticle usage.

2 Scope

2.1 This is a standard that covers host and equipment communication for equipment that handles reticles. This only includes equipment that handles reticles both in and outside of a reticle pod.

2.2 The scope of this document is to define standards that facilitate the host's knowledge and role in automated and manual reticle pod transfers, reticle transfers to and from the reticle pod, internal reticle movement, identification and verification of ReticleID, inspection and qualification of reticles, and tracking reticle usage. Specifically, this document provides state models and scenarios that define the host interaction with the equipment for the following:

- Reticle pod transfer between AMHS vehicles and production equipment, bare reticle stockers, and reticle inspection equipment reticle load ports.
- Reticle transfers to/from lithography production equipment internal reticle library space.
- Equipment and reticle load port access mode switching.
- Reticle Pod to load port association.
- Reticle Pod ID verification and Reticle Pod slot map verification.

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 Limitations

3.1 This standard applies to semiconductor equipment that handles reticles and reticle carriers. This standard is intended to be used for lithography production, bare reticle storage, and reticle inspection equipment. It may or may not be applied to other types of equipment.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E15 — Specification for Tool Load Port

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

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

SEMI E41 — Exception Management (EM) Standard

SEMI E53 — Event Reporting

SEMI E84 — Specification for Enhanced Carrier Handoff Parallel I/O interface

SEMI E99 — The Carrier ID Reader/Writer Functional Standard: Specifcation of Concepts, Behavior, and Services

SEMI E100 — Specification for a Reticle SMIF Pod (RSP) Used to Transport and Store 6 inch or 230 mm Reticles

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *AGT* — Automated Guided Transport

5.1.2 *AGV*— Automated Guided Vehicle

5.1.3 *AMHS* — Automated Material Handling System

5.1.4 *ARHS* — Automated Reticle Handling System

5.1.5 *GEM* — Generic Equipment Model

5.1.6 *IRL* — Internal Reticle Library

5.1.7 *OHT* — Overhead Hoist Transport

5.1.8 *PGV* — Person Guided Vehicle

5.1.9 *PIO* — Parallel Input/Output Interface

5.1.10 *RGT* — Rail Guided Transport

5.1.11 *RGV* — Rail Guided Vehicle

5.1.12 *RSP* – Reticle SMIF Pod

5.2 *Definitions*

5.2.1 *Automated Material Handling System* — an automated system to store and transport materials within the factory.

5.2.2 *Automated Reticle Handling System* — a specific type of Automated Material Handling System to store and transport reticles and reticle pods within the factory.

5.2.3 *automation* — the degree to which activities of machines or production systems are self-acting. In this standard automation provides methods that will reduce the amount of operator intervention required.

5.2.4 *buffer* — a set of one or more locations for holding reticles at/inside the production equipment.

5.2.5 *collection event* — a collection event is an event (or grouping of related events) on the equipment that is considered to be significant to the host.

5.2.6 *content map* — ordered list of reticle identifiers corresponding to slot 1,2,3…n. (Note that this is redundant with the definition in Table 6).

5.2.7 *host* — the factory computer system or an intermediate system that represents the factory and the user to the equipment.

5.2.8 *Internal Pod Buffer* — storage area for reticle pod that is internal to the equipment.

5.2.9 *internal reticle library* — a set of locations within the equipment to store reticles. These locations exclude load ports.

5.2.10 *kitting* — the act of placing a group of 1 or more reticles in a reticle pod for removal from a bare reticle stocker. This is accomplished via a ReticleTransferJob that specifies one or more reticles for removal from the stocker, or by one or more MoveReticle services that specify a destination that is a Pod Location.

5.2.11 *load* — the operation of placing a pod on a load port.

5.2.12 *load port* — the interface location on the equipment where pods are loaded and unloaded.

5.2.13 *Multi-Reticle Pod* — TBD

5.2.14 *object instantiation* — the act of storing of information related to a physical or logical entity so that it can be recalled on demand based on its public identifier.

5.2.15 *on-line equipment* — equipment that is connected to, and able to communicate fully with, the host.

5.2.16 *pod* — in this document pod refers to an RSP or a Multi Reticle SMIF Pod.

5.2.17 *PodID* — a readable and unique identifier for the pod.

5.2.18 *PodID read* — the process of the equipment reading the PodID from the carrier.

5.2.19 *PodID tag (tag, ID tag)* — a physical device for storing PodID and other information. There are two basic types of tags, read-only tags and read/write tags. [SEMI E99]

5.2.20 *process equipment* — equipment used to produce product, such as semiconductor devices. This excludes metrology and material handling equipment.

5.2.21 *production equipment* — equipment used to produce product, such as semiconductor devices, including substrate sorting, process, and metrology equipment and excluding material handling equipment.

5.2.22 *properties* — a set of name value pairs assigned to an object or used in a service message to include additional information about the object (i.e., pod, port, etc.).

5.2.23 *re-initialization* — a process where production equipment is either powered off then on or when some kind of hardware or software reset is initiated to cause the equipment to reset and possibly reload its software. On production equipment that contains some kind of mass storage device this can also be called a “reboot”.

5.2.24 *read position* — any position where the tag on a pod can be read.

5.2.25 *reticle* — a mask that contains the patterns to be reproduced on a substrate; the image may be equal to or larger than the final projected image.

5.2.26 *Reticle SMIF Pod (RSP)* — minienvironment compatible carrier capable of holding one 6 inch or one 230 mm reticle in a horizontal orientation during transport and storage and is compatible with a Standard Mechanical Interface (SMIF) per SEMI E19.4.

5.2.27 *single communication connection* — exactly one physical connection using exactly one logical session and a standard set of messages.

5.2.28 *slot map* — the information that relates which slots in a reticle pod hold reticles, both correctly and incorrectly.

5.2.29 *slot map read* — the process of the equipment reading the slot map for substrate position and placement within the pod.

5.2.30 *standard message set* — messages conforming to standard message specifications.

5.2.31 *transfer unit* — maximum number of pods allowed in a specific transfer service:

- AA is the maximum number of pods allowed for acquisition at the transfer source.
- BB is the maximum number of pods allowed for deposit at the transfer destination.
- CC is the maximum number of carrier pods allowed for transfer in one transport vehicle.

The transfer unit is the minimum of AA, BB, and CC.

NOTE 1: At the time this document was originally written, December 2000, transfer unit for reticle transfer to process equipment is expected to be equal to one. Transfer unit to other equipment may be greater than one.

5.2.32 *unload* — the operation of removing a pod from a load port.

5.2.33 *write position* — any position on a load port or in an internal buffer from which the tag on a pod can be written to. This position may vary on any particular equipment depending on the write technology selected by the end user. The read position and the write position may or may not be the same position.

6 Requirements

6.1 Reticle and Pod Management compliant equipment is required to provide certain capabilities defined by other standards: accessibility to status information, event reporting, alarm management, and equipment control. These requirements shall be satisfied through compliance to the following sets of standards:

6.2 *Generic Equipment Model Standard (GEM) SEMI E30*

- Event Notification
- Status Data Collection
- Equipment Constants
- Alarm Management
- Equipment Control
- Remote Control
- Error Messages
- Dynamic Event Report Configuration

6.3 *Object-Based Standards*

- Object Services Standard (SEMI E39)
- Event Reporting (SEMI E53)
- Exception Management (SEMI E41)

7 Conventions

7.1 *Objects*

7.1.1 Whenever the equipment is required to know about specific kinds of entities, and required to manage information concerning these entities, it is useful to treat these entities as objects that comply with the basic requirements of SEMI E39 (OSS). This is especially true whenever there are a large number of objects of a given type or when the entities are transient rather than permanent. In both cases, it is difficult to describe a general way for the host and equipment to specify which particular entity is referenced and to get information related only to a specific one out of many.

7.1.2 By defining these entities as objects that comply with SEMI E39, it is only necessary for the host to specify the type of object and its specific identifier in order to inquire about one or more properties of the specific entity of interest.

7.1.3 *Object Properties*

7.1.3.1 A property (attribute) is information about an individual object that is presented as a name/value pair. The name is a formally reserved text string that represents the property, and the value is the current setting for that property.

7.1.3.2 Properties shall be accessible to the host via the service GetAttr and SetAttr for the Reticle Pod object, Reticle object, and Reticle location object:

- get the list of IDs for the current reticle pods at the equipment, and
- get the specified properties for one or more individual reticle pods.

7.1.4 *Rules for Object Properties*

- Attributes with RO access can not be changed using SetAttr service as defined in OSS.
- Attributes with RW access can be changed using SetAttr service as defined in OSS.

- Additional attributes may be specified by the user or the equipment supplier by using an attribute name starting with "UD" (User Defined). Care should be taken to ensure the name of the attribute is unique.

7.1.5 *Object Attribute Table*

7.1.5.1 The object attribute table is used to list all the attributes related to the defined object as shown below the access is defined as Read only (RO) or Read/Write (RW). The REQD column is used to specify whether the attribute is required for implementation. Finally, the Form column is used to specify the format of that particular attribute.

Table 1 Object Attribute Table

Attribute Name	*Definition*	*Access*	*Reqd*	*Form*
ObjType	Object type	RO	Y	Text = "Carrier"

7.2 *State Model Methodology*

7.2.1 A state model has three elements: definitions of each state and sub-state, a diagram of the states and the transitions between states, and a state transition table. The diagram of the state model uses the Harel State Chart notation. An overview of this notation is presented in an Appendix of SEMI E30. The definition of this notation is presented in Science of Computer Programming 8, "Statecharts: A Visual Formalism for Complex Systems", by D. Harel, 1987. [1]

7.2.2 *State Model Requirements*

7.2.2.1 The state models included in this standard are a requirement for RPMS compliance. A state model consists of a state model diagram, state definitions, and a state transition table. All state transitions in this standard, unless otherwise specified, shall correspond to collection events.

7.2.2.2 A state model represents the host's view of the equipment, and does not necessarily describe the internal equipment operation. All Reticle and Pod Management state model transitions shall be mapped sequentially into the appropriate internal equipment collection events that satisfy the requirements of those transitions. In certain implementations, the equipment may enter a state and have already satisfied all of the conditions required by the RPMS state model for transition to another state. In this case, the equipment makes the required transition without any additional actions in this situation.

7.2.2.3 Some equipment may need to include additional sub-states other than those in this standard. Additional sub-states may be added, but shall not change the Reticle and Pod Management defined state transitions. All expected transitions between Reticle and Pod Management states shall occur.

7.2.2.4 Transition tables are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. A transition table contains columns for Transition number, Previous State, Trigger, New State, Actions, and Comments. The "trigger" (column 3) for the transition occurs while in the "previous" state. The "actions" (column 5) includes a combination of:

- Actions taken upon exit of the previous state,
- Actions taken upon entry of the new state, and
- Actions taken which are most closely associated with the transition.

Table 2 State Transition Table

Num	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*

1 Elsevier Science, P. O. Box 945, New York, NY 10159-0945, http://www.elsevier.nl/homepage/browse.html

7.3 *Services*

7.3.1 Services are functions or methods that may be provided by either the equipment or the host. A service message may be either a request message, which always requires a response, or a notification message that does not require a response.

7.3.2 *Service Message Description*

7.3.2.1 A service message description table defines the parameters used in a service, as shown in the following table:

Table 3 Service Message Description Table

Service Name	*Type*	*Description*

[#1] Type can be either "N" = Notification or "R" = Request & Response.

7.3.2.2 Notification type messages are initiated by the service provider (e.g., the equipment) and the provider does not expect to get a response from the service user. Request messages are initiated by a service user (e.g., the host). Request messages ask for data or an activity from the provider. Request messages expect a specific response message (no presumption on the message content).

7.3.3 *Service Message Parameter Definition*

7.3.3.1 A service parameter dictionary table defines the description, range, and type for parameters used by services, as shown in the following table:

Table 4 Service Message Parameter Definition Table

Parameter Name	*Form*	*Description*

[#1] A row is provided in the table for each parameter used on a service.

7.3.4 *Service Message Definition*

7.3.4.1 A service message description table defines the parameters used in a service message. It also describes each message and its cause/effect to the equipment. The columns labeled Req/Ind and Resp/Conf link the parameters to the direction of the message.

Service Parameter	*Req/Ind*	*Resp/Conf*	*Description*

7.3.4.2 The columns labeled Req/Ind and Rsp/Conf link the parameters to the direction of the message. The message sent by the initiator is called the "Request". The receiver terms this message the "Indication". The receiver may then send a "Response", which the original sender terms the "Confirmation".

7.3.4.3 The following codes appear in the Req/Ind and Rsp/Conf columns and are used in the definition of the parameters (e.g., how each parameter is used in each direction):

"M"	Mandatory Parameter – must be given a valid value.
"C"	Conditional Parameter – may be defined in some circumstances and undefined in others. Whether a value is given may be completely optional or may depend on the values of other parameters.
"U"	User-Defined Parameter.
"-"	The parameter is not used.
"="	(for response only) Indicates that the value of this parameter in the response must match that in the primary (if defined).

7.4 *Alarm Requirements Definition*

7.4.1 An alarm requirements definition table defines the specific set of alarms required by RPMS. The table is divided up by equipment configuration, and then by alarm. The danger and affected columns are marked with "X" characters to show each alarm and its possible impact to operators, equipment, and material. The table format is shown in the following example:

Equipment		*Danger*		*Affected*		
Configuration	*Alarm Text*	*Potential*	*Imminent*	*Operator*	*Equipment*	*Material*
Configuration 1	Alarm 1	X		X	X	X
	Alarm 2		X			X
Configuration 2	Alarm 3	X		X	X	
	Alarm 4	X			X	X

8 Overview

8.1 The Reticle and Pod Management standard defines the behavior, data, and services required for equipment supporting automated reticle pod transfer, reticle pod management, and reticle management. This document provides a standard interface for host/equipment communications regarding the transfer of reticle pods, reticle pod identification and verification, transfer of reticles to and from equipment, reticle identification and verification, reticle inspection and qualification, and tracking of reticles. The standardized reticle pod transfer host interface include transfers to and from the external reticle load ports and internal reticle pod locations, the standardized reticle host interface include transfers to and from the reticle pod, transfers to and from the internal reticle library positions, and reticle identification, inspection, verification, qualification and tracking of reticles.

8.2 *Single Connection Requirement*

8.2.1 The expectation of the production and storage equipment supplier is that this standard be implemented in conjunction with the GEM interface to their production equipment and without the use of a separate communication connection.

9 Load Port

9.1 A reticle pod load port (port) is used by the factory to load and unload reticle pods to and from Lithography, reticle inspection, and bare reticle storage equipment. A reticle pod load port may be used as a reticle pod input load port, a reticle pod output load port, or as a reticle pod input/output load port, depending upon equipment type, configuration and/or factory practices. This classification may be fixed or it may be programmable by the user. A reticle pod load port is generally designed to handle one specific carrier type, reticle SMIF pods (RSP).

9.1.1 The equipment supplier is free to implement Load Ports as objects, but this is not a requirement for compliance to this standard.

9.2 *Load Port Numbering*

9.2.1 The reticle pod load port number shall be assigned incrementally from the bottom left to bottom right, then top left to top right when facing the reticle load ports. The numbering system should start with 101 to differentiate from FOUP load ports. The reticle pod load port-numbering requirement is to provide a common reference base to external entities, such as humans.

9.3 *Reticle Pod Slot Numbering*

9.3.1 The slot numbers for a reticle pod shall be assigned incrementally from the bottom, starting with "1."

9.4 *Reticle Pod Load Port Resource Sharing*

9.4.1 A model of a reticle pod load port must account for any mechanical assemblies that are either active during reticle pod transfer or are capable of interacting with the transfer. The reticle pod load port is responsible for such mechanisms when the reticle pod load port is in the TRANSFER READY state. If these mechanisms are shared with other reticle pod load ports, then the sharing must be coordinated.

9.5 *Reticle Pod Load Port Transfer State Model*

9.5.1 The purpose of the Reticle Pod Load Port Transfer State Model is to define the host view of a reticle pod transfer, which includes the host interactions with the equipment necessary to transfer reticle pods to and from equipment reticle pod load ports. Each reticle pod load port on the equipment shall maintain an independent instance of this state model.

9.5.2 *Reticle Pod Load Port Transfer State Model Diagram*

9.5.2.1 Figure 1 is the diagram for the Reticle Pod Load Port Transfer State Model.

9.5.3 *Reticle Pod Load Port Transfer State Definitions*

9.5.3.1 *RETICLE POD LOAD PORT TRANSFER* — The super state for the IN SERVICE and OUT OF SERVICE states.

9.5.3.2 *OUT OF SERVICE* — Transfer to/from this reticle pod load port is disabled. A transition to IN SERVICE is required to continue using this reticle pod load port for transfers.

9.5.3.3 *IN SERVICE* — Transfer to/from this reticle pod load port is enabled. A transition to OUT OF SERVICE disables the reticle pod load port for transfer use.

9.5.3.4 *TRANSFER READY* — A sub-state of IN SERVICE. The reticle pod load port is available for pod transfer. The transfer can either be manual or automated, and can be a load or an unload. This state contains two sub-states, which are used depending on whether or not a reticle pod is present on the load port (READY TO LOAD and READY TO UNLOAD).

9.5.3.5 *READY TO LOAD* — A sub-state of TRANSFER READY. When transitioning to the TRANSFER READY state, if a reticle pod is not present on the specified reticle pod load port, this is the active sub-state. In this state, the reticle pod load port is available to be loaded with an external reticle pod.

9.5.3.6 *READY TO UNLOAD* — A sub-state of TRANSFER READY. When transitioning to the TRANSFER READY state, if a reticle pod is present on the specified reticle pod load port, this is the active sub-state. In this state, the reticle pod load port is available for unloading of a reticle pod from the reticle pod load port to material handling equipment. When the reticle pod load port is being used by the equipment, the state shall transition to TRANSFER BLOCKED.

9.5.3.7 *TRANSFER BLOCKED* — The reticle pod transfer state is neither READY TO LOAD nor READY TO UNLOAD. Because of reticle pod load port related activity being performed, transfer is not available to/from this reticle pod load port at this time.

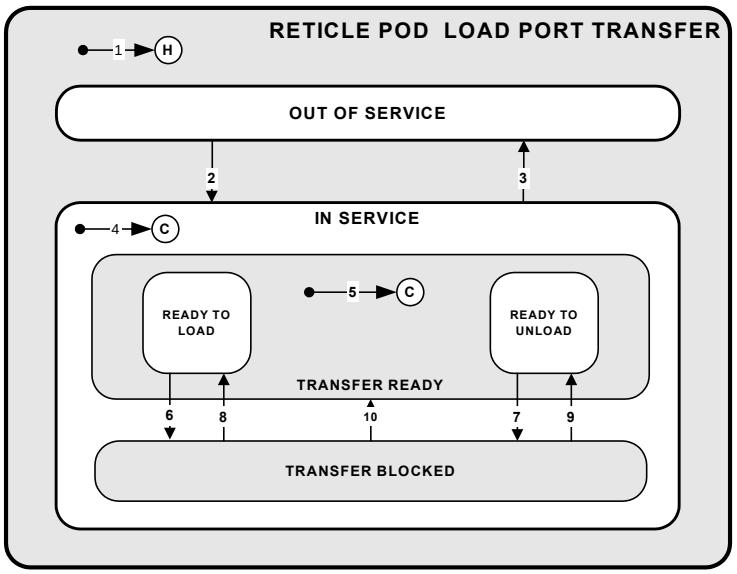


Figure 1
Reticle Pod Load Port Transfer State Model Diagram

9.5.4 *Reticle Pod Load Port Transfer State Transition Table*

Table 5 Reticle Pod Load Port Transfer State Transition Definition

#	*Previous State*	*Trigger*	*New State*	*Actions*	*Comments*
1	(no state)	System reset.	OUT OF SERVICE or IN SERVICE (History)		This transition is based on what the current transfer status was prior to system reset. Data required to be available for this event report: PortID
2	OUT OF SERVICE	The host or an operator has invoked the ChangeServiceStatus service for this load port with a value of IN SERVICE.	IN SERVICE		Reticle Pod Load port is now usable for transfer. Data required to be available for this event report: PortID
3	IN SERVICE	The host or an operator has invoked the ChangeServiceStatus service for this load port with a value of OUT OF SERVICE.	OUT OF SERVICE		Reticle Pod Load port is now rendered unusable for transfer. Attempted usage of the reticle pod load port for reticle pod transfer after the state transition results in an alarm. Data required to be available for this event report: PortID
4	IN SERVICE	*Service:* The host or an operator has invoked the ChangeServiceStatus service for this load port with a value of IN SERVICE. *System Reset:* This transition can be activated by an equipment re-initialization.	TRANSFER READY or TRANSFER BLOCKED		This is the default entry into IN SERVICE. The state is TRANSFER BLOCKED if the reticle pod, or reticle pod load port, is not available for pod transfer. Otherwise, the state is TRANSFER READY. Data required to be available for this event report: PortID
5	TRANSFER READY	*Service:* The host or an operator has invoked the ChangeServiceStatus service for this load port with a value of IN SERVICE. *System Reset:* This transition can be activated by an equipment re-initialization. *Failed Transfer:* If a transfer fails, this transition is activated by transition #10.	READY TO LOAD or READY TO UNLOAD		When entering the TRANSFER READY state, if a reticle pod is present, the sub-state is READY TO UNLOAD, else the sub-state is READY TO LOAD. If the state is READY TO LOAD, data required to be available for this event report: PortID If the state is READY TO UNLOAD, data required to be available for this event report: PortID PodID

#	Previous State	Trigger	New State	Actions	Comments
6	READY TO LOAD	*Manual*: "The equipment recognizes the logical indication of the start of a manual load transfer. This trigger is configurable by the user, examples are included in table 8." *Automated*: The PIO load transfer is beginning and the PIO ready signal is activated.	TRANSFER BLOCKED		Data required to be available for this event report: PortID
7	READY TO UNLOAD	*Manual*: The equipment recognizes a logical indication of the start of an unload transfer. *Automated*: The PIO unload transfer is beginning and the PIO ready signal is activated.	TRANSFER BLOCKED		Data required to be available for this event report: PortID
8	TRANSFER BLOCKED	*Manual:* The reticle pod unload transfer has completed, and the reticle load port is now empty and ready for load transfer. *Automated:* The PIO unload transfer ends with a PIO complete signal.	READY TO LOAD		Data required to be available for this event report: PortID
9	TRANSFER BLOCKED	*Manual: Activities using reticles contained within the pod have completed* the reticle pod is ready to be removed. *Automated:* Handling for reticles destined for the reticle pod has completed, or a CancelPod/ CancelPodAtPort service has been issued, and the reticle pod is ready to be removed. This is indicated by a PIO unload request signal.	READY TO UNLOAD		The reticle pod on the reticle pod load port can now be unloaded from the reticle load port to an external entity. Data required to be available for this event report: PortID PodID
10	TRANSFER BLOCKED	The transfer was unsuccessful, and the reticle pod was not loaded or unloaded.	TRANSFER READY		The sub-state of TRANSFER READY, which is decided by transition #5. Data required to be available for this event report: PortID

10 Reticle Pod Object

10.1 Information about a reticle pod is encapsulated as an object. This allows the host to exchange information with the equipment about one or more specific reticle pods using services defined in SEMI E39, Object Services Standard. A reticle pod has properties (attributes) that are defined in Table 6, Pod Attribute Definition.

10.2 *Object Instantiation*

10.2.1 The reticle pod object is a software representation of the reticle pod in the equipment. Under normal circumstances this object is instantiated by the equipment when the host uses the Bind, ReticleTransferJob, or PodNotification service or when the equipment successfully reads the PodID from the reticle pod. A reticle pod object is instantiated by PodID read only if there are no currently existing objects of the same object type with the PodID just read. A reticle pod object can also be instantiated by either the ProceedWithPod or CancelPod Services on an UNASSOCIATED port. (This implies a failed PodID read event.) The ContentMap attribute will be an empty list (a list of zero) when the instantiation is done by PodID read. The SlotMap attribute should be a list consisting of all slots enumerated as "UNDEFINED" when the reticle pod object is instantiated by PodID read.