SEMI E170-0215
SPECIFICATION FOR PRODUCTION RECIPE CACHE (PRC)

This Standard was technically approved by the Information & Control Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on February 10, 2015. Available at www.semiviews.org and www.semi.org in February 2015; originally published July 2014.

SUPERSEDED

1 Purpose

1.1 The purpose of this Standard is to provide a standardized recipe transfer and storage mechanism from Recipe Server to equipment, which is used for semiconductor manufacturing systems or similar ones, where the management of the master copies of the recipes for mass-production is centralized on the Recipe Server.

1.2 The purpose of this Standard is to provide a standardized mechanism to transfer production recipes from Recipe Server to equipment, and to store the recipes securely in the equipment, so that the recipes in the equipment are exact subset copies of the recipes in the Recipe Server and are protected from non-authorized access, in a factory in which the master copies of production recipes are managed on the Recipe Server.

1.3 The purpose of this Standard is to provide a recipe transfer and storage mechanism which provides the following benefits.

- Protection of the recipes in equipment from non-privileged access by defining a protected recipe space and Recipe User Group management capability
- Authentication of consistency of recipes in equipment with Recipe Server by providing an optional Pre-Execution Check mechanism
- Reduction of operator efforts to download recipes that are used for production from Recipe Server to equipment by providing an automatic transfer mechanism
- Reduction of operator effort required to maintain consistency of recipes in equipment with recipes in Recipe Server by providing an automated consistency maintenance mechanism
- Reduction of communication and computation overhead and cost due to recipe download related transactions by adopting Cache Mode operation

1.4 The purpose of this Standard is to provide a recipe transfer and storage mechanism which is compatible with current practices in the following aspects for easier deployment.

- Recipe identification and uniqueness control
- Recipe linkage
- Variable Parameter management
- Recipe operations except for production execution
- Consistency check of a recipe with the recipe in Recipe Server just before execution (Pre-Execution Check)
- Conventional recipe download modes such as Pre-Download Mode or Full Download Mode

2 Scope

2.1 This Standard covers a definition and usage of Production Execution Mode (PEM) state which represents the status of equipment, whether the equipment is in use for mass production or not.

2.2 This Standard covers a definition and usage of Recipe User Group for recipe security control.

2.3 This Standard covers a definition of new recipe space named Production Recipe Cache (PRC) in equipment, which is dedicated for host controlled process execution in which recipes shall be synchronized with the host and shall be protected from modification by unauthorized operations.

2.4 This Standard covers definitions of mechanisms and operations which manage PRC so that recipes in PRC are protected from non-privileged access and are consistent with the recipes on the Recipe Server in the host.

Documents Document Sharing Hub
Full document details: SEMI E170-0215

NOTICE: SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 *No Message Definition* — This Document does not define any messaging. Messaging may be defined in a separate standard.

3.2 *Master Copies on Recipe Server* — This Standard has a precondition that the master copies of the recipes that are used in PEM On state are stored and managed on Recipe Server. This Standard cannot be applied to the factory in which a Recipe Server does not exist.

3.3 *Linked Recipe Compatibility* — This Standard does not define any requirements for linked recipes but is intended to be compatible with currently implemented recipe linkage schemes by allowing the structure of PRC to be composed as equivalent with the structure of the Conventional Recipe Space.

3.4 *Recipe Identification* — This Standard does not define any recipe identification scheme requirements but is intended to be compatible with the existing recipe identification schemes.

3.5 *Variable Parameters* — This Standard does not define any specific requirements for Variable Parameters but is intended to be compatible with the current implementations.

3.6 *User Authentication and Access Privilege* — This Standard does not define any requirement for user authentication and access privileges but only recommends the use of access control on PRC resources and functions. User authentication and access privilege should be defined separately from this Standard.

4 Referenced Standards and Documents

4.1 *SEMI Standards and Safety Guidelines*

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

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

SEMI E40 — Standard for Processing Management

SEMI E87 — Specification for Carrier Management (CMS)

SEMI E94 — Specification for Control Job Management

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *FOUP* — front opening unified pod

5.1.2 *GEM* — generic equipment model

5.1.3 *MES* — manufacturing execution system

5.1.4 *PJ* — process job

5.1.5 *PRC* — Production Recipe Cache

5.1.6 *PEM* — Production Execution Mode

5.2 *Definitions*

5.2.1 *Cache Full* — the required recipe cannot be written into the PRC because the PRC does not have enough space to write the specified recipe.

5.2.2 *Cache Hit* — the required recipe is found in the PRC.

5.2.3 *Cache Miss* — the required recipe is not found in the PRC.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-0215

5.2.4 *carrier* — a container, such as a FOUP or open cassette, with one or more positions for holding substrates. [SEMI E87]

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. [SEMI E30]

5.2.6 *Conventional Recipe Space* — an existing recipe space on the equipment, which is normally used by every user, including the host, for various purposes such as equipment installation, process development and production operation.

5.2.7 *host* — the factory computer system or an intermediate system that represents the factory and the user to the equipment [SEMI E87], the intelligent system which communicates with the equipment. [SEMI E4, SEMI E5]

5.2.8 *multi-part equipment* — equipment that has a capability to accomplish multiple operations that require recipe execution by multiple users simultaneously.

5.2.9 *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. [SEMI E87]

5.2.10 *operator* — a human who operates the equipment to perform its intended function (e.g., processing). The operator typically interacts with the equipment via the equipment supplied operator console. [SEMI E30]

5.2.11 *privileged user* — a user who is allowed to use system functions that are not available to other users.

5.2.12 *production equipment* — equipment used to produce product, such as semiconductor devices, including substrate sorting, process, and metrology equipment and excluding material handling equipment. [SEMI E87]

5.2.13 *Production Execution Mode (PEM)* — a mode of equipment, in which the equipment is used for production. The state which determines whether the equipment operates in this mode or not shall be set by the host or privileged user. In PEM On state, Online/Remote/Auto may typically be used; however other combinations may also be used for some purposes such as operator assistance, so PEM is an independent mode from Online/Offline, Remote/Local or Auto/Manual.

5.2.14 *Production Execution Mode Recipe (PEM Recipe)* — a recipe which is used in PEM On state. PEM Recipe consists of not only recipes for mass production lots but also recipes for engineering lots, etc., as long as the recipes are used in PEM On state. Any recipe that is prepared to be used in the PEM On state shall be considered as a PEM Recipe.

5.2.15 *Production Recipe Cache (PRC)* — a recipe space that resides in equipment and holds exact subset copies of PEM Recipes on the Recipe Server by using PRC Operation.

5.2.16 *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., carrier, port, etc.). [SEMI E87]

5.2.17 *Recipe Executor* — a component of a module that stores and executes recipes. [SEMI E40]

5.2.18 *Recipe Execution Space* — a recipe space in the equipment which holds the recipe(s) under execution.

5.2.19 *Recipe Header* — a set of descriptors of the recipe which contains information related to the recipe such as linkage information. Recipe Header is optional information defined by user or standardized if needed.

5.2.20 *Recipe Queue Space* — a recipe space in the equipment which holds the recipe(s) already referred and queued for execution.

5.2.21 *Recipe Server* — a computer system in the host, which maintains the master copy of the recipes that are used by the equipment for host-controlled process executions.

5.2.22 *substrate* — material held within a carrier. This can be product, or durables such as reticles. [SEMI E87]

5.2.23 *Unique ID* — an additional ID for a recipe, which provides a unique name to the recipe. In case Recipe ID cannot ensure naming uniqueness of recipes, Unique ID may be used in addition to the Recipe ID.

5.2.24 *user* — a human or humans who represent the factory and enforce the factory operation model. A user is considered to be responsible for many setup and configuration activities that cause the equipment to best conform to factory operations practices. [SEMI E30]

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-0215

5.2.25 *Recipe User Group* — a group which consists of one or more users who have the same privilege to access specified recipes from a security control view point. The recipes may be specified as the ones in the same recipe space. Recipe User Group is not intended to be used for safety control purposes.

6 Conventions

6.1 *Objects*

6.1.1 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 Object Services Standard (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.

6.1.2 By defining these entities as objects that comply with OSS, 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.

6.1.3 *Object Properties*

6.1.3.1 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.

6.1.3.2 Properties shall be accessible to the host via the service GetAttr. Using SEMI E39 Object Services Standard, for example, it is possible to:

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

6.1.4 *Rules for Object Properties*

- Attributes with RO (Read Only) access cannot be changed using SetAttr service as defined in OSS.
- Attributes with RW (Read/Write) 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.

6.1.5 *Object Attribute Table*

6.1.5.1 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 (Y), conditional (C), or optional (O) 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*

6.2 *State Model Methodology*

6.2.1 A state model consists of four elements: a State Model Diagram, a State Model Definition Table, a State Definition and a State Transition Table.

6.2.2 *State Model Diagram* — 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]

[1] Elsevier Science, P. O. Box 945, New York, NY 10159-0945; http://www.journals.elsevier.com/science-of-computer-programming/

SUPERSEDED
Documents
Document Sharing Hub
Full document details: SEMI E170-0215

6.2.3 *State Model Definition Table* — The State Model Definition Table used in this Standard has the following format. This table defines states and possible transition(s) from each state side by side. Each state has one or more transitions. In case the transition comes from outside this table, the State column for the transition may be blank (see #1). Also, in case the transition comes from unspecified multiple states with the same condition, the state definition column may say '#Any state,' and there may not be an explicit transition number.

6.2.3.1 *Definition of State* — Columns under the State column define States with No. (Number), Name and Abstract of Definition. 'No.' corresponds to the state number in the associated state diagram. 'Name' defines a name of each state. 'Abstract of Definition' provides an abstract of the State Definition in the State Definition Table.

6.2.3.2 *Definition of Transition* — Columns under the Transition column define Transitions with No. (Number), Abstract of Trigger, Abstract of Action, New State, and Comments. 'No.' corresponds to transition number in the associated state diagram. 'Abstract of Trigger' and 'Abstract of Action' provide abstracts of the Trigger and Action in the State Transition Table accordingly. 'New State' defines a state number to move after the transition is completed by pointing one of the states defined in the state definition in left side of the table. The Comment column may be used to put comments to each transition or their From state.

Table 2 State Model Definition Table

State			*Transition*				
No.	*Name*	*Abstract of Definition*	*No.*	*Abstract of Trigger*	*Abstract of Action*	*New State*	*Comments*
-#1	-#1	-#1	T00			S00	
Snn#2	# Any state#2	-#2	-			S03	
S00			T01			S01	
S04#3			T02			S00	
S01			T03			S02	
			T04			S03	
S02			T05#4			S00	
S10#5			-	-	-	-	
S11			T10			S12	
S12			T11			S11	
S03			T06			S00	

#1 In case the transition comes from outside of this table, the state definition column may be blank.

#2 In case the transition comes from unspecified multiple states with the same condition, the state definition column may say '#Any state'.

#3 A state which has substates.

#4 One transition path has multiple cases (trigger and action pairs).

#5 A state separated by a dotted line is a parallel state of the state above.

6.2.4 *State Definition Table* — State definition tables are provided in conjunction with the state diagrams to explicitly describe the definition of each state. A state definition table contains columns for State Number, Mnemonic, State Definition, and Comments.

Table 3 State Definition Table

Num	*Mnemonic*	*State Definition*	*Comments*

6.2.5 *State Transition Table* — State transition tables are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. A state transition table contains columns for Transition Number, Previous

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-0215

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 4 State Transition Table

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

6.2.6 *State Model Requirements*

6.2.6.1 *Requirement* — The state models included in this Standard are a requirement for compliance. Equipment must maintain state models for each of the required state models as defined in this Standard. Equipment shall maintain individual and unique state models for each logical entity instantiated or physical entity in the equipment that has state models associated with it.

6.2.6.2 *Representation as the Host View* — A state model represents the host's view of the equipment, and does not necessarily describe the internal equipment operation. All 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 state models in this Standard for transition to another state. In this case, the equipment makes the required transition without any additional actions in this situation.

6.2.6.3 *Additional Substates* — Some equipment may need to include additional substates other than those in this Standard. Additional substates may be added, but shall not change the defined state transitions in this Standard. All expected transitions between states in this Standard shall occur.

6.2.6.4 *Uniqueness of Event Identifier* — The event identifier reported during a particular state transition change for each of these state models shall be shared for all associated state models but unique for each transition. For example, if the equipment has two load ports and the load port state model defines 10 transitions, there must be exactly 10 event identifiers for each load port transfer state model but not 10 for each physical load port. The information identifying the physical entity or logical entity undergoing the transition will be contained within the associated event report.

6.2.6.5 *Events* — All state transitions in this Standard, unless otherwise specified, shall correspond to collection events. More explicitly, there must be a unique collection event for each state transition.

6.2.6.6 *Events for Multiple AND Substates* — In case a state model is defined with multiple AND substates, the equipment may report all state entry events with only one collection event.

6.2.6.7 *Events for Conditional Path* — In case conditional paths are defined in the state model, it is not necessary to report any state transition(s) until a terminal state is reached at which time each transition used to reach that state is reported.

6.3 *Object*

6.3.1 *Recognition of Object* — From the host point of view, an object is instantiated if the host is able to query the equipment about that object, its current state, and other attributes. Once instantiated, the object is considered destroyed (no longer instantiated) if the response to such queries is 'unknown object'.

6.3.2 *Object Identifier (ObjID)* — The purpose of an Object Identifier is to allow references to an object within the system. The object identifier is assigned when an object is instantiated and should be unchanged or persistent until the end of the object lifecycle. The Object Identifier shall be unique at the equipment during lifecycle of the object.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-0215

6.4 *Services*

6.4.1 *Definition of Service* — 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.

6.4.1.1 *Notification Message Service* — 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.

6.4.1.2 *Request Message Service* — 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).

6.4.2 *Service Message Description* — A service message description table defines the parameters used in a service, as shown in the following table:

Table 5 Service Message Description Table

Service Name	*Type*	*Description*

#1 Type can be either 'N' = Notification or 'R' = Request & Response.

6.4.3 *Service Message Parameter Definition* — A service parameter dictionary table defines the description, range, and type for parameters used by services, as shown in the following table:

Table 6 Service Message Parameter Definition Table

Parameter Name	*Form*	*Description*

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

6.4.4 *Service Message Definition* — A service message description table defines the parameters used in a service message, and also, describes each message and its cause/effect to the equipment, as shown in the following table:

Table 7 Service Message Definition Table

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

6.4.4.1 *Definition of Req/Ind and Rsp/Conf Columns* — 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'.

6.4.4.2 *Definition of Codes for Req/Ind and Rsp/Conf Columns* — 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):

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-0215

Table 8 Codes For Req/Ind and Rsp/Conf Columns

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).

6.5 *Variable Data Definitions*

6.5.1 This defines variable data requirements. Values of these variables are available to the host via collection event reports and host status queries.

6.5.2 *Event Report Requirement* — The identifier of that object and all of the attributes of that object shall be available for inclusion in event reports associated with that object.

6.5.3 *Object Attribute Variable in Non-extinction Event* — The object attribute variables in event reports linked to non-extinction event(s) shall contain the values of the attributes after the transition. This requirement allows the receiver of the report to know the current condition of the object.

6.5.4 *Object Attribute Variable in Extinction Event* — The object attribute variables in event reports linked to extinction event(s) shall contain the values of the attributes before the transition unless it is specifically stated that the destruction transition modifies the attribute value. This requirement allows the receiver of the report to know the final condition of the object at the time it was deleted.

6.5.5 Subscripted variables are used either as items within a list or to differentiate data representing different entities. Subscripted variables are always valid.

6.5.6 *Table Format* — The following table defines variable data that shall be provided by the production equipment.

Table 9 Variable Data Definitions

Variable Name	*Description*	*Type*	*Access*	*Comment*

7 PRC Conceptual Descriptions

7.1 *Conceptual Descriptions* — This chapter defines and describes a concept of PRC. This chapter does not contain any requirements.

7.2 *Equipment Centric vs. Recipe Server Centric* — There may be several management policies of production recipes from equipment centric to Recipe Server centric. This Standard is designed to be applicable to the methods described in the following Figure.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-0215

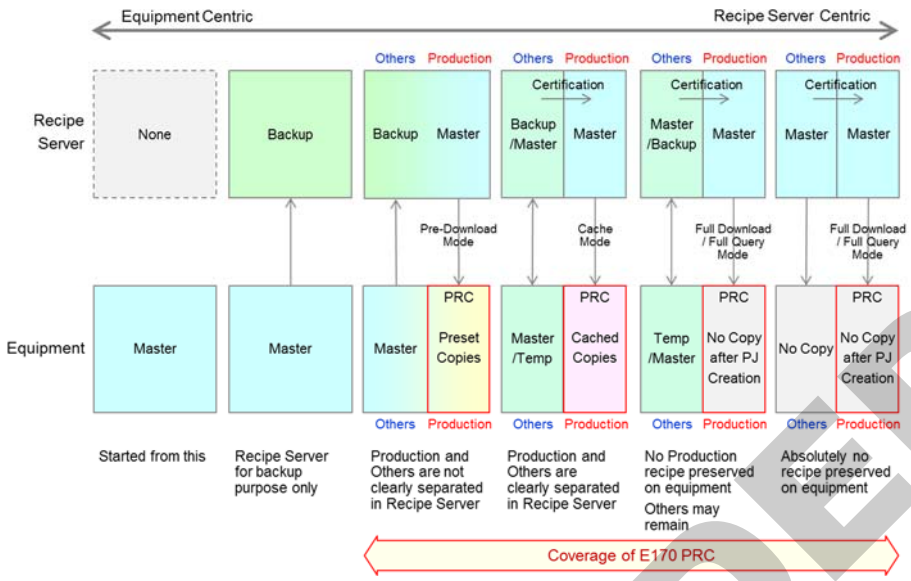

Figure 1
Typical Recipe Server-Equipment Models and Coverage of PRC

7.3 *Intended Application of This Standard* — The intended application of this standard is to construct a Recipe Server centric recipe management system which provides 'What you have done on the Recipe Server is what you will have on the equipment' operability and 'Secure and safe Production Execution path isolated from operators'. The key features are:

- Clear separation between 'Production' and 'Preparation' activities.
- 'Centralized Management' of the master copy of 'Production Recipes' on Recipe Server.
- Only privileged user can handle Production Recipes (e.g. Certification).
- Leave 'Conventional Recipe Space' flexible as it is, for installation, maintenance, recipe development, etc.
- Introduction of 'Production Recipe Cache' as secure and safe 'Production Execution Path' for Production Recipes.
- Introduction of 'Production Execution Mode' flag which selects 'Production Recipe Cache' instead of 'Conventional Recipe Space' for secure production execution, when the flag shows the equipment is in 'Production Execution Mode'.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-0215

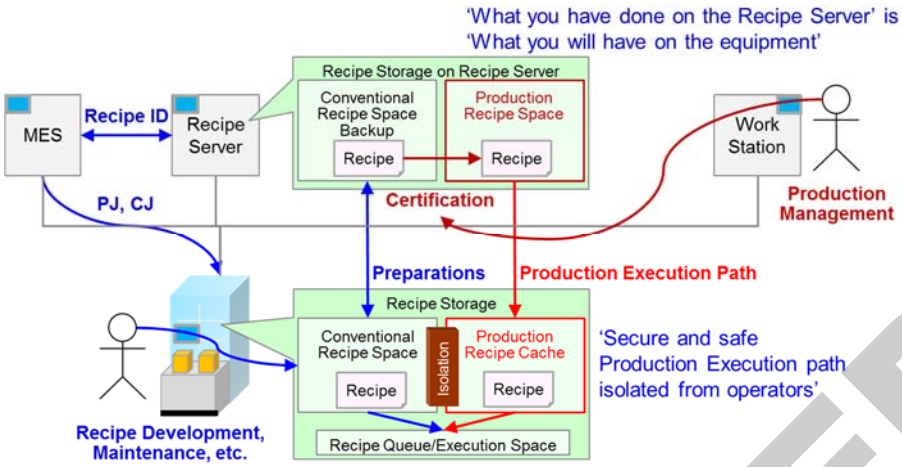

Figure 2
Example of Intended Application of PRC

7.4 *Direct Service from Recipe Server in the Host* — Introduction of Query Recipe from equipment may ease the use of direct service from Recipe Server in the host system instead of the use of service through MES.

7.5 *Reduction of Computing and Communication Loads for Recipe Download and Check* — Introduction of protected recipe space 'PRC' and Cache Mode operation reduces computing and communication loads for recipe download and avoids pre-execution check overhead. Recently used recipes are safely preserved in PRC and reused for consequent PJ creations with the same recipe.

7.6 *Deployment Considerations* — Introduction of Enable/Disable capability by the host makes deployment easier. The host can enable PRC when the host is ready to use PRC.

7.7 *Cascading Considerations* — Introduction of Cache Mode operation or Pre-Download Mode operation gives more freedom to recipe service timings for better cascading of process execution while reducing recipe related transactions.

7.8 *Operational Compatibility* — To be upward compatible with conventional operations, this Standard covers the following PRC Operation Modes. Pre-execution check is not logically required for PRC which is properly implemented and operated, however, may be required to help deployment path from conventional Pre-execution check based system or for double check purpose in some situations.

- PRC Operation Modes
  - Full Download Mode: The host downloads required recipes every time just before demanding a PJ creation.
  - Full Query Mode: Equipment queries required recipes every time upon PJ creation.
  - Pre-Download Mode: The host downloads recipes prior to PJ creation demands for some duration.
  - Cache Mode: Equipment holds recently used recipes in the PRC and queries required recipes in case they are not in the PRC.
- Pre-Execution Check Operation
  - Pre-Exe Check: Equipment uploads check information of the required recipe to the host for consistency check upon PJ creation, and the host responds with affirmative message or download of the correct recipe.

7.9 *Linked Recipe Compatibility* — To be compatible with current implementations, PRC introduces a concept which makes the recipe linkage scheme of PRC the same with Conventional Recipe Space.

7.9.1 *Management of Host-Managed Sub-Recipes* — Sub-recipes, which are under management of the host, may be managed similarly in PRC. Those recipes should reside in the PRC, and should be linked together in the same way as the recipes in the Conventional Recipe Space. This Standard does not define the linkage specification.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-0215

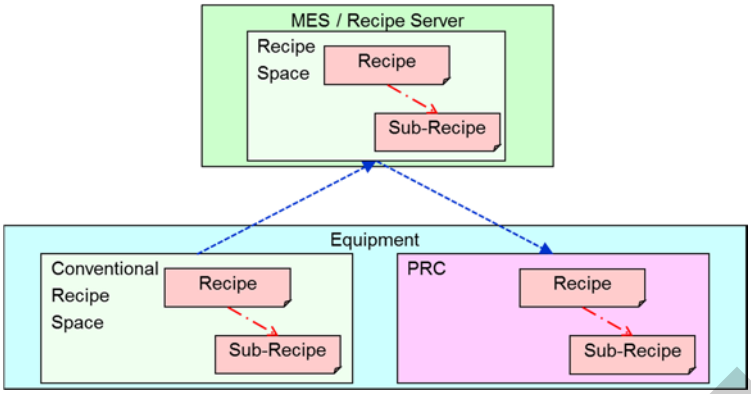

Figure 3
Linkage to Host-Managed Sub-Recipes

7.9.2 *Management of Locally Managed Sub-Recipes* — Sub-recipes which are not under management of the host but are linked under the recipes managed by host are out of the scope of PRC. Those recipes may reside somewhere in the equipment (may reside in a part of Conventional Recipe Space), and may be linked from the recipes in PRC in the same way as the link from the recipes in the Conventional Recipe Space. This Standard does not define the way of linkage.

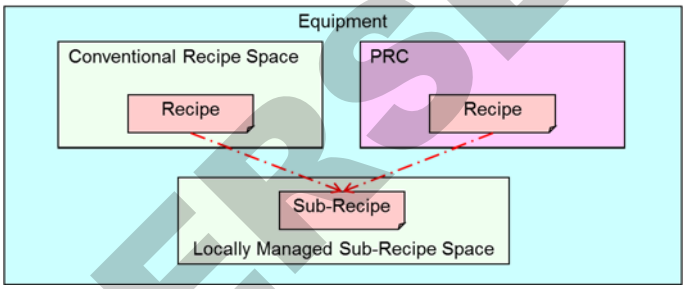

Figure 4
Linkage to Locally Managed Sub-Recipes

7.10 *PRC Space Structure Concept* — The PRC may be used in the following ways to maintain compatibility with existing implementations of recipe space.

7.10.1 *Subspace Structure in PRC* — PRC may be structurally divided into multiple tiers of multiple subspaces. Alignment of subspace structure with the one in the Conventional Recipe Space and/or the recipe space in Recipe Server may be useful for following purposes.

7.10.1.1 *Linked Recipe Aspect* — This subspace concept may be used to deal with Linked Recipe.

7.10.1.2 *Recipe Security Aspect* — This subspace concept may also be used for security purposes.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-0215

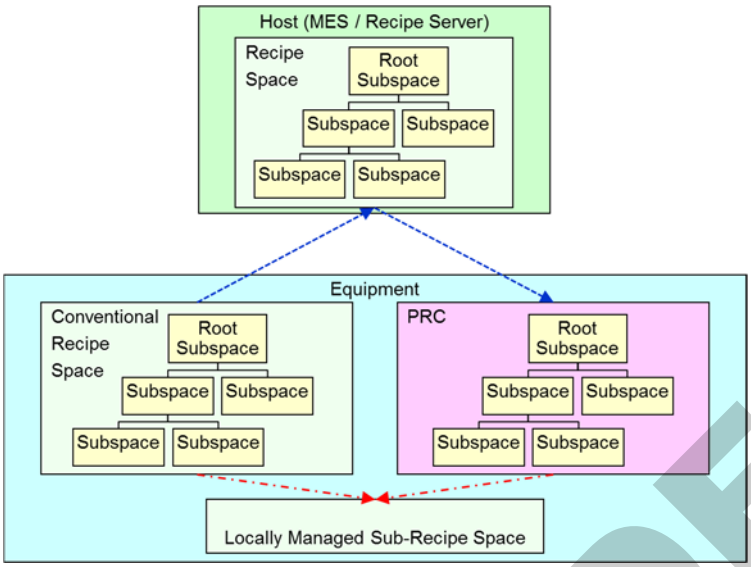

Figure 5
PRC Space Structure Concept

7.10.2 *How to Specify Subspace* — To be compatible with the current implementations, the same subspace specification (path specification) mechanism, which is used in the Conventional Recipe Space, may be used.

7.10.3 *Subspace Structure Relationship between Conventional Recipe Space and PRC* — The subspace structure of PRC may be aligned with the subspace structure of Conventional Recipe Space in the equipment for easier correlation.

7.10.4 *Subspace Structure Relationship between PRC and Recipe Server* — The subspace structure in the recipe serving function in the host may be aligned with the subspace structure of PRC in equipment for easier correlation.

8 PRC Space Requirements

8.1 *PRC Space Implementation Requirement* — Equipment compliant with this Standard shall have a recipe space named PRC in addition to the Conventional Recipe Space.

8.1.1 *PRC as Logical Space* — The space requirement of PRC is to prepare a space which can logically separate PEM Recipes from others. This Standard does not define an implementation method. For example, equipment may adopt a method which stores all recipes in Conventional Recipe Space as they are, mark PEM Recipes, and deal with the marked recipes as PEM Recipes in PRC. In other words, equipment may adopt a method which stores all recipes in the same location, but enables a flag to mark PEM Recipes. The PEM Recipes are then considered to be in PRC.

8.2 *Usage of PRC and Conventional Recipe Space* — Usage of PRC and Conventional Recipe Space are shown in the following table in relation with PRCOBJ State and PEMJ State.

Table 10 Usage of PRC and Conventional Recipe Space

States		*Recipe Spaces*		*Comment*
PRCOBJ State	*PEMJ State*	*Conventional Recipe Space*	*PRC*	
PRC Disabled	Not applicable	Holds all recipes the equipment has Used for all purposes	Not used	This is the state which is compatible with the conventional operation

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-0215

States		*Recipe Spaces*		*Comment*
PRCOBJ State	*PEMJ State*	*Conventional Recipe Space*	*PRC*	
PRC Enabled	PEM Off	Holds recipes other than PEM Recipes that are managed on Recipe Server Used for process execution	Holds copies of PEM Recipes that are managed on Recipe Server Not used for process execution Used for preparations of PRC before PEM On such as Cache Clear operation or Download operation	This is the state for non-mass production operations such as installation, maintenance and process development
	PEM On	Holds recipes other than PEM Recipes that are managed on Recipe Server Used for process execution of recipes that are not managed on Recipe Server	Holds copies of PEM Recipes that are managed on Recipe Server Used for process execution of recipes that are managed on Recipe Server	This is the state for mass production

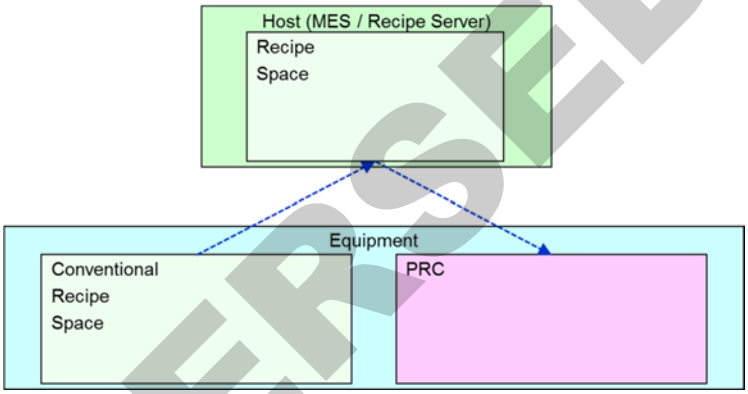

Figure 6
Concept of PRC

8.3 *Multi-Part Equipment Compatibility (Optional)* — In case the equipment has a capability to divide itself into multiple parts that can operate in PEM On state or in PEM Off state independently (as a consequence, each part has its own Recipe Execution Space, and also may have its own Recipe Queue Space), the PEMJ state of the each part shall separately be managed and the use of PRC shall separately be controlled as shown in the following figure.

8.3.1 *Multi-Part Equipment with One PRC* — In case security control is not required between the recipes for multiple parts of the equipment, one PRC may be shared across all parts of the equipment.

8.3.2 *Multi-Part Equipment with Dedicated PRCs* — In case security control is required between the recipes for the multiple parts of the equipment, each part of the equipment shall have a dedicated PRC.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-0215

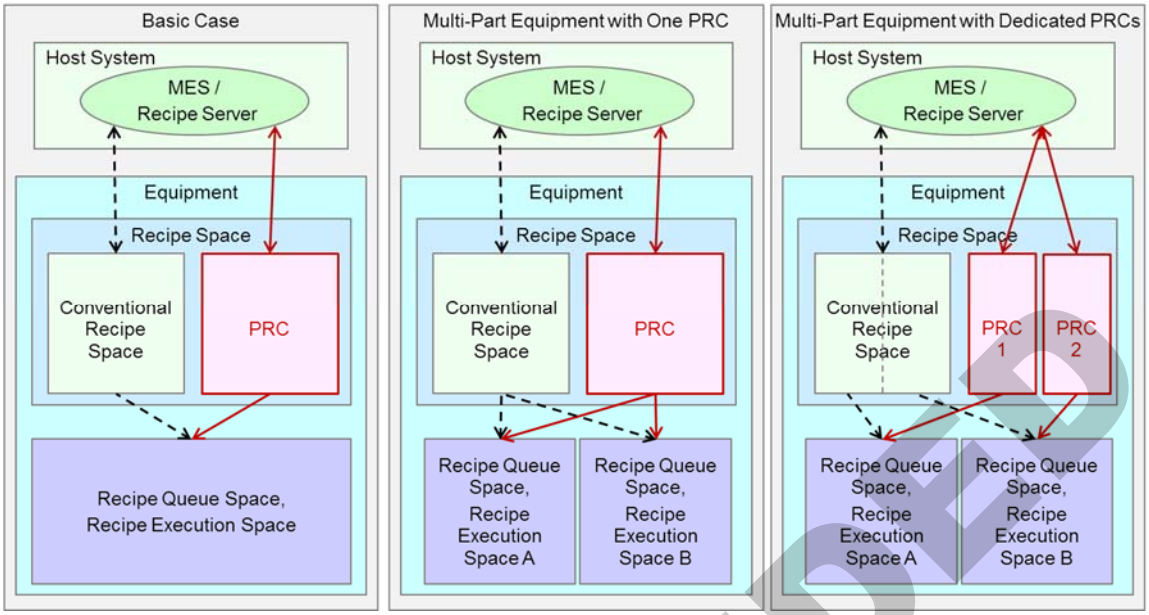

Figure 7
PRC Implementation for Multi-Part Equipment

9 PRC Operational Requirements

9.1 *Operational Preconditions* — This section describes preconditions for operations of the PRC.

9.1.1 *Single Point of Control (SPOC)* — The host side (e.g., something like an Equipment Interface) should maintain which node in the host takes care of which functionality of PRC Operation.

9.1.2 *Single Point of Recipe Management (SPORM)* — The master copy of PEM Recipes shall be managed by the host (normally by Recipe Server) and all PEM Recipe related interactions defined in this Standard shall be done between the host (including Recipe Server) and the equipment. Backup copies may be somewhere in the host, and temporary copies to support seamless execution are normally in the equipment.

9.1.3 *PRC as Temporary Recipe Space* — PRC shall be considered as a temporary recipe space which may be cleared whenever the host or privileged user decides for consistency or security reasons.

9.1.4 *Copy Out of the Recipes Already Referred (Recipe Queue Space and Recipe Execution Space)* — Recipes already used to create PJs shall be copied from the PRC to other protected recipe spaces such as Recipe Queue Space or Recipe Execution Space which hold the recipes while the PJs are queued or executed accordingly, so that recipes in PRC can be overwritten or deleted at any time. Recipes in the protected recipe space (such as Recipe Queue Space or Recipe Execution Space) shall not be able to be read or written by non-host-controlled or non-privileged-user-controlled actions.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-0215