SEMI E170-1217
SPECIFICATION FOR SECURED FOUNDATION OF RECIPE MANAGEMENT SYSTEM (SFORMS)

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 August 18, 2017. Available at www.semiviews.org and www.semi.org in December 2017; originally published July 2014; previously published May 2017.

1 Purpose

1.1 The purpose of this Standard is to provide a standardized 'secure, safe and efficient recipe operation mechanism between Recipe Management System (RMS) and equipment' which is used as a foundation of RMS for semiconductor manufacturing systems or similar ones.

1.2 The purpose of this Standard is to provide a standardized 'secure, safe and efficient recipe operation mechanism between RMS and equipment' which especially supports RMS with the following concepts:

- Management of recipes is centralized to RMS (Single Point Of Recipe Management [SPORM] concept)
- Storage of the master copies of the recipes are centralized to Recipe Server in RMS (Recipe Server Centric concept)

1.3 The purpose of this Standard is to support RMS in the following aspects:

- Extension of recipe identification capability
- Protection of the recipes in equipment from non-privileged access by defining protected recipe spaces per purposes
- Reduction of operator efforts required to maintain consistency of recipes in equipment with recipes in Recipe Server by providing an automated consistency maintenance mechanism based on cache
- 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 based on cache
- Reduction of communication and computation overhead and cost due to recipe download related transactions by adopting cache operation
- Addition of remote access capability to access recipes on the host RMS by using recipe applications (such as recipe editor) on equipment or other workstation

1.4 The purpose of this Standard is to provide a recipe operation mechanism which supports similar with or a superset of current practices in the following aspects for easier deployment.

- Recipe identification and uniqueness control
- Recipe linkage
- Variable Parameter management
- 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 secure, safe and efficient recipe operation mechanism intended to be used as a foundation of RMS.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

2.2 This Standard covers a definition and usage of extended recipe identification scheme which identifies recipes from multiple aspects including uniqueness and security.

2.3 This Standard covers a definition and usage of secured recipe space management mechanism which manages multiple recipe spaces in order to provide logically separated recipe spaces for different security level of recipe operations.

2.4 This Standard covers a definition and usage of an equipment status flag which shows whether the equipment is in use for mass production or not.

2.5 This Standard covers a definition and usage of recipe feeding mechanism from the host to equipment which enables secure, safe and efficient recipe operation for production execution by the host.

2.5.1 This Standard covers a definition of new recipe space named Production Recipe Cache (PRC) as one of secured recipe space 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.

2.5.2 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.

2.6 This Standard covers a definition and usage of remote access operation mechanism which enables remote access of the recipes on Recipe Server in RMS by using recipe applications on equipment or other workstation.

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 for host triggered PJ creation 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 introduces extended recipe identification scheme for new requirements but is intended to be compatible with the existing recipe identification schemes for existing functionalities.

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 — Specification for the 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.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

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 *PEM* — Production Execution Mode

5.1.5 *PJ* — process job

5.1.6 *PRC* — Production Recipe Cache

5.1.7 *RAC* — Remote Access Cache

5.1.8 *RMS* — Recipe Management System

5.2 *Definitions*

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

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

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

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 *extended recipe identifier (Recipe XID)* — an identifier of a recipe which is extended with Version ID, Security ID, Type ID, and Equipment ID in addition to conventional Recipe ID.

5.2.8 *formalized recipe* — a recipe which is recognized by and under control of the RMS in the host.

5.2.9 *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]

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

5.2.11 *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]

5.2.12 *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]

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

5.2.14 *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]

5.2.15 *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.

5.2.16 *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

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

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.

5.2.17 *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.

5.2.18 *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]

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

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

5.2.21 *Recipe Management System (RMS)* — an information system that manages recipes. Especially, the factory information system that manages recipes registered to the factory and is a part of the host. Typically, RMS has a Recipe Server which holds the master copy of the recipes.

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

5.2.23 *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.

5.2.24 *Security Class* — a category of recipe which has the same security requirements.

5.2.25 *Security ID* — an identifier of a recipe which identifies the Security Class of the recipe.

5.2.26 *Secured Recipe Space (SRS)* — a recipe space which logically discriminates recipes per Security Class identified by Security ID of each recipe.

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

5.2.28 *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]

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

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.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

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]

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.

[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 E170-1217

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 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*

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

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.

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:

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

Table 5 Service Message Description Table

Service Name	*Triggered by*	*Type*#1	*Req*#2	*Description*

#1 The 'Type' column is used to indicate whether the service consists of a request/response message pair, 'R', or a single notification message, 'N'.

#2 The 'Req' column is used to indicate whether the service is required, 'R', or optional, 'O'.

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):

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.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

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 Conceptual Descriptions

7.1 *Position of Conceptual Descriptions* — This section defines and describes a concept of this Standard. This section does not contain any requirements.

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

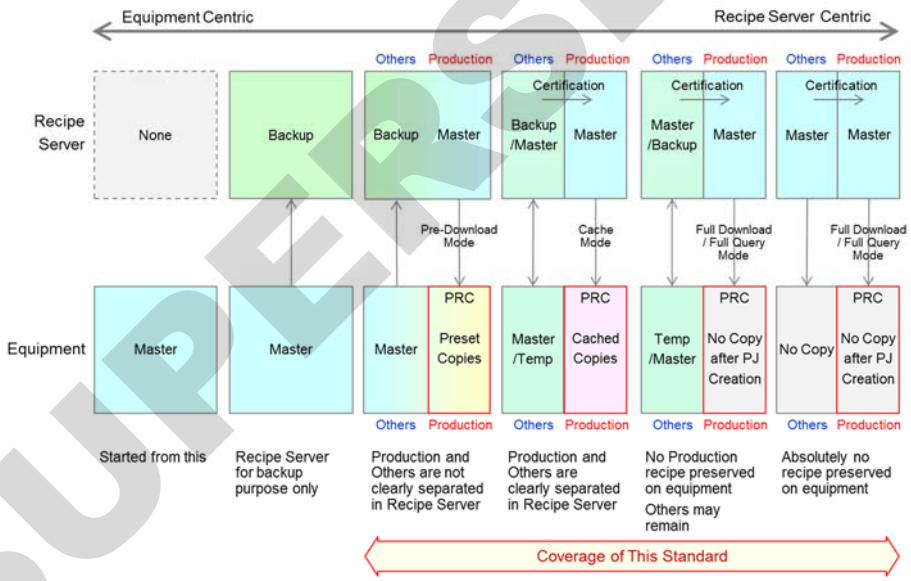

Figure 1
Typical Recipe Server-Equipment Models and Coverage of This Standard

7.3 *Introduction of SPORM Concept* — In order to manage numbers of recipes in secure, safe, and efficient way, this Standard introduces SPORM concept. In SPORM concept, recipes are managed in the following manner.

- A recipe is formalized when it is registered to RMS (which is normally in the host).
  - ➢ Basically, recipes which are not formalized are not used for production related operations.
- The master copy of formalized recipes are stored in a Recipe Server (normally, under RMS in the host).

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

- ➢ Temporary copies of formalized recipes which are made to support quick and efficient operations (such as recipe editing or recipe execution) may be in equipment (normally in cache, queue, or execution spaces).
- All recipe management operations are done under recognition and approval of RMS (normally in the host).
- Backup scheme which prepares for system failure of the host is not in scope of SPORM.

7.4 *Intended Application of This Standard* — The intended application of this Standard is to construct a SPORM based 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, safe, and efficient recipe operation mechanism. The key features are:

- Clear separation between 'Production' and 'Preparation' activities.
- 'Centralized Management' of the master copy of 'Formalized Recipes' on Recipe Server.
- Only privileged user can handle Production Recipes (e.g. Certification).
- Introduction of RecipeXID for more secure identification.
- Introduction of 'Secured Recipe Space' to provide dedicated recipe spaces per purposes.
  - ➢ 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'.
- Introduction of 'Remote Access Cache' in order to access recipes on Recipe Server by using equipment applications.

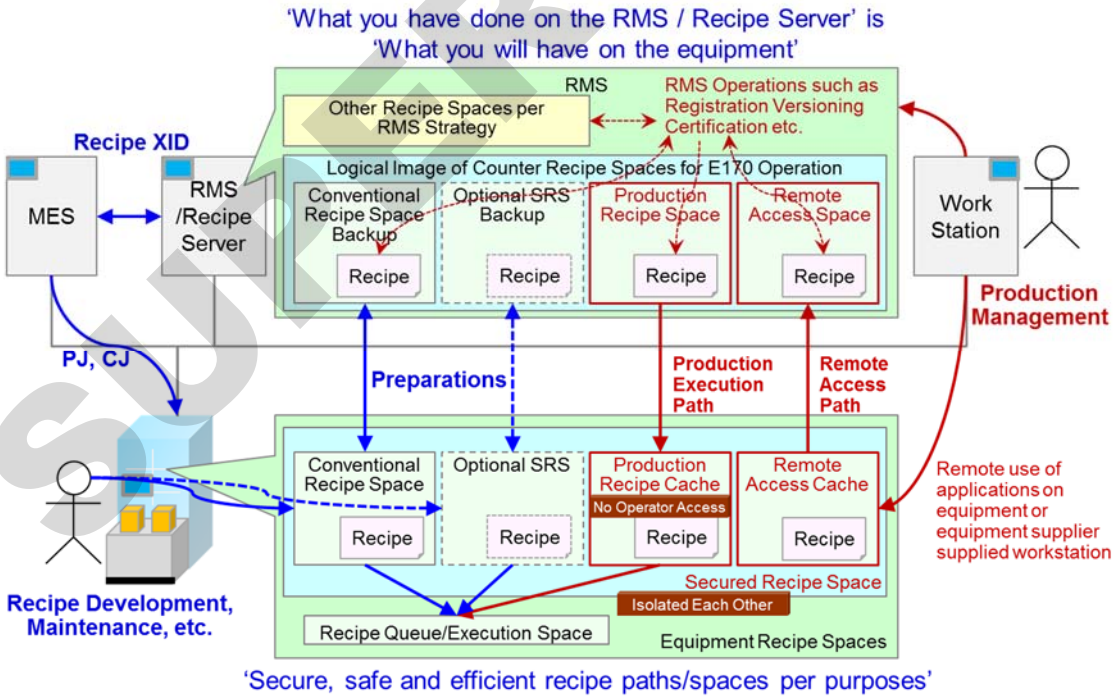

Figure 2
Example of Intended Application of This Standard

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

7.5 *Introduction of RecipeXID* — In order to identify recipes from multiple aspects, this Standard introduces Extended Recipe ID called RecipeXID. RecipeXID consists of the following identifiers.

- Recipe ID: Main recipe identifier (PPID).
- Version ID: An optional recipe identifier which helps unique identification in case Recipe ID is not always unique.
  - Only the host assigns the value
  - Zero length is assigned when the recipe is locally edited on equipment
- Security ID: An additional recipe identifier which indicates the Security Class of the recipe. Security Class is used to identify:
  - In which logical recipe space the recipe shall be stored
  - By which operator the recipe may be used
  - For which operation the recipe may be used
- Type ID: An optional recipe identifier which indicates type of the recipe.
- Equipment ID: An optional recipe identifier that indicates equipment which the recipe is tuned for.
  - May define 'Ideal' equipment for reference recipe

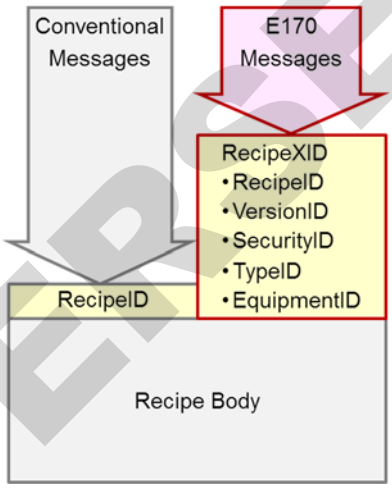

Figure 3
Introduction of Recipe XID

7.6 *Introduction of Secured Recipe Space* — In order to deal recipes with different security requirements, this Standard introduces Secured Recipe Space which logically discriminates recipes per their Security Class specified by SecurityID.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

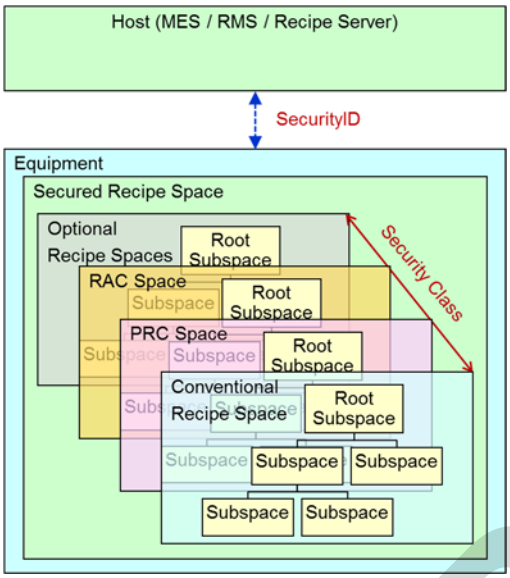

Figure 4
Introduction of Secured Recipe Space

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

7.6.1.1 *Management of Host-Managed Sub-Recipes* — Sub-recipes, which are under management of the host, may be managed similarly in SRS. Those recipes should reside in the SRS, 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.

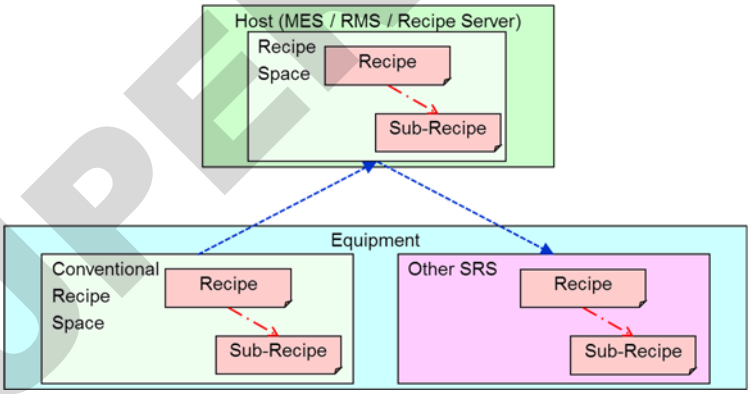

Figure 5
Linkage to Host-Managed Sub-Recipes

7.6.1.2 *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 this Standard. 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 other SRS in the same way as the link from the recipes in the Conventional Recipe Space. This Standard does not define the way of linkage.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

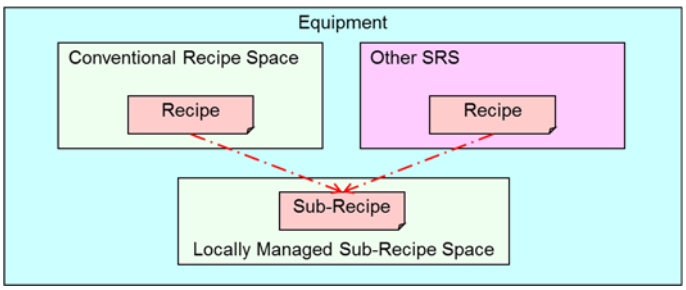

Figure 6
Linkage to Locally Managed Sub-Recipes

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

7.6.2.1 *Subspace Structure in SRS* — SRS 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.

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

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

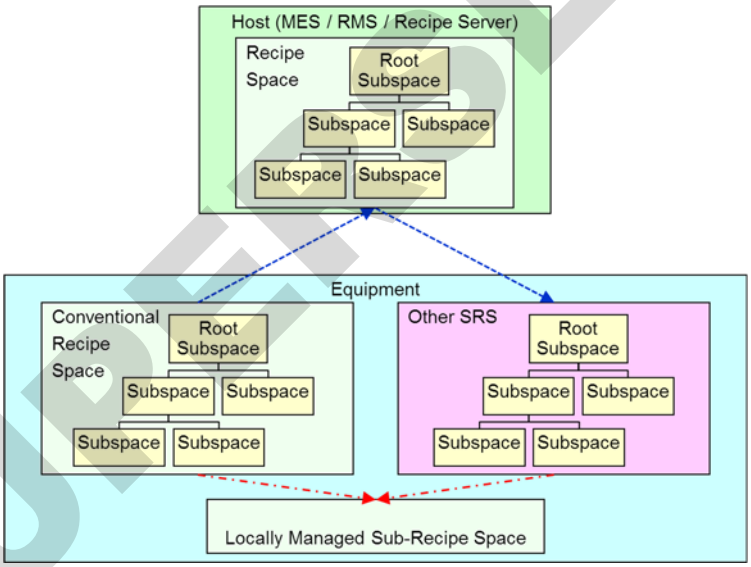

Figure 7
SRS Space Structure Concept

7.6.2.2 *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.

7.6.2.3 *Subspace Structure Relationship Between Conventional Recipe Space and Other SRS* — The subspace structure of other SRS may be aligned with the subspace structure of Conventional Recipe Space in the equipment for easier correlation.

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

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

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

7.7 *Introduction of Production Recipe Cache Operation* — For efficient production recipe transaction, this Standard introduces PRC Operations.

7.7.1 *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.

7.7.2 *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.

7.7.3 *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.

7.7.4 *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.

7.7.5 *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.

7.8 *Introduction of Remote Access Cache Operation* — For SPORM based factory RMS, flexible use of equipment supplier supplied recipe applications (such as recipe editor) on equipment or other work stations to manage the recipes on Recipe Server is important. For this purpose, this Standard provides functionalities to support Remote Access.

7.8.1 *Purpose of Remote Access Operations* — The purpose of Remote Access operation is to perform recipe operations (such as edit, compare etc.) directly to the recipes on the host (RMS) by using equipment applications on equipment or a workstation. Remote Access operation is independent from PRC operation (RAC Space and PRC Space are also independent in equipment), and how to reflect the result of Remote Access operation to Production Recipes which go through PRC operation depends on the business rule in RMS.

7.8.2 *Relationship Between RAC and PRC Operations* — Remote Access operation is independent from PRC operation (RAC Space and PRC Space are also independent in equipment), and how to reflect the result of Remote Access operation to Production Recipes which go through PRC operation depends on the business rule in RMS.

7.8.3 *Preconditions for Standardization* — In order to simplify Remote Access operation for standardization, following pre- and post-processes which use the host system (e.g., RMS operation) are considered as preconditions.

- Pre-process: Prior to a Remote Access operation, the following preparations are expected to respond to the Remote Access operation request from equipment (or workstation).
  - Assign SecurityID which will be used by the equipment to request Remote Access operation
  - Prepare directory of required recipes for the planned work

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

- Post-process: After completion of the Remote Access operation, the following post processing are expected.
  - Acceptance process of the newly registered recipes
  - Utilization process of the results from other recipe work (such as comparison result)

7.8.4 *Intended Remote Access Operations* — Intended operations are as below.

- An operator requests Remote Access session at equipment (or workstation).
- The equipment requests OperatorID and OperatorPassword, that are required for operator authentication by the host, to the operator, assigns SecurityID, and then asks the host for establishment of a RAC session.
- The host authenticates the operator and responds to the equipment.
- Normally, the operator requests directory of recipes to the host, and the host responds with an appropriate RecipeXIDList considering the SecurityID and EquipmentID provided from the equipment.
- The operator may query required recipes to the host.
- The operator uses those recipes for required works such as edit, compare, etc.
- The operator may post the result of the work to the host. The work may result with the information other than recipe; however, the information should be dealt as quasi-recipe from the identification aspect in order to post it to the host through recipe transaction mechanism. TypeID may be used to identify a type of the result.
- The operator closes the session upon completion of the work. The equipment reports to the host and clear recipes and other working results in the RAC.

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

8 Extended Recipe Identification Requirements

8.1 *RecipeXID* — In order to identify recipes from multiple aspects, this Standard introduces Extended Recipe ID called RecipeXID. RecipeXID consists of the following identifiers.

8.1.1 *Recipe ID* — The RecipeID is the main identifier of recipe.

8.1.2 *Version ID* — The VersionID is an optional recipe identifier that can be combined with the RecipeID to uniquely identify the recipe. If the host uses this field, it has the advantage of being factory-set so that uniqueness of the RecipeID and VersionID combination can be assured by the host. The VersionID is a field that can be set only with the cooperation of the host. The VersionID value is intended to be assigned by the factory. In a typical case, the recipe is created at the equipment with the VersionID set to a zero-length string. Upon upload, the recipe might be checked-in to a recipe management system. A part of this check-in could be the modification of the recipe header with a factory generated VersionID.

8.1.3 *Security ID* — The SecurityID is an additional recipe identifier that is used to indicate the Security Class which the recipe belongs to. The SecurityID is not used for unique identification of recipes. The SecurityID is intended to be used for the following purposes.

- In which logical recipe space the recipe shall be stored.
- By which operator the recipe may be used.
- For which operation the recipe may be used.

8.1.4 *Type ID* — The TypeID is an optional recipe identifier which indicates the type of the recipe (such as Main recipe, process chamber recipe, pump recipe, etc.).

8.1.5 *Equipment ID* — The EquipmentID is an additional recipe identifier that indicates equipment which the recipe is tuned for.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

Table 10 RecipeXID Definition

ID Name	*Mnemonic*	*Form*	*Definition*	*Note*
Recipe ID	RecipeID	RecID which is defined in SEMI E40	Recipe ID	RecID may not always be a unique identifier
Version ID	VersionID	Maximum 256 characters	A string that identifies this particular adaptation of a base recipe that shares the same RecipeID. It is intended to be populated based upon factory business rules. Upon creation or modification of a recipe, the VersionID shall be set to a zero-length string unless the recipe editor is able to work in conjunction with the host for proper VersionID assignment. Any two recipes in the factory with the same RecipeID and non-zero-length VersionID shall have the same content for the recipe body.	A recipe editor that is not integrated with factory business rules would not be able to determine an appropriate VersionID. The host may populate the VersionID at a later time, for example at the time the recipe is checked-in to the factory recipe management system. If the host does not choose to use the VersionID for a particular recipe, the VersionID will remain a zero-length string. When a VersionID is assigned, the combination of RecipeID and VersionID would provide unique identification of the recipe within the factory.
Security ID	SecurityID	Maximum 16 characters	An identifier which indicates Security Class of the recipe.	Indicates in which Security Class the recipe shall be dealt with. The host shall define SecurityID per actions to be taken with/between the equipment. SecurityID is not an ID for unique identification of recipe.
Type ID	TypeID	Maximum 16 characters	An optional identifier which indicates the type of the recipe. TypeID shall be set to a zero-length string when it is not used.	
Equipment ID	EquipmentID	Maximum 256 characters	An optional identifier that identifies for which equipment this particular recipe is prepared. The EquipmentID shall be set to a zero-length string when it is not used.	

9 Secured Recipe Space Requirements

9.1 *Secured Recipe Space* — Equipment compliant with this Standard shall be compliant with the SRS requirements defined in this section.

9.2 *SRS and Security Class* — SRS is a recipe space which logically discriminates recipes per Security Class identified by SecurityID of each recipe. Each Security Class is deemed to have its own logical recipe space in SRS.

9.3 *Message Requirements* — SRS uses following messages.

9.3.1 Conventional Messages — The messages based on the recipe identification scheme which uses RecipeID (PPID) as an only identifier of recipe. For upward compatibility, access with conventional messages shall be directed to Conventional Recipe Space by applying VersionID= ‘’ (zero length), SecurityID= ‘Conventional’, TypeID= ‘’ (zero length), and EquipmentID= ‘’ (zero length) except for the following cases.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

9.3.1.1 *PJ Creation from the Host* — PJ creation demand from the host shall refer the recipes which belong to the Security Class specified by PEMFlag regarding the recipes which master copies are managed on Recipe Server. The references of recipes which master copies are managed in equipment are not affected by PEMFlag.

9.3.1.2 *SecurityID in Recipe Body* — In case SecurityID (and VersionID as required) is embedded in Recipe Body, the conventional message may be used instead of the equivalent extended message.

9.3.2 *Extended Messages* — The messages which use extended recipe identification scheme (RecipeXID).

Table 11 Secured Recipe Space Definitions

Logical Recipe Space	*Security Class (SecurityID)*	*Messages*		*Definition*	*Note*
		Conventional	*Extended*		
Conventional Recipe Space	'Conventional'	Usable	Usable	Default recipe space. For upward compatibility with current implementation, currently used recipe space is mapped to this Security Class.	
PRC Space	'Production' or 'Production n' In case multiple PRCs are used, numerical character n should be added as 'Production n'	Not usable Except for: PJ creation during PEM On	To be used Except for: PJ creation	Recipe space for PRC. Holds subset of exact copies of PEM Recipes that are managed on Recipe Server.	'Which PRC should be used for which PJ creation' is equipment dependent.
RAC Space	'Remote' or 'Remote n' In case multiple RACs are used, numerical character n should be added as 'Remote n'	Not usable	To be used	Recipe space for Remote Access This space may hold temporary copy of required recipes that are managed on Recipe Server. After a remote access operation, modified or created recipes may be uploaded to Recipe Server and all related recipes used for the remote access shall be deleted from RAC. One RAC Space is used for one Remote Access connection for secure operation.	
Optional Recipe Spaces	'<user defined>'	Not usable	To be used	Recipe space for user defined Recipe Security Class.	May be used to provide isolated recipe spaces to different roles.

9.4 *PEM State and PEMFlag* — PEMFlag determines the PEM State whether the equipment operates in PEM On or PEM Off. PEMFlag holds the Security Class, which is used for the host triggered PJ creation, to indicate PEM State. PEMFlag may be used for protection purpose even in the case SecurityID is directly specified upon PJ creation demand.

9.4.1 *Multiple PEMFlags for Multi-Part Equipment* — In the case the equipment has a capability to divide itself into multiple parts that can operate in PEM On or PEM Off independently, the PEMFlag of the each part may separately be managed.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217

Table 12 PEM State and PEMFlag Definitions

PEM State	*PEMFlag (Security Class to be used for PJ Creation)*	*Definition*	*Usage of Secured Recipe Space*	
			Conventional Recipe Space	*PRC Space*
PEM Off (Default)	'Conventional'	A state for non-mass production operations such as installation, maintenance, process development, etc.	Shall be used for PJ creation for the equipment or the part of equipment controlled by this PEMFlag. May be used for PJ creation for other part of the equipment controlled by associated PEMFlags.	Shall not be used for any PJ creation. May be used for preparations of PRC before PEM On such as Cache Clear operation or Download operation.
PEM On	'Production' or 'Production n'	A state for mass production operations.	Shall not be used for PJ creation for the equipment or the part of equipment controlled by this PEMFlag. May be used for PJ creation for other part of the equipment controlled by associated PEMFlags.	Shall be used for PJ creation for the equipment or the part of the equipment managed by this PEMFlag.

9.5 *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 PEM state of the each part shall separately be managed and the use of PRC shall separately be controlled as shown in the following figure.

9.5.1 *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.

9.5.2 *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.

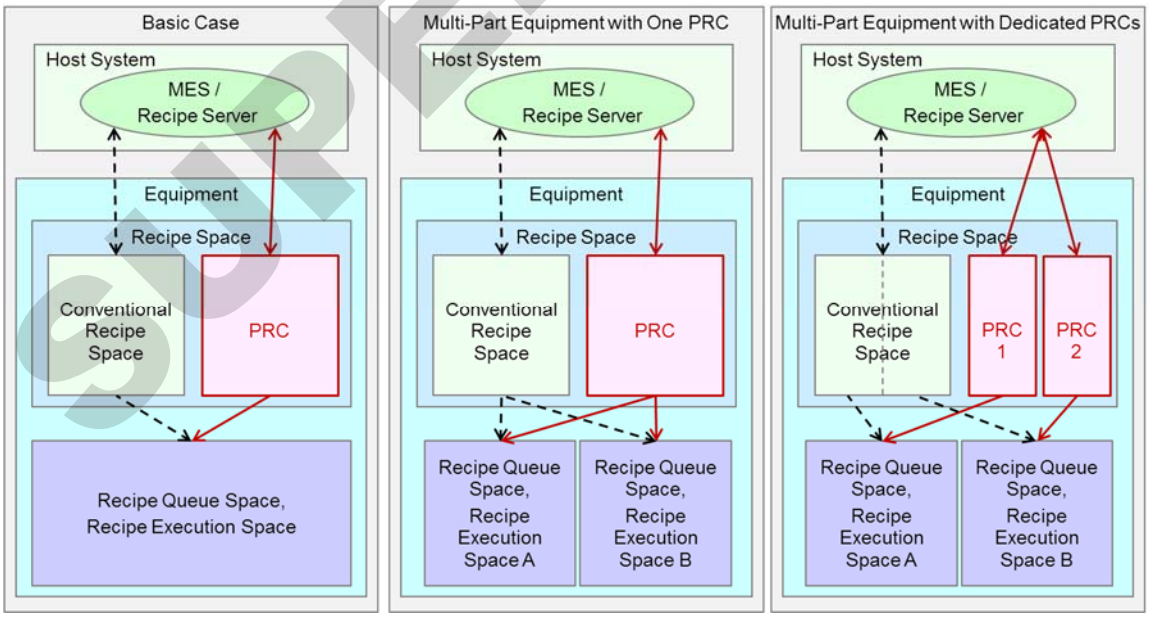

Figure 8
PRC Implementation for Multi-Part Equipment

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E170-1217

9.6 *SRS Object (SRSOBJ) Requirement* — Equipment compliant with SRS function of this Standard shall have SRS Object management capability.

9.6.1 *Definition of SRSOBJ* — The SRSOBJ is a software representation of SRS. Information about SRS is encapsulated as an object. This allows the host to exchange information with the equipment about SRSOBJ by using services defined in SEMI E39 Object Services Standard.

9.7 *SRSOBJ Object Descriptions*

9.7.1 *SRSOBJ Object Instantiation* — Under normal circumstances SRSOBJ object is instantiated by the equipment when the equipment is started up.

9.7.2 *SRSOBJ Object Identifier (ObjID)* — The SRSOBJID is the SRSOBJ Object Identifier. The equipment is responsible for ensuring uniqueness of the SRSOBJID prior to instantiation.

9.7.3 *SRSOBJ Object Destruction* — A SRSOBJ Object reaches the end of its lifecycle when the equipment is shut down.

9.8 *SRSOBJ Object Attribute Definitions* — The following table defines the attributes of SRSOBJ object.

9.8.1 *Who to Maintain the Attributes* — All attributes in the following table are always maintained and updated by the equipment.

Table 13 SRSOBJ Attribute Definition

Attribute Name	*Definition*	*Access*[#1]	*Reqd*	*Form*
ObjID	SRSOBJ object Identifier.	RO	Y	Text. SRSOBJID Numerical text expression of positive integer. ObjID is equipment defined.
ObjType	Object Type.	RO	Y	Text = ‘SRSObject’
SRSSwitch	Enable/Disable entire SRS functionalities. Default is Disabled.	RO	Y	Enumerated: Disabled, Enabled
SecurityClassList	List of implemented Security Classes.	RO	Y	List of SecurityClass (Maximum 16 characters)
PEMFlagList	List of implemented PEMFlags. This attribute is required when PRC is implemented.	RO	Y	List of PEMFlag (Maximum 16 characters) PEMFlag = ‘Conventional’:PEM Off PEMFlag = ‘Production’ or ‘Production n’:PEM On
QueryStat	Status of QueryRecipe. RecipeXIDList under query is held.	RO	Y	RecipeXIDList Zero length when no QueryRecipe takes place
Pre-ExeCheckStat	Status of Pre-ExeCheck. RecipeXID and CheckInformation under check are listed.	RO	Y	List of RecipeXID CheckInformation Zero length when no Pre-ExeCheck takes place

#1 Even though a value may be marked as RO (read only), the initial value for the attribute may be provided by the host.

9.9 *SRSOBJ Services* — This section defines the message services required to support SRSOBJ functionalities. Equipment compliant with SRS function of this Standard shall have following services.

9.9.1 *SRSOBJ Service Message Description* — The following table is a list of SRSOBJ services.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E170-1217