

SEMI E170-0520

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 December 12, 2019 and January 30, 2020. Available at www.semiviews.org and www.semi.org in May 2020; originally published July 2014; previously published April 2019.

1 Purpose

1.1 The purpose of this Standard is to provide a standardized 'secure 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.

2 Scope

2.1 The scope of this Standard is to define a secure and efficient recipe operation mechanism intended to be used as a foundation of RMS.

2.2 The scope of this Standard is to define an extended recipe identification scheme and its usage that identifies recipes from multiple aspects including uniqueness and security.

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 None.

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 — Specification for Object Services Standard: Concepts, Behavior, and Services

SEMI E40 — Standard for Processing Management

SEMI E87 — Specification for Carrier Management (CMS)

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

5 Terminology

5.1 Abbreviations and Acronyms

5.1.1 COP — Conventional Operation

5.1.2 CRS — Conventional Recipe Space

5.1.3 EOP — Equipment Operation

5.1.4 FOP — Factory Operation

5.1.5 FOUP — front opening unified pod

5.1.6 GEM — generic equipment model

5.1.7 MES — manufacturing execution system

5.1.8 PJ — process job

5.1.9 *POP* — PRC Operation

5.1.10 *PRC* — Production Recipe Cache

5.1.11 *RMS* — Recipe Management System

5.1.12 *ROP* — RMS Operation

5.1.13 *SPOC* — Single Point of Control

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 *Check-in* — an action to register a recipe into Repository in RMS or equipment. Normally, uniqueness management of the recipes in Repository is done upon Check-in.

5.2.6 *Check-out* — an action to copy a recipe from Repository in RMS or equipment. Normally, security management of recipes in Repository is done upon Check-out.

5.2.7 *collection event* — an event (or grouping of related events) on the equipment that is considered to be significant to the host. [SEMI E30]

5.2.8 *Conventional Recipe Space (CRS)* — 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.

5.2.9 *Equipment Recipe* — a recipe the base part of RecipeXID was assigned by equipment.

5.2.10 *extended recipe identifier (Recipe XID)* — an identifier of a recipe which is extended with additional identifiers.

5.2.11 *RMS Recipe* — a recipe the base part of RecipeXID was assigned by RMS.

5.2.12 *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]

5.2.13 *Master Copy* — a recipe approved as the original and stored in Repository.

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

5.2.15 *New Copy* — a recipe newly created or modified and not registered as Master Copy yet.

5.2.16 *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]

5.2.17 *Offline* — a lowest control level of equipment. ‘In the lowest level (OFF-LINE), the equipment allows no host control and only very limited information. [SEMI E30]’

5.2.18 *Online Local* — a middle control level of equipment. ‘The middle level (LOCAL) allows the host full access to information, but places some limits on how the host can affect equipment operation. [SEMI E30]’

5.2.19 *Online Remote* — a highest control level of equipment where remote execution of process by the host is possible. ‘In the highest level (REMOTE), the host may control the equipment to the full extent possible. [SEMI E30]’

5.2.20 *Operation Copy* — a collective name of recipes prepared for SRO. Operation Copy includes Reference Copy and New Copy.

5.2.21 *PJ Creation* — an action which creates Process Job defined in SEMI E40.

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

5.2.23 *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]

5.2.24 *Production Recipe* — a recipe which is used for host triggered PJ creation. Production Recipes are not only recipes for mass production lots but also recipes for engineering lots, etc., if the recipes are used by host triggered PJ creation. Any recipe that is prepared to be used for host triggered PJ creation shall be considered as a Production Recipe.

5.2.25 *Production Recipe Cache (PRC)* — a logical recipe space which holds Production Recipes to be used for PJ creation demand from the host.

5.2.26 *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]

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

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

5.2.29 *Recipe Management System (RMS)* — a factory information system that manages recipes in the factory and is a part of the host. The term RMS may be replaced with the term host in some cases.

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

5.2.31 *Reference Copy* — a recipe copied from Master Copy for SRO.

5.2.32 *Secured Recipe Operation (SRO)* — a control function which provides recipe operation secured by discriminating recipes per their Recipe XID and by authorizing user who can execute operation of the recipes.

5.2.33 *Repository* — a secured recipe space which stores master copies of recipes.

5.2.34 *Single Point of Control (SPOC)* — a control scheme in which the only one communication contact point controls the whole equipment.

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

5.2.36 *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]

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 Req'd 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

<i>Attribute Name</i>	<i>Definition</i>	<i>Access</i>	<i>Req'd</i>	<i>Form</i>

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 each state by 'No.', 'Name', and 'Definition Overview'. 'No.' (Number) corresponds to the state number in the associated state diagram. 'Name' defines a name of the state. 'Definition Overview' provides an outline of the definition of the state.

6.2.3.2 *Definition of Transition* — Columns under the 'Transition' column define each transition by 'No.', 'Name', 'Trigger Overview', 'Action Overview', 'New State', and 'Comments'. 'No.' (Number) corresponds to transition number in the associated state diagram. 'Name' defines a name of the transition. 'Trigger Overview' and 'Action Overview' provide outlines of the trigger and action accordingly. 'New State' defines a state number to move after the completion of the transition by pointing one of the states defined in the state definition in left side of the table. The 'Comments' column may be used to put comments to each transition or its 'From' state.

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

Table 2 State Model Definition Table

State			Transition					
No.	Name	Definition Overview	No.	Name	Trigger Overview	Action Overview	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	
			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, Mnemonic, Trigger, Actions, and Comments. The column ‘Trigger’ for the transition occurs while in the ‘Previous’ State. The column ‘Actions’ 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	Mnemonic	Trigger	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.

6.4 Transactions and 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 *Transaction Description* — A transaction description table defines the parameters used in a transaction, as shown in the following table:

Table 5 Transaction Description Table

<i>Transaction Name</i>	<i>Originated by</i>	<i>Type</i> ^{#1}	<i>Req</i> ^{#2}	<i>Description</i>

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

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

6.4.3 *Transaction 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 Transaction Parameter Definition Table

<i>Parameter Name</i>	<i>Form</i>	<i>Description</i>

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

6.4.4 *Service Message Definition and Event Message Definition* — A service message definition and an event message definition table define the parameters used in a service messages and event messages, and also, describes each message and its cause/effect to the equipment, as shown in the following table:

Table 7 Service Message Definition Table

<i>Message Parameter</i>	<i>Req/Ind</i>	<i>Rsp/Conf</i>	<i>Description</i>

Table 8 Event Message Definition Table

<i>Message Parameter</i>	<i>Req/Ind</i>	<i>Rsp/Conf</i>	<i>Description</i>

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 9 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 10 Variable Data Definitions

<i>Variable Name</i>	<i>Description</i>	<i>Type</i>	<i>Access</i>	<i>Comment</i>

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 *Targeted RMS* — The RMS that this Standard intends to support is an RMS with the following characteristics that require cooperation of equipment. Note that this Standard does not define encryption/decryption as the area should be defined by separate Standard which can be applied not only for this Standard but for others.

7.2.1 *Recipe Security Management from Leakage* — A RMS which manages recipe security from leakage.

- Allow to hold only the requisite minimum (or optimum) number of recipes on the equipment to minimize the risk
- Delete recipes on equipment by time or by disconnection from the host, etc.

7.2.2 *Recipe Operation Management* — A RMS which manages not only recipe itself but also operations of recipes such as:

- Recipe editing, recipe execution, etc.
- Manual by user or Automated by MES/RMS

7.2.3 *Recipe Security Management Among Recipe Operations* — A RMS which manages recipe security among multiple recipe operations by multiple users.

- User authentication by equipment or by RMS for individual recipe operations
- Recipe isolation among recipe operations, etc.

7.2.4 *Practical Recipe Uniqueness Management* — A RMS which provides user-friendly recipe uniqueness management.

- Separate uniqueness management of recipes for RMS Recipes and for Equipment Recipes
- Human readable revision codes, etc.

7.2.5 *Efficient Operation* — A RMS which reduces system load by optimizing transaction frequency of recipes.

- Cache operation by either RMS or equipment
- Reduction of recipe download frequency to reduce recipe check load of equipment after download of recipe, etc.

7.2.6 *RMS-Equipment Direct Communication for Recipes* — To make factory information control system more modular, RMS communicates directly with equipment to perform recipe operations. To be compatible with SPOC, the host side (e.g., something like an Equipment Interface) should maintain redirection of the communication to MES or RMS.

- Prepare recipe transactions for RMS

7.2.7 *Single Point of Recipe Management (SPORM)* — The master copies of factory-managed recipes are managed by RMS.

- Only temporary copies are in the equipment to support seamless execution of production.
- The master copies of Equipment Recipes (for maintenance etc.) are managed by the equipment. Backup copies may be held in RMS.

7.3 *Intended Application Image* — The intended application image of this Standard is to construct RMS which provides ‘Secure and efficient’ recipe operation mechanism. An example image of intended application of this Standard is shown in the following figure.

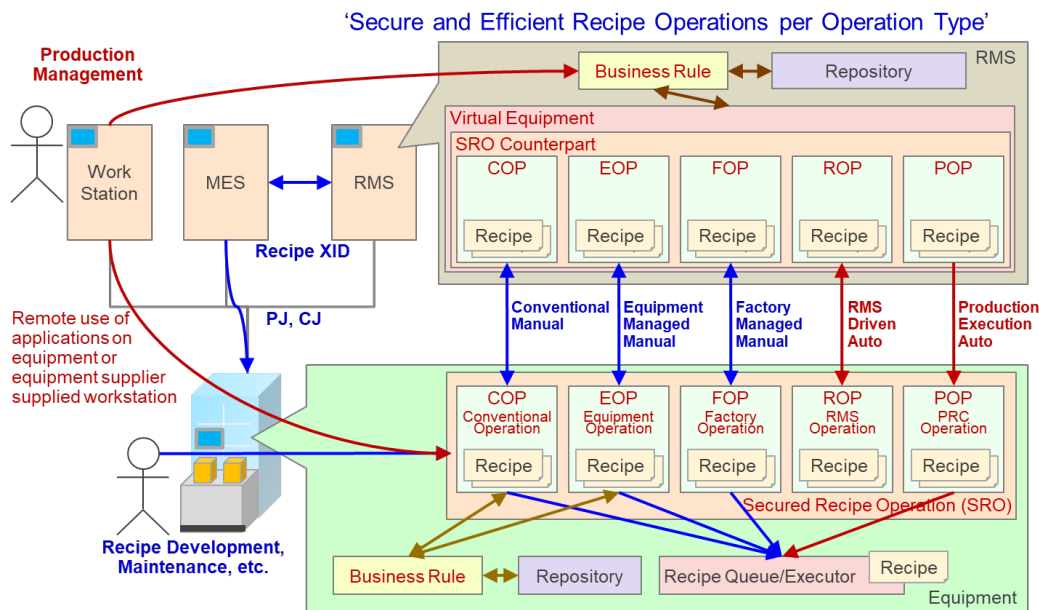


Figure 1
Example Image of Intended Application of This Standard

7.3.1 *Intended Application on Equipment* — This Standard expects equipment to have the following elements.

7.3.1.1 *Secured Recipe Operation (SRO) per Operation Type* — In order to share recipes securely and efficiently between RMS and equipment, securely separated recipe operation called SROs which have operation types shown in the table ‘SRO and SRO Counterpart’ are provided. The SROs are separated from each other for recipe operation security. The user is authenticated to log in an SRO. SRO may be used from Work Station in the factory.

7.3.1.2 *Repository (in Equipment)* — Master Copies of Equipment Recipes are held in Repository in the equipment.

7.3.1.3 *Business Rule (of Equipment)* — Check-in/out is controlled based on the Business Rule of the equipment.

7.3.1.4 *Recipe Queue/Executor* — Recipes to be executed manually or automatically are securely copied to Recipe Queue/Executor.

7.3.2 *Intended Application on RMS* — This Standard expects RMS to have the following elements.

7.3.2.1 *Virtual Equipment (VE)* — A mapping of equipment which may have the following functions.

- Backup space of the recipes in the equipment.
- Counterpart of operations (SRO Counterpart) in the equipment.
- Equipment recipe applications such as recipe editor.

7.3.2.2 *Repository (in RMS)* — Master Copies of RMS Recipes are held in Repository in RMS.

7.3.2.3 *Business Rule (of RMS)* — Check-in/out is controlled based on the Business Rule of RMS.

Table 11 SRO and SRO Counterpart

	<i>COP</i>	<i>EOP</i>	<i>FOP</i>	<i>ROP</i>	<i>POP</i>
Operation Type	Conventional Operation	Equipment Operation	Factory Operation	RMS Operation	PRC Operation
SRO Counterpart (In RMS)	Backup of recipes under COP. Deploy recipes in COP to other equipment.	Backup of recipes under EOP. Deploy recipes in EOP to other equipment.	Provide recipes to be used for FOP, and collect recipes worked out under FOP.	Demand recipe operation to ROP, and collect recipes worked out under ROP.	Allocate recipes to be used by PJ creation regardless of connection state, and provide operability of 'What you have done on the Recipe Server is what you will have on the equipment'.
SRO (In Equipment)	An operation which handles the recipes in CRS (Conventional Recipe Space).	An operation which handles the recipes managed by the equipment.	An operation which handles the recipes managed by RMS.	An operation which automatically handles recipes by the order from RMS.	An operation which allocates required recipes for PJ creation on equipment.

7.4 *Base Concepts* — This Standard is designed based on the following concepts.

7.4.1 *Overall Concept* — The following figure shows the overall concept of this Standard.

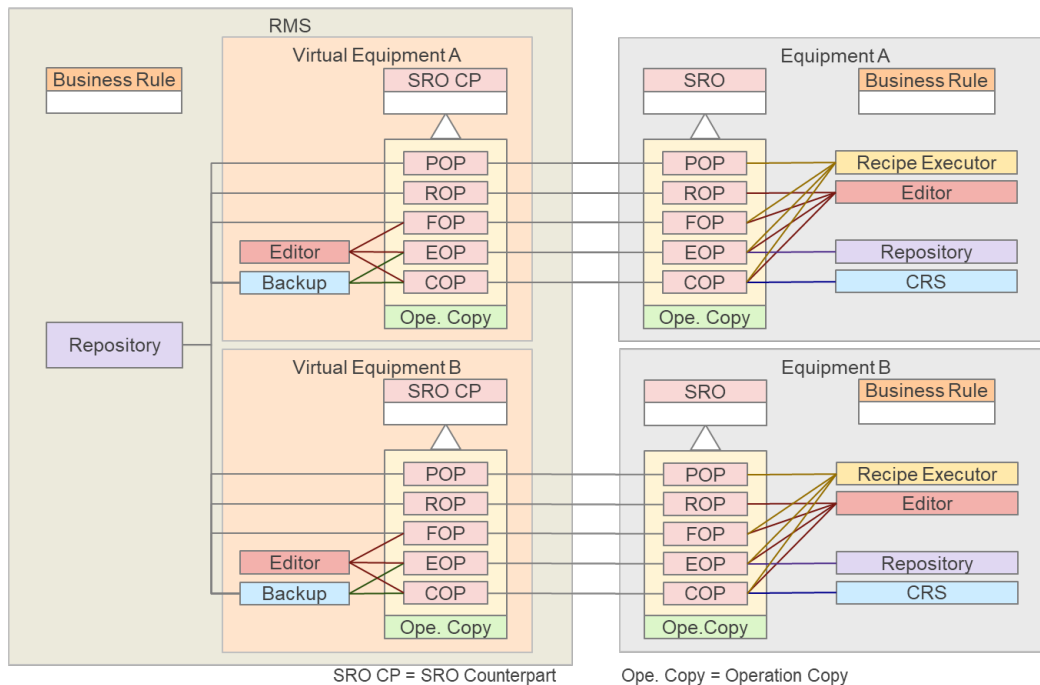


Figure 2
Overall Concept

7.4.1.1 *Copy Management* — Management of the original and rental of recipes is done by ‘Electronic Library Model’. Here is an analogical explanation by using Library and Book.

- *Registration of a Book to the Library* — An Operation Copy to be a Master Copy is uniquely named and checked-in the Repository by the Business Rule.
- *Lending a Book from the library* — A copy of Master Copy may be checked-out from the Repository as a Reference Copy marked with an SRO name and lent to the SRO (or SRO Counterpart) through the Business Rule.
- *Use of the Book* — The Reference Copy may be used by the SRO (or SRO Counterpart) for process execution or modified recipe creation.
- *Creation of New Book* — A recipe may be created from scratch or from Reference Copy and may be managed uniquely under the SRO (or SRO Counterpart).
- *Management of Books Lent Out or Created* — A Reference Copy and New Copy are managed by the SRO (or SRO Counterpart) as members of Operation Copies.
- *Return of the Book Lent Out from the Library* — The Reference Copy is deleted by the SRO (or SRO Counterpart) after use. Notification to the Repository (Library) may be needed.

7.4.1.2 *Uniqueness Management* — Uniqueness of RecipeXID is managed as below.

- *Uniqueness of Master Copy* — Uniqueness of Master Copy is managed in Repository by Business Rule within CopyID.
- *Uniqueness of Operation Copy* — Uniqueness of Operation Copies are separately managed by SROs per OperationID.

7.4.1.3 *Security Management* — Copies of recipes are managed by a ‘Electronic Library Model’.

- *Master Copies in Repository* — Master Copies of recipes are managed in Repository in RMS and equipment.
- *Operation Copies Under SROs* — Operation Copies of recipes are separately managed under SROs.
- *Management by Business Rule* — Both Repository and SROs are managed by Business Rule.

7.4.1.4 *Operation Management* — Recipe operations are managed by using SRO.

- *Activation of SRO* — SROs are activated by Business Rule.
- *Operation Copy Assignment* — Operation Copies for each SRO are assigned by the Business Rule.
- *Types of SROs* — Different type of SRO for different operation for security.
- *Deployment Route of Equipment Recipe* — Equipment Recipe developed locally in equipment may be deployed through EOP.

7.4.1.5 *Backup Management* — VE in RMS manages the backup of the equipment such as:

- *Backup of the Operation Copies in SRO at Any Time* — The Operation Copies in SRO may be saved to SRO Counterpart at any time as appropriate.
- *Backup of the Operation Copies in SRO Upon Deletion of SRO Counterpart* — The Operation Copies in SRO Counterpart may be saved somewhere secure in VE upon deletion of SRO Counterpart.

7.4.1.6 *Deployment Consideration* — For the equipment which does not have full compliance with the concept of this Standard, this Standard provides the following function.

- *COP* — COP provides similar accessibility to the recipes with conventional identification in CRS. With this function, those recipes may be backed up by RMS or transferred to other equipment that is compatible with the recipe.

7.4.1.7 *Operation Type Conformance* — SRO specifications per operation types.

7.4.2 *Introduction of RecipeXID* — In order to manage recipes securely and conveniently, this Standard introduces Extended Recipe ID called RecipeXID.

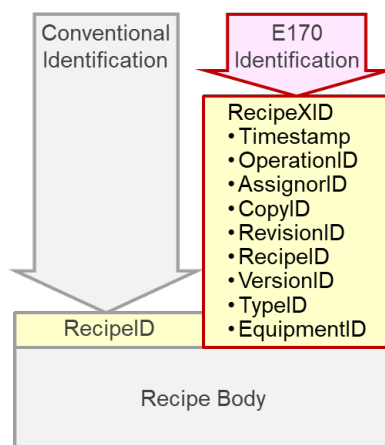


Figure 3
Introduction of Recipe XID

7.4.3 *Introduction of Secured Recipe Operation (SRO)* — To deal recipes securely, this Standard introduces SRO which logically discriminates recipes per their Recipe XID and authenticates user who can use those operations.

7.4.3.1 *SROUser Management* — SRO manages a user who uses the SRO. SROUser of COP, EOP, and FOP is a human user, of ROP is RMS, and of POP is MES.

7.4.3.2 *Recipe Management* — SRO manages recipes which have the same OperationID with the SRO.

7.4.3.3 *RMS Connection Management* — SRO manages the connection to RMS.

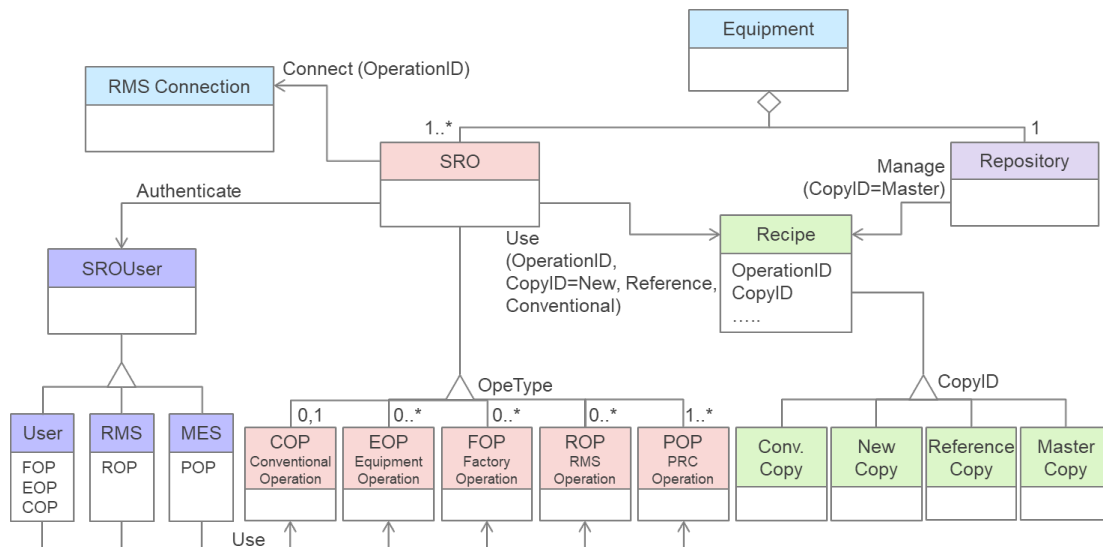


Figure 4
Image of Secured Recipe Operation

7.4.4 *Introduction of Production Recipe Cache (PRC) Operation* — For efficient production recipe transactions between RMS and equipment, this Standard introduces PRC Operation (POP).

7.4.4.1 *Operation Modes of PRC* — To be upward compatible with conventional operations, this Standard provides the following PRC Operation Modes. 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 transactions and recipe check overhead at equipment.

- *Pre-Download Mode (Optional)* — RMS has full control capability of PRC content for RMS-controlled cache operation. RMS downloads the required recipes at the optimum timing prior to the associated PJ creation request and may retain the recipes in the cache for the next use or may delete them from the cache, considering the balance of security and recipe transaction efficiency.
- *Cache Mode (Optional)* — Equipment-controlled cache operation. Equipment keeps allowed numbers of recently used recipe for allowed duration in PRC. Equipment looks up PRC for the required recipes for PJ creation and queries the recipes to RMS if they are not in PRC.
- *Full Download Mode (Optional)* — RMS downloads required recipes every time just before MES demands a PJ creation and deletes the recipes immediately after the use. This mode gives recipe check overhead to equipment every time.
- *Full Query Mode (Optional)* — Equipment queries required recipes every time upon PJ creation and deletes them after use.

7.4.4.2 *Pre-Execution Check Operation (Optional)* — 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 cases.

- *Pre-Exe Check* — Equipment uploads check information of the required recipes to RMS for consistency check upon PJ creation, and RMS responds with an affirmative message or with a negative message which contains correct recipes.

7.4.5 Linked Recipe Concept — To be compatible with current implementations, the same linkage scheme currently used may be used within the SRO.

7.4.5.1 Linkage to Hidden Recipes — Equipment may have recipes hidden from the recipe management scheme compliant with this Standard, such as subcomponent management recipes. Those recipes may be linked in the way currently implemented.

7.4.5.2 ProcessLink Recipe — This Standard introduces ProcessLink Recipe which provides linkage information of a set of recipes in the standardized way.

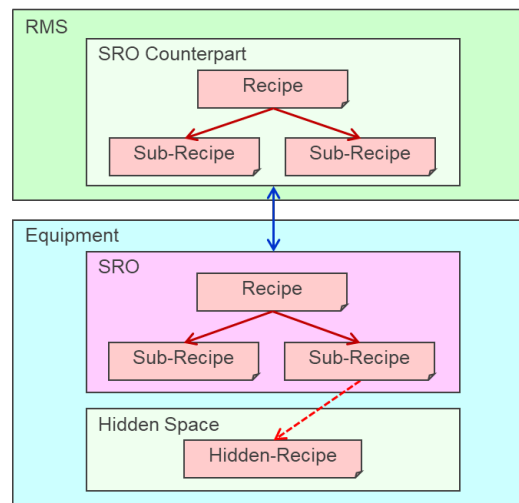


Figure 5
Linked Recipe Concept

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

7.4.7 Multi-Part Equipment — In the case the equipment has a capability to divide itself into multiple parts that can independently operate in Offline, Online Local, or Online Remote state, multiple SROs may be assigned to support those parts independently.

7.5 List of Capabilities — The following table shows the capabilities this Standard provides.

Table 12 List of Capabilities

Capability	Section	Keyword	Comment
Recipe Identification	<i>Extended Recipe Identification Requirements</i>	-	
Extended recipe identification	<i>RecipeXID Definition</i>	RecipeXID	
Timestamp the recipe is finally revised	-	Timestamp	
Operation the recipe is being used	-	OperationID	
Base Part is RMS-assigned or equipment-assigned	<i>AssignorID Definition</i>	AssignorID	
Type of copy	<i>CopyID Definition</i>	CopyID	
Revisions done by SRO	-	RevisionID	
Main ID of the recipe	-	RecipeID	
Version ID within the same RecipeID	-	VersionID	
Type of recipe	<i>TypeID Definition</i>	TypeID	
Equipment the recipe is tuned for	-	EquipmentID	

<i>Capability</i>	<i>Section</i>	<i>Keyword</i>	<i>Comment</i>
Type of recipe: ProcessLink, ProcessMain, ProcessUnit, Activity, RecipeJob	<i>TypeID Definition</i>	TypeID	
RMS Recipe and Equipment Recipe	<i>AssignorID Definition (RMS Recipe and Equipment Recipe)</i>	AssignorID, RASIGN, EASIGN	
Copy management	<i>CopyID Definition</i>	CopyID, Repository	
Master Copy	-	MCOPY	
Reference Copy	-	RCOPY	
New Copy	-	NCOPY	
Conventional Copy	-	CCOPY	
Operation Copy	<i>Operation Copy Definition</i>	-	
How to use OperationID, AssignorID, CopyID, and RevisionID	<i>Usage of the Management Part of RecipeXID</i>	Management Part of RecipeXID	
Additional Recipes	<i>Additional Recipe Format Definition</i>	-	
ProcessLink Recipe	<i>ProcessLink Recipe Format</i>	ProcessLink Recipe	
Activity Recipe	<i>Activity Recipe Format</i>	Activity Recipe	
Secured Copy Management Rule	<i>Secured Recipe Management Requirements</i>	-	
Copy management rules	<i>Copy Management Rule Throughout RMS and Equipment</i>	Repository Check-in, Check-out	
Use of Operation Copy	<i>Use of Operation Copy</i>	Creation, execution, etc.	
OpeType Dependent Specifications	<i>OpeType Dependent Specifications</i>	OpeType	
Multi-Part Equipment Compatibility (Optional)	<i>Multi-Part Equipment Compatibility</i>	Multiple POPs	
Secured Recipe Operation	<i>Secured Recipe Operation (SRO) Requirements</i>	-	
Security management	-	-	
Authentication of recipe operation user in equipment	<i>Local Operation</i>	EQUserID, EQPassword	
Authentication of recipe operation use for RMS	<i>Opening of Connection</i>	RMSUserID, RMSPassword	
Automatic close by timer	<i>Local Operation</i>	AutoClose	
Isolation among operations		OperationID	
Recipe Operation Type	<i>Operation Type (OpeType) Definition</i>	OpeType	
Use of recipes in CRS (Conventional Recipe Space)	-	COP	
Manual operation (edit, execute, etc.) of Equipment Recipes and Equipment Recipe candidates (including copies of RMS Recipes provided by RMS).	-	EOP	
Manual operation (edit, execute, etc.) of RMS Recipes and RMS Recipe candidates (including copies of Equipment Recipes provided by the equipment).	-	FOP	
RMS driven recipe operations in equipment (such as recipe version conversion due to equipment version change)	-	ROP	
MES driven recipe execution (by PJ)	-	POP	

<i>Capability</i>	<i>Section</i>	<i>Keyword</i>	<i>Comment</i>
Operation Copy management upon disconnection or termination of SRO	<i>Closing of Connection</i>	-	
Automatic backup to VE in RMS	-	AutoPost	
Automatic clear	-	AutoClear	
Multi-Part Equipment Compatibility	<i>Multi-Part Equipment Compatibility (Optional)</i>	Dedicated SROs per part	
SROOBJ Transactions	<i>SROOBJ Transactions</i>	-	
RMS gets attributes of SRO	<i>GetSROAttributes Transaction</i>	GetAtt of E39 OSS	
RMS sets attributes of SRO	<i>SetSROAttributes Transaction</i>	SetSROAttributes Service	
RMS gets existing OperationID in the equipment which belongs to the OpeType of the SRO.	<i>GetOperationIDList Transaction</i>	GetOperationIDList Service	
Equipment requests RMS to open a Connection.	<i>OpenConnection Transaction</i>	OpenConnection Event	
Equipment requests RMS to close the Connection.	<i>CloseConnection Transaction</i>	CloseConnection Event	
RMS clears all recipes which have the specified OperationID in the equipment.	<i>ClearOperation Transaction</i>	ClearOperation Service	
RMS gets a list of RecipeXID which has the specified OperationID.	<i>GetRecipeXIDList Transaction</i>	GetRecipeXIDList Service	
RMS deletes specified recipes in the equipment.	<i>DeleteRecipe Transaction</i>	DeleteRecipe Service	
RMS writes recipes down to the equipment.	<i>WriteRecipe Transaction</i>	WriteRecipe Service	
RMS reads specified recipes from the equipment.	<i>ReadRecipe Transaction</i>	ReadRecipe Service	
RMS requests the equipment to verify specified recipes	<i>VerifyRecipe Transaction</i>	VerifyRecipe Service VerifyRecipe Event	
Equipment requests and gets a list of RecipeXID (RecipeXIDList) which has specified OperationID to RMS.	<i>QueryRecipeXIDList Transaction</i>	QueryRecipeXIDList Event	
Equipment requests and gets specified recipes from RMS.	<i>QueryRecipe Transaction</i>	QueryRecipe Event QueryRecipe Service	
Equipment posts recipes to RMS.	<i>PostRecipe Transaction</i>	PostRecipe Event	
Process Execution	-	POP	
Recipe allocation to equipment for MES driven recipe execution	-	-	
Recipe assignment to equipment regardless of its readiness	<i>Preparation of Operation Copies in POP Counterpart</i>	Virtual Equipment SRO Counterpart POP	
Download recipes by RMS before PJ demand from MES	<i>Modes of PRC</i>	Pre-Download Mode	
Optimum recipe holding on equipment	<i>Modes of PRC</i>	Cache Mode	
Download recipe by RMS every time just before PJ demand	<i>Modes of PRC</i>	Full Download Mode	
Query recipe by equipment every time upon PJ creation	<i>Modes of PRC</i>	Full Query Mode	
Limited number of recipes on equipment for security	<i>Maximum Number Management of Recipes Retained in PRC (Optional)</i>	MaxNumberList	

Capability	Section	Keyword	Comment
Limited time to keep recipes on equipment for security	<i>Maximum Time Management of Recipes Retained in PRC (Optional)</i>	MaxTime	
POP Transactions (Optional)	<i>PRC Operation Transactions (Optional)</i>	-	
RMS gets attributes for PRC Operation	<i>GetPRCAAttributes Transaction</i>	GetPRCAAttributes Service	
RMS sets attributes for PRC Operation	<i>SetPRCAAttributes Transaction</i>	SetPRCAAttributes Service	
RMS specifies recipes required for subsequent PJ creation message that does not support RecipeXID.	<i>PreSpecifyRecipe Transaction</i>	PreSpecifyRecipe Service PreSpecifyRecipe Event	
Equipment queries RecipeXID list to be used for the specified PJ.	<i>QueryPJRecipeXIDList Transaction (Optional)</i>	QueryPJRecipeXIDList Event	
Equipment sends Pre-Execution Check information to RMS, and RMS responds with the result (and correct recipes when the check result is NG).	<i>Pre-ExeCheck Transaction (Optional)</i>	Pre-ExeCheck Event Pre-ExeCheck Service	

8 Extended Recipe Identification Requirements

8.1 *RecipeXID Definition* — In order to identify recipes from multiple aspects, this Standard introduces Extended Recipe ID called RecipeXID.

8.1.1 *Image of RecipeXID Structure* — The following figure shows the image of RecipeXID structure.

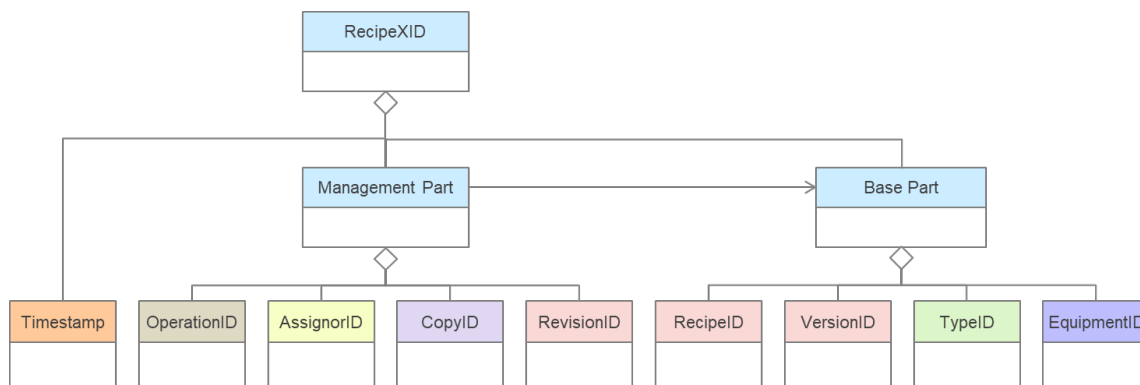


Figure 6
Image of RecipeXID Structure

8.1.2 *Component Definitions of RecipeXID* — The following table defines the components of RecipeXID.

Table 13 RecipeXID Component Definition

<i>ID Name</i>	<i>Mnemonic</i>	<i>Form</i>	<i>Definition</i>	<i>Note</i>
Timestamp	Timestamp	Text.	The time any component of RecipeXID is changed ➤ Use 16-byte format YYYYMMDDhhmmsscc	
Management Part				
Operation ID	OperationID	Maximum 16 characters	An identifier that is used to indicate which Operation the recipe belongs to. ➤ OperationID starts with three characters COP, EOP, FOP, ROP, or POP per OpeType. Actual OperationID is determined in the way specified per OpeType. ➤ RRM (RMS Repository-managed) and ERM (equipment Repository-managed) are used for the recipes in Repository in RMS and equipment, accordingly.	
Assignor ID	AssignorID	Enumerated: RASIGN, EASIGN	An identifier which indicates who assigned the Base Part of RecipeXID. RASIGN = RMS assigned EASIGN = Equipment assigned See the section ' <i>AssignorID Definition (RMS Recipe and Equipment Recipe)</i> ' for detail.	See the section ' <i>Usage of the Management Part of RecipeXID</i> ' for detail.
Copy ID	CopyID	Enumerated: MCOPY, RCOPY, NCOPY, CCOPY	An identifier which indicates which type of copy the recipe is. MCOPY = Master Copy RCOPY = Reference Copy NCOPY = New Copy CCOPY = Conventional Copy	See the section ' <i>CopyID Definition</i> ' for detail.
Revision ID	RevisionID	Maximum 256 characters	An identifier which identifies that the recipe is revised in SRO or SRO Counterpart. ➤ A string, which starts with R for revision by RMS or starts with E for revision by equipment, shall be added separated by '.' upon each revision. (e.g., Erevision1.Rrevision1.Erevision2) ➤ RevisionID shall be set to a zero-length string upon check-in to Repository as Master Copy. ➤ RevisionID shall be set to a zero-length string when it is not used.	See the section ' <i>Usage of the Management Part of RecipeXID</i> ' for detail.
Base Part				
Recipe ID	RecipeID	RecID which is defined in SEMI E40	Main identifier of recipe. RecipeID shall be unique under the same combination of OperationID, CopyID, and AssignorID.	RecID may not always be a unique identifier.
Version ID	VersionID	Maximum 256 characters	An additional identifier that can be combined with the RecipeID to uniquely identify the recipes which are different versions under the same RecipeID. ➤ Multiple recipes which have the same RecipeID shall uniquely be managed by VersionID. ➤ VersionID shall be assigned by Assignor, based on the Business Rule. ➤ VersionID shall be set to a zero-length string when it is not used.	
Type ID	TypeID	Enumerated: See the table ' <i>TypeID Definition</i> '.	An additional recipe identifier which indicates a type of the recipe. See the section ' <i>TypeID Definition</i> ' for detail.	

ID Name	Mnemonic	Form	Definition	Note
Equipment ID	EquipmentID	Maximum 256 characters	An additional recipe identifier that indicates equipment which the recipe is prepared or tuned for. ➤ EquipmentID shall be set to a zero-length string when it is not used.	

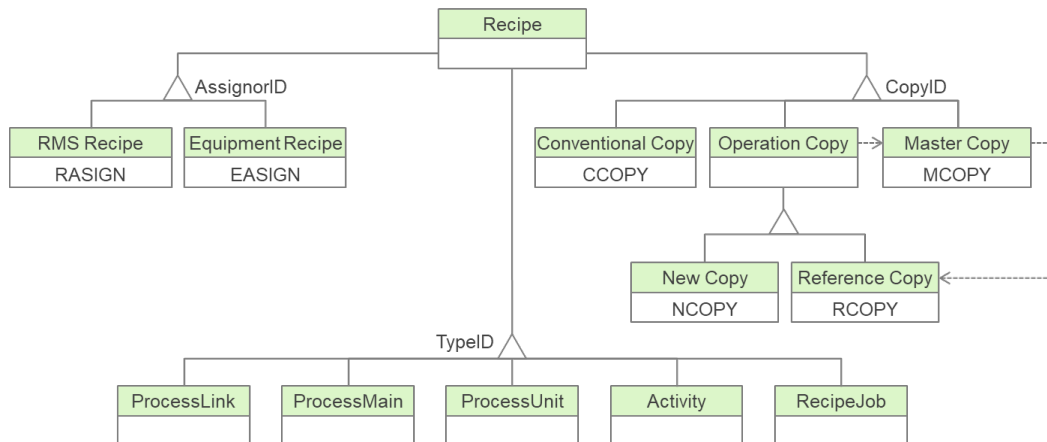


Figure 7
Recipe Classifications

8.1.2.1 *AssignorID Definition (RMS Recipe and Equipment Recipe)* — The following table shows the definitions of AssignorID (RMS Recipe and Equipment Recipe).

Table 14 AssignorID Definition (RMS Recipe and Equipment Recipe)

AssignorID	Recipe	Definition	Note
RASIGN	RMS Recipe	Base Part of RecipeXID is assigned by RMS. RecipeXID of RMS Recipe is uniquely assigned within Master Copy of RMS based on the Business Rule of RMS.	
EASIGN	Equipment Recipe	Base Part of RecipeXID is assigned by the equipment. RecipeXID of an Equipment Recipe is uniquely assigned within Master Copy of the equipment based on the Business Rule of the equipment.	

8.1.2.2 *CopyID Definition* — The following table shows the CopyID definition.

Table 15 CopyID Definition

Type of Copy	CopyID	Definition	Note
Master Copy	MCOPI	A copy which is authorized and checked-in to the Repository.	
Reference Copy	RCOPY	A copy which is copied from Master Copy for the reference and modification in SRO and SRO Counterpart.	
New Copy	NCOPY	A copy which is created and is not checked-in the Repository yet.	
Conventional Copy	CCOPY	A copy which is in CRS. For COP use Only.	

8.1.2.3 *TypeID Definition* — The following table shows the TypeID definition.

Table 16 TypeID Definition

<i>Enumeration</i>	<i>TypeID</i>	<i>Definition</i>	<i>Note</i>
0	NotUsed	Not Used	-
1	ProcessLink	A type of the recipe which contains a list of recipes to be used for process execution.	-
2	ProcessMain	A type of the recipe which is in the top of linked recipe and contains information which controls ProcessUnit recipes.	Format is equipment defined
3	ProcessUnit	A type of the recipe which contains process information for a process unit.	Format is equipment defined
4	Activity	A type of recipe which contains the information from RMS to FOP user.	-
5	RecipeJob	A type of recipe which contains the information from RMS to the equipment for ROP.	The format is not defined in this revision.
6~63	-	Reserved for this Standard.	-
64~	-	Open for users	-

8.1.3 *Operation Copy Definition* — The copies used for an Operation includes the copies shown below.

Table 17 Operation Copy

<i>Type of Copy</i>		<i>AssignorID</i>	<i>CopyID</i>	<i>Definition</i>	<i>Note</i>
Operation Copy				A copy type of recipe which is being used for the operation identified by OperationID. Operation Copy includes the following copies based on OpeType definitions.	The SRO which is using the Operation Copy is identified by the same OperationID.
RMS	Reference Copy	RASIGN	RCOPY	A copy which is made from Master Copy of RMS Recipe for the reference and modification. For EOP, FOP, ROP, and POP.	These copies may be provided by RMS Business Rule per OperationID.
	New Copy	RASIGN	NCOPY	New Copy which is created by RMS. For EOP, FOP, ROP, and POP.	
Equipment	New Copy	EASIGN	NCOPY	New Copy which is created by the equipment. For EOP, FOP, and ROP.	
	Reference Copy	EASIGN	RCOPY	A copy which is made from Master Copy of Equipment Recipe for the reference and modification. For EOP, FOP, and ROP.	

8.1.4 *Usage of the Management Part of RecipeXID* — The following tables show the definition of the usage.

- The following table shows the possible combinations, and actual use of each combination is limited per OpeType.
- This Standard prohibits some check-in paths depending on OpeType.

Table 18 Usage of the Management Part of RecipeXID (Except for COP)

Original RecipeXID			Operation by ^{#1}	New RecipeXID					Note
OperationID	AssignorID	CopyID		OperationID	AssignorID	CopyID	RevisionID	Base Part	
From any combination or scratch			Creation by RMS	XXXnnnn	RASIGN	NCOPY	Set Rnnn	Newly set	
XXXnnnn	RASIGN	NCOPY	Edit by RMS	XXXnnnn	RASIGN	NCOPY	Add Rnnn	No change	
XXXnnnn	RASIGN	NCOPY	Edit by equip.	XXXnnnn	RASIGN	NCOPY	Add Ennn	No change	
XXXnnnn	RASIGN	NCOPY	Check-in by RMS	RRM	RASIGN	MCOPY	Not used	May change	
XXXnnnn	RASIGN	NCOPY	Check-in by equip.	ERM	EASIGN	MCOPY	Not used	May change	
From any combination or scratch			Creation by equip.	XXXnnnn	EASIGN	NCOPY	Set Ennn	Newly set	
XXXnnnn	EASIGN	NCOPY	Edit by equip.	XXXnnnn	EASIGN	NCOPY	Add Ennn	No change	
XXXnnnn	EASIGN	NCOPY	Edit by RMS	XXXnnnn	EASIGN	NCOPY	Add Rnnn	No change	
XXXnnnn	EASIGN	NCOPY	Check-in by equip.	ERM	EASIGN	MCOPY	Not used	May change	
XXXnnnn	EASIGN	NCOPY	Check-in by RMS	RRM	RASIGN	MCOPY	Not used	May change	
RRM	RASIGN	MCOPY	Check-out by RMS	XXXnnnn	RASIGN	RCOPY	Not used	No change	
XXXnnnn	RASIGN	RCOPY	Edit by RMS	XXXnnnn	RASIGN	RCOPY	Add Rnnn	No change	
XXXnnnn	RASIGN	RCOPY	Edit by equip.	XXXnnnn	RASIGN	RCOPY	Add Ennn	No change	
XXXnnnn	RASIGN	RCOPY	Check-in by RMS	RRM	RASIGN	MCOPY	Not used	May change	
XXXnnnn	RASIGN	RCOPY	Check-in by equip.	ERM	EASIGN	MCOPY	Not used	May change	
ERM	EASIGN	MCOPY	Check-out by equip.	XXXnnnn	EASIGN	RCOPY	Not used	No change	
XXXnnnn	EASIGN	RCOPY	Edit by equip.	XXXnnnn	EASIGN	RCOPY	Add Ennn	No change	
XXXnnnn	EASIGN	RCOPY	Edit by RMS	XXXnnnn	EASIGN	RCOPY	Add Rnnn	No change	
XXXnnnn	EASIGN	RCOPY	Check-in by equip.	ERM	EASIGN	MCOPY	Not used	May change	
XXXnnnn	EASIGN	RCOPY	Check-in by RMS	RRM	RASIGN	MCOPY	Not used	May change	

#1 'Equip.' is the shortened form of 'Equipment'.

#2 'XXXnnnn' means an actual OperationID except for COP.

#3 'Rnnn' and 'Ennn' means actual revision string by RMS and equipment accordingly.

Table 19 Usage of the Management Part of RecipeXID (For COP)

Original RecipeXID			Operation by ^{#1}	New RecipeXID					Note
OperationID	AssignorID	CopyID		OperationID	AssignorID	CopyID	RevisionID	Base Part	
From any combination or scratch			Provided by RMS	COP	EASIGN	CCOPY	Not used	Newly set	
From any combination or scratch			Creation by equip.	COP	EASIGN	CCOPY	Not used	Newly set	
COP	EASIGN	CCOPY	Edit by equip.	COP	EASIGN	CCOPY	Not used	Modified	

#1 'Equip.' is the shortened form of 'Equipment'.

8.2 *Additional Recipe Format Definition* — This Standard introduces the following recipes.

8.2.1 *ProcessLink Recipe Format* — The recipe body of Process Link Recipe has the following format.

Table 20 ProcessLink Recipe Format

Item	Form	Description	Note
ProcessMain Recipe	RecipeXID	RecipeXID of ProcessMain Recipe	
ProcessUnit Recipes	List of RecipeXID	List of RecipeXID of ProcessUnit Recipes	

8.2.1.1 *Getting ProcessLink Recipe* — The ProcessLink Recipe is set into RecipeBody of RecipeResultList by using ReadRecipe Transaction with setting ‘ProcessLink’ in TypeID of RecipeXIDList in ReadRecipe Message Parameter.

8.2.2 *Activity Recipe Format* — The recipe body of Activity Recipe has the following format.

Table 21 Activity Recipe Format

<i>Item</i>	<i>Form</i>	<i>Description</i>	<i>Note</i>
ActivityDescription	Maximum 10k byte characters	Human readable text expression of the activity requirements.	
Reference Copy IDs	List of RecipeXID	List of RecipeXID of Reference Copies to be used for the activity.	
New Copy IDs	List of RecipeXID	List of RecipeXID of New Copies to be used for the result of the activity.	

8.3 *Recipe Classification by Usages* — This Standard uses the following usage classification of recipes when recipes need to be classified by usage.

Table 22 Recipe Classification by Usages

<i>Classification by Usages</i>	<i>Applicable RecipeXID</i>			<i>Definition</i>	<i>Note</i>
	<i>OperationID</i>	<i>AssignorID</i>	<i>CopyID</i>		
Production Recipe				A recipe which is used for host triggered PJ creation. Production Recipes are not only recipes for mass production lots but also recipes for engineering lots, etc., if the recipes are used by host triggered PJ creation. Any recipe that is prepared to be used for host triggered PJ creation shall be considered as a Production Recipe.	
	POP	RASIGN	RCOPY	A recipe checked-out from the Repository, normally for mass production.	
			NCOPY	A recipe not checked-in the Repository, normally for an engineering lot.	
	COP	EASIGN	CCOPY	A recipe in CRS. COP is used for PJ creation when POP is not active.	

8.4 *Message Requirements* — This Standard uses the following messages.

8.4.1 *Dedicated Messages* — The Standard defines dedicated messages which use RecipeXID to specify recipe.

8.4.2 *Conventional Message* — For upward compatibility, the messages which use RecipeID (PPID) as an only identifier of recipe shall be interpreted as defined below.

8.4.2.1 *PJ Creation from the Host* — SEMI E40 is used for PJ creation. See the section ‘*PRC Operation (POP) Requirements*’ for detail.

8.4.2.2 *Recipe Transaction Messages* — Recipe transaction messages may be used to access CRS directly under user’s responsibility.

9 Secured Recipe Management Requirements

9.1 *Secured Recipe Management* — A recipe is managed under the following rules.

9.2 *Preconditions* — This Standard assumes the following conditions.

9.2.1 *Master Copy Management Scheme* — This Standard does not define Master Copy management scheme itself but assumes that Master Copies of RMS Recipes and Equipment Recipes are independently managed somewhere in RMS and equipment accordingly.

9.2.2 *Master Copy for RMS Recipe and Equipment Recipe* — RMS manages Master Copy of RMS Recipe, and equipment manages Master Copy of Equipment Recipe.

9.2.3 *Repository for Master Copy* — Master Copy is managed in one logical space ‘Repository’.

9.3 *Copy Management Rule Throughout RMS and Equipment* — This Standard expects that both RMS and equipment manage own Master Copies of recipes under their control by Business Model by using Repository in each side. This Standard uses Reference Copies of the Master Copies provided by own Business Rules from both RMS and equipment under OpeType rule. Also, Operation Copies made by SRO may be referred and registered as Master Copies by the own Business Rules in both RMS and equipment under OpeType rule. Note that COP uses the recipes with conventional identification scheme (CCOPY, instead of Operation Copy) in CRS (instead of Repository).

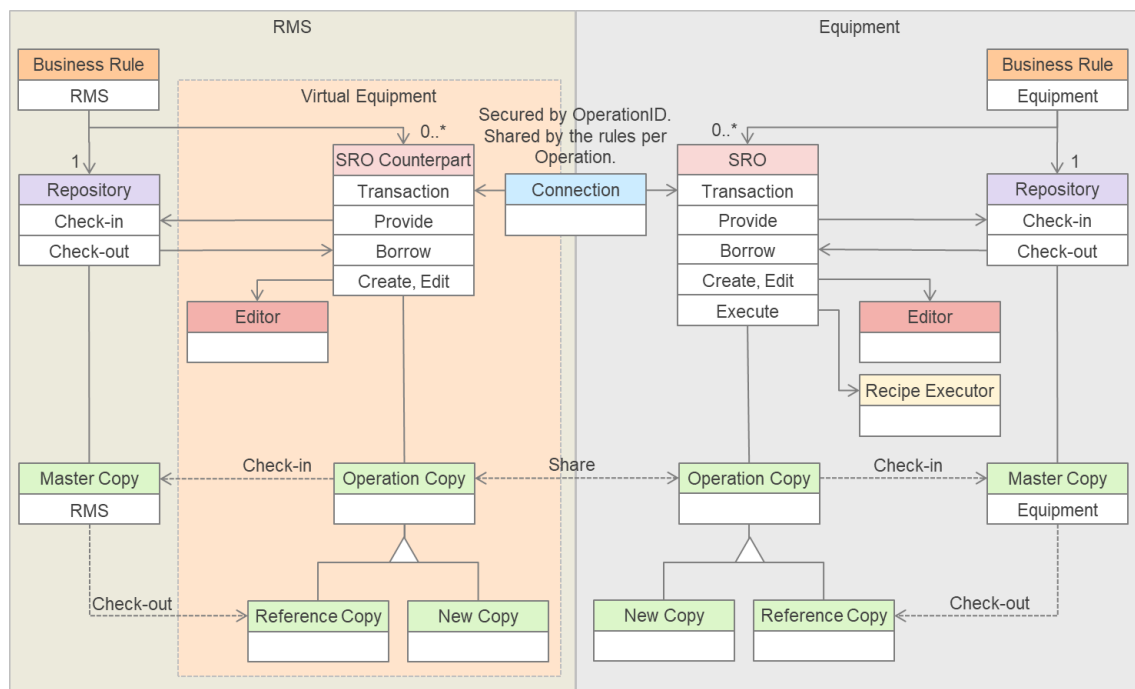


Figure 8
Image of Overall Copy Management (Except for COP)

9.3.1 *Master Copy Management* — Master Copy of a recipe is managed as below.

9.3.1.1 *No Direct Access to Master Copy* — For security, consistency, and uniqueness managements, Master Copies in Repository shall not be accessed from any recipe operations directly but via the Check-in/Check-out function of Repository controlled by the Business Rule.

9.3.1.2 *Check-in and Check-out* — Check-in and Check-out of Master Copy shall be done based on the Business Rules which RMS and the equipment independently have, and those copies shall be exchanged with SRO in the way compatible with the definitions in this Standard.

9.3.1.3 *Operation Copy for Operations* — Formally accessed Master Copy for operation is checked out as an Operation Copy.

9.3.1.4 *Uniqueness Management* — Uniqueness management of Master Copy shall be done upon Check-in based on the Business Rule.

9.3.1.5 *Security Management* — Security management of Master Copy shall be done upon Check-out based on the Business Rule.

9.3.1.6 *Independent from RMS-Equipment Communication* — Check-in and Check-out are internal actions of RMS or equipment and do not occur directly upon any recipe transaction between RMS and equipment.

9.3.2 *Master Copy Operations in Equipment* — The Master Copy of Equipment Recipe shall be operated per the following rules based on equipment Business Rule.

9.3.2.1 *Check-in as a Master Copy from SRO* — An Operation Copy under control of SRO may be checked in as a Master Copy into the Repository. Normally, the Operation Copy is appropriately renamed upon Check-in.

9.3.2.2 *Check-out of Master Copy for SRO* — A Master Copy may be checked out for the use by SRO as a Reference Copy marked by OperationID the same with the one the SRO has.

9.3.3 *Master Copy Operations in RMS* — The Master Copy of RMS Recipe is operated per the following rules based on RMS Business Rule.

9.3.3.1 *Check-in as a Master Copy via SRO Counterpart* — An Operation Copy under control of SRO may be checked in as a Master Copy into the Repository. Normally, the Operation Copy is appropriately renamed upon Check-in.

9.3.3.2 *Check-out of Master Copy via SRO Counterpart* — A Master Copy may be checked out to SRO Counterpart for the use by SRO as a Reference Copy marked by OperationID the same with the one the SRO has.

9.3.4 *Operation Copy Management* — An Operation Copy is managed by each SRO which has the same OperationID with the Operation Copy has.

9.3.4.1 *Operation Copy for SRO* — Normally, Operation Copies to be used for the SRO are prepared by the Business Rule in both RMS and equipment, upon activation of the SRO or later. Operation Copy consists of the following elements depending on OpeType.

- Reference Copy made from Master Copy of RMS Recipe and provided to the SRO Counterpart through RMS Business Rule
- Reference Copy made from Master Copy of Equipment Recipe and provided to the SRO through equipment Business Rule
- New Copy which is made in the SRO Counterpart by RMS or provided to the SRO Counterpart through RMS Business Rule. New Copy may include appropriately renamed Operation Copy from other SROs (even in other VE)
- New Copy which is made in the SRO by the equipment or provided to the SRO through the equipment Business Rule

9.3.4.2 *Uniqueness Management Within OperationID* — The RecipeXID uniqueness of Operation Copy upon creation or modification shall be maintained within the OperationID by the SRO which has the same OperationID. The RecipeXID given by the SRO is temporary, and will normally be renamed upon check-in.

9.3.4.3 *Recipe Transaction Between RMS and Equipment* — Recipe transactions between RMS and equipment simply move recipes between SRO in equipment and SRO Counterpart in RMS.

- *No Direct Execution of Check-in or Check-out* — Recipe transactions between RMS and equipment do not directly execute Check-in or Check-out.
- *OpeType Dependent* — Recipe transactions between RMS and equipment shall be done based on the rules each OpeType has.

9.3.5 *Use of Operation Copy* — An Operation Copy may be operated as below depending on OpeType.

9.3.5.1 *Creation* — A New Copy may be created from scratch.

9.3.5.2 *Modification* — An Operation Copy may be modified. Also, a New Copy may be created from Reference Copy.