SEMI E139-0708
レシピとパラメータに関する管理の規定（RaP）

Notice: This translation is a REFERENCE COPY ONLY. If differences should exist between the English version and a translation in any other language, the English version is the official and authoritative version.

免責事項:この SEMI スタンダードは，投票により作成された英語版が正式なものであり，日本語版は日本の利用者各位の便宜のために作成したものです。万が一英語と日本語とに差異がある場合には英語版記載内容が優先されます。

SEMI スタンダード日本語翻訳版をご利用にあたっての注釈を本文の末尾に記載しております（「すべきである」「しなければならない」について等）。

本スタンダードは，global Information & Control Committee で技術的に承認されている。現版は 2008 年 5 月 13 日，global Audits and Reviews Subcommittee にて発行が承認された。2008 年 6 月に www.semi.org で。初版は 2005 年 3 月発行，前版は 2008 年 3 月に発行された。

SUPERSEDED

目次

1 目的
2 範囲
3 制限
4 参照スタンダードおよび文書
4.1 SEMI スタンダード
4.2 ISO スタンダード
4.3 OMG スタンダード
5 用語
5.1 SEMI 編集用語
5.2 略語および頭字語
5.3 定義
6 表記法
6.1 オブジェクトモデリング
6.1.1 統一モデリング言語（UML）
6.1.2 クラス図
6.1.3 クラス名
6.1.4 抽象クラスと具象クラス
6.1.5 クラスの属性定義
6.1.6 関連の記載
6.1.7 関連の可航性
7 RaP の概要
7.3 RaP の関係物
7.6 RaP の環境
7.7 レシピの実行
7.8 レシピの作成と編集
7.9 レシピの追跡と保存
7.10 通信の概要
7.11 主要なレシピの概念
7.12 ProcessDefinitionElement
7.12.2 レシピ構造
7.12.3 パラメータ化
7.12.4 TransferContainer
7.13 目的の実現

注:SEMI による書面での了承なき場合は，本書の全部もしくは一部の転載を禁じる。
©Semiconductor Equipment and Materials International（http://www.semi.org/）
©SEMI ジャパン 〒102－0074 東京都千代田区九段南 4－7－15

AVA Documents
Document Sharing Hub

Full document details: SEMI E139-0708 (Japanese)

8 要求事項
8.1 要求事項概要
8.2 レシピオブジェクトモデル
8.3 PDE クラス図
8.3.2 PDEheader クラス
8.3.3 AntecedentData クラス
8.3.4 ReferencedPDE クラス
8.3.5 ExecutionTarget クラス
8.3.6 PDEparameter クラス
8.3.7 PDEbody クラス
8.3.8 PDEbodyReference クラス
8.4 RaP サービス
8.5 RaPnode の要求事項と解説
9 準拠
10 関連文書

図リスト

図 1 — RaP の解決すべき課題
図 2 — RaPnodes
図 3 — PDE の内容
図 4 — PDE と分離された PDEbody
図 5 — レシピ構造の図解
図 6 — パラメータの流れの図解
図 7 — PDE クラス図
図 8 — RaPnode クラス図
図 9 — TransferContainer の図解

表リスト

表 1 — 属性表の形式
表 2 — Class xxx のたどり着ける関連
表 3 — *PDE* クラスの属性
表 4 — *PDE* のたどり着ける関連
表 5 — *PDEheader* クラスの属性
表 6 — *PDEheader* のたどり着ける関連
表 7 — *AntecedentData* クラスの属性
表 8 — *AntecedentData* のたどり着ける関連
表 9 — *ReferencedPDE* クラスの属性
表 10 — *ExecutionTarget* クラスの属性
表 11 — *PDEparameter* クラスの属性
表 12 — *PDEbody* クラスの属性
表 13 — *PDEbodyReference* クラスの属性
表 14 — *RaPnode* クラスの属性
表 15 — サービスの説明
表 16 — サービスパラメータ一覧
表 17 — サービスパラメータ
表 18 — 要求／指示 と 応答／確認 の記号

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

表 19 — *getPDEdirectory*()サービスパラメータ
表 20 — *deletePDE*()サービスパラメータ
表 21 — *getPDEheader*() サービスパラメータ
表 22 — *getPDE*()サービスパラメータ
表 23 — *requestToSendPDE*() サービスパラメータ
表 24 — *sendPDE*()サービスパラメータ
表 25 — *resolvePDE*()サービスパラメータ
表 26 — *verifyPDE*()サービスパラメータ
表 27 — Manifest エントリデータの定義
表 28 — RaP 準拠表

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

1 目的

1.1 本規定の目的は工場内情報管理系（Factory Information & Control System: FICS）と装置間の協調動作を規定するものであり，この協調動作は装置に於けるプロセス実行に関する仕様（例えば装置のレシピ）を管理するために行われるものである。

1.2 本目的の達成のため備える必要のある，主たる要素が 6 つある:

- *装置上，ならびに装置外部でのレシピの管理* — 装置内，ならびに FICS レベルでの分割レシピの管理機能の装備。統合用の同管理アプリケーションの要求が無くとも備えなければならない。
- *レシピの完全管理* — ある装置においてプロセスの実行方法を定めるレシピは FICS 内で使われることを意図したレシピと同等であることの保障。
- *プロセスの完全管理* — 装置の使用に際して変えることが可能で，プロセスの結果に影響を与えるような，装置の構成にかかわる量値／設定がプロセスジョブの使い方のなかでホストが管理できること。
- *調整用パラメータの定義* — 他の SEMI スタンダード（例，SEMI E40）で利用され，すべての装置の間で一貫した方法で，パラメータの調整ができるようにしたレシピのパラメータの定義。
- *装置上，ならびに装置外部でのレシピの作成と編集* — FICS とレシピを作成／編集する機能を備えるシステム間での，一貫したやり取りのプロトコルの定義。
- *情報のアクセス能力* — レシピに関しする重要な情報が FICS から覗けるようにすること。

1.2.1 上記目的の達成方法に関する議論は § 7 を参照のこと。総括については¶7.13 に掲載する。

2 範囲

2.1 本規定は半導体工場で使われる装置のレシピ，およびその装置の行うプロセスを定義する要素の自動管理を行うために，概念，振る舞い，ならびにサービスを規定するものである。本規定は半導体製造装置，レシピ，工場内情報管理系（Factory Information and Control System: FICS）の間相互のやり取りの機構についても示すものである。

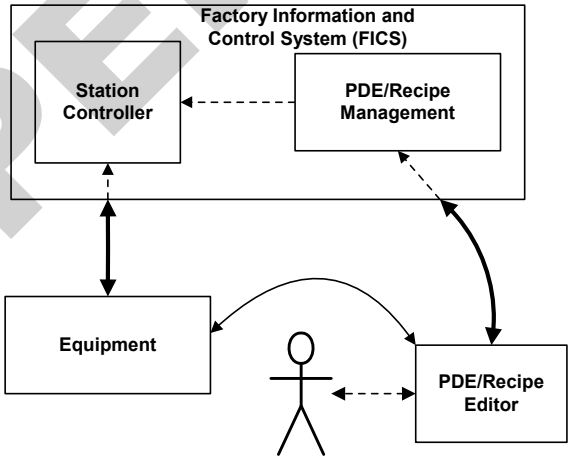

図 1
RaP の解決すべき課題

2.2 図 1 は本規定で定める通信のやり取りを図示したものである。太線の矢印は完全に規定を行うインタフェースであることを示す。破線の矢印はレシピ管理の概念の理解を助ける目的で示したもので，ここでは何の規定をするものではない。細線の矢印は一部だけ規定を行うインタフェース（装置とレシピ編集機器の間）であるが，サプライヤ独自の可能性もある。このインタフェースに関してはメッセージ通信の詳細は規定しないものの，そのインタ

SUPERSEDED
Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

フェースの備えるべき機能に付いては規定するものである。この件は本書の後の部分でもう少し説明するものとする。

2.3 要約すると以下が本書で規定すインタフェースである:

- FICS ↔ Equipment – メッセージ通信サービスすべて
- FICS ↔ *PDE*/Recipe Editor – メッセージ通信サービスすべて
- Equipment ↔ *PDE*/Recipe Editor – 機能要件のみ

注意: このスタンダードはその使用に関連したすべての安全問題を取り扱うことを意図していない。このスタンダードの使用者は，その責任において，適切な安全および健康上の作業方法を確立し，また使用前に規制上その他の制限への適用性を判断するものである。

3 制限

3.1 以下の事項は本書の範囲外であり，本書で規定することはない:

- *PDEbody* の内容や書式。
- メッセージのフォーマットやメッセージサービスに関する通信の詳細事項。
- 装置内で *PDE* がどの様に使われ，どの様に事項されるか。
- FICS がどうやって装置上の *PDE* 実行を起動させるのか。
- どの様にして，何時，パラメータの値が装置に伝えられるか。
- どの *PDE* でどういったパラメータの値がどの状況下で使われたかを記録する際にどの様にしてデータを集めるか。
- ユーザが人間である場合の *PDE*/Recipe 編集機器の係わり方。
- FICS のアプリケーションの設計（例，図 1 の様なステーションコントローラや *PDE*/Recipe 管理機能）。本書の内容の理解を深めるためにこの様なアプリケーションを示したものである。

4 参照スタンダードおよび文書

4.1 *SEMI スタンダード*

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

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

SEMI E40 — Standard for Processing Management

SEMI E120 — Specification for the Common Equipment Module (CEM)

4.2 *ISO スタンダード*[1]

uuid: ISO/IEC 11578:1996 Information technology - Open Systems Interconnection – Remote Procedure Call (RPC)

4.3 *OMG スタンダード*[2]

Unified Modeling Language (UML) Specification, Version 1.4, OMG Specification 01-09-67

注意: 別途指示のない場合，引用されたすべての文書は最新の出版バージョンである。

1 International Organization for Standardization, ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30, http://www.iso.ch

2 Object Management Group, Inc., 250 First Ave. Suite 100, Needham Mass. 02494, Telephone: 781.444.0404; Fax: 781.444.0320; http://www.omg.org/

SUPERSEDED
AVA Documents Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

5 用語

5.1 *SEMI 編集用語*

5.1.1 本規定で使用される多くの用語の定義や説明は SEMI 編集用述語集で見つけることが可能であり，SEMI のウェブサイト http://www.semi.org/ から利用可能である；OMG UML 仕様も同様。オブジェクトモデルに関する用語については OMG UML 仕様による定義を尊重するものとする。

5.2 *略語および頭字語*

5.2.1 *FICS (Factory Information and Control System)* — 工場内情報管理系

5.2.2 *PDE (Process Definition Element)* — プロセス定義要素

5.2.3 *RaP (Recipe and Parameter Management)* — レシピとパラメータの管理 ー 本仕様。

5.2.4 *UML (Unified Modeling Language)* — 統一モデリング言語

5.3 *定義*

5.3.1 *チェックサム (Checksum)* — あるデータの並び（ファイルや文字列など）から計算し，そのデータを一意に識別できる，単純でユニークな値。“digital fingerprint” あるいは “message digest”などと呼ばれることがある。異なる 2 組のデータの並びが同じチェックサムを持つことは起こりにくい。それ故，チェックサムはデータの忠実性の確認に用いることができる。典型的な手順としては，1) データの並びのチェックサムを計算する；2) データの並びとチェックサム値とを他に送付する；3) 受け取り手はチェックサム値を新たに計算し，送り手からのものと比較する。チェックサムが同一だった場合，データに手が加わったり壊れていたりすることが無いことが約束される。

5.3.2 *工場内情報管理系 (Factory Information and Control System: FICS)* — 工場の運用や設置された装置を管制するソフトウエアシステム。一般に MES，ステーションコントローラ，レシピ管理機器などと呼ばれる部分を含めても良い。

5.3.3 *モジュールパラメータ (Module Parameter)* — 装置モジュール（典型的なものはプロセスモジュール）上でのプロセスを左右する設定。プロセスモジュールの場合，モジュールパラメータは温度，圧力，流量，などを表すことになる。装置のレシピはそのモジュールでの目的を遂行するために，モジュールパラメータを操作するのが一般である。

5.3.4 *PDE (Process Definition Element: プロセス定義要素)* — 装置上での動作やプロセスの実行を行う仕様。装置上のある動作に対するレシピには複数の *PDE* が含まれていることもある。*PDE* は FICS が個々に係わり，あるいは関知する最小のプロセス定義単位である。何れの *PDE* にも *PDEheader* と *PDEbody* がひとつずつある。

5.3.5 *PDEbody* — プロセス定義要素の実行関与部分。一般的には *PDEbody* は *PDE* 構造の中に収められる。場合により *PDEbody* が分離されたものとして存在しても構わないが，常にその *PDE* 自体が占有する関係でなければならない。

5.3.6 *PDEeditor* — *PDE* を生成，消去，変更する能力を提供するソフトウエアシステム。*PDEeditor* 装置のシステムに一部として存在する場合もあるが，その装置の製造者や他の業者により用意される独立のシステムの場合もある。

5.3.7 *PDEheader* — プロセス定義要素の前書き部分。*PDEheader* は対応する *PDEbody* に関する情報を寄せ集めたものである。この情報は *PDEbody* をアクセスせずとも *PDE* を管理，活用するに十分足るものである。

5.3.8 *uuid* — 普遍的にユニークな識別子。この 128 ビットの空間（多くは 36 個の文字列で表現される）は標準規定 ISO/IEC 11578:1996 – Remote Procedure Call（RPC）に従って計算されるこの空間は，何時いかなる場でもユニークであることが保証される。このユニークの保証は例え uuid が別のシステムにより独立に生成されたとしても保たれる。

5.3.9 *変数パラメータ (Variable Parameter)* — レシピ本体内で定義されていて，正式に定義された変数（設定）。外部から実際の値を与えることができる。

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

6 表記法

6.1 本書では下記の表記法が用いられる:

- RaP 特有な用語を際立たせるため，本規定内の用語を*斜字体*で表現するものとする。斜字体とする用語にはクラス名，属性，サービスも含む。サービス名の終わりには()を付け，例えば "*getPDE*()"などとする。
- "XxxList"と命名される多数個からなるメッセージパラメータを避けるため，本書では "Xxx からなるリスト" の様にしてリストを示す記法を採用するものとする。この場合（リストではなく）パラメータ自身の定義とデータ型を指定するものとする。用語"リスト"は同じデータ型の項目からなる 0 個以上の集まり（または集合体）を意味する。

6.2 *オブジェクトモデリング*

6.2.1 *統合モデリング言語（UML）*

6.2.1.1 本規定は UML の記法をクラス図と例示用のオブジェクト図のすべてに用いることとする。その他のオブジェクトモデル化用の図法は本規定では用いないものとする。

6.2.2 *クラス図*

6.2.2.1 UML のクラス図は意味を明快に定義するもので，本規定の正規の一部分である。この図の内容の詳細を文字で繰り返す必要は必ずしもない。

6.2.3 *クラス名*

6.2.3.1 クラス名は語頭を大文字とする。クラスの属性とサービス名は小文字で始まるものとする。

6.2.4 *抽象クラスと具象クラス*

6.2.4.1 各クラスを抽象か具象かに区別する。抽象クラスは直接，実装されることはない（即ち，インスタンスはできない）。具象とされたクラスは何れも直接，実装して構わない。抽象クラス名は UML のクラス図中，*斜字体*で示すものとする。

6.2.5 *クラスの属性定義*

6.2.5.1 クラスの属性定義は表形式で下記の表 1 の様に定義する。UML のクラス図で "+" 記号は公開されている（"public"）属性であることを示す。本規定に登場する属性はすべて公開であり，それ故，"+" 記号は省略する。

表 1 属性表の書式

属性名	*定義*	*アクセス*	*必要性*	*形態*
		RO か RW	Y か N	下記参照のこと

6.2.5.1.1 *アクセス* — インスタンスからサービス提供側[3]の方向に対して，属性は設定可能（読み書き，即ち，RW）か設定不可（読み出し専用，即ち，RO）

6.2.5.1.2 *必要性* — この属性が必須か否か: Y – Yes, or N – No.

6.2.5.1.3 *形態* — 属性のデータ型を定める。本規定でのデータ型は高レベルの定義であり，個々の技術の対応するデータ型へ適宜マッピングすることになる。本規定では以下のデータ型がメッセージ通信サービスのパラメータにも用いられる。

- *バイナリ* — 何がしかの値を持つデータバイトの並び。構造が明らかでないため，バイナリ値は "unformatted" とい呼ばれることがある。
- *ブール数* — 真（"true"）か 偽（"false"）の値をとる。
- *列挙型* — リストに限定された可能な値が許される形式。本書ではこの値は文字列で表現されるが，これと異なる実装（例，この文字列に対応する整数）でも構わない。

3 属性表は複数のスタンダードで使用されている共通書式であり，RaP で定義されているすべての属性は ReadOnly (RO)である。

SUPERSEDED
AVA Documents Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

- *Error* — この型は発生したエラーに関する情報を内容とする構造体である。構造体の形態は実装に依存し，個々の技術を使ったメッセージ通信（例，SECS-II や XML/SOAP）を規定する，RaP の各々の枝番規定で定義されなければならない。
- *整数* — 数値。常に整数である値。形態とバイト数は実装時の定義に委ねる。
- *実数* — 整数と分数の何れも取り得る数値。形態とバイト数は実装時の定義に委ねる。
- *文字列* — 文字が連なったもの。制約と長さは実装用技術に委ねるが，そうでない場合は本規定で定めるものとする。文字列は UTF-8 で実装するよう，推奨するものである。
- *Checksum* — ある続いたデータに対して，指定の方法で計算したチェックサムの値。
- *Time* — 注目事象の発生の日時の表現。この型の項目の構造と形態は実装時の定義に委ねる。
- *UUID* — ¶4.2 で参照する ISO 標準の策定による永続的にユニークな識別子（universally unique identifier），ASCII 文字で表現される。この型はすべての章で “UUID” と表現されるが，これは ISO 標準の表現 “uuid” と対比させたもので，属性の内容物の型が “UUID” であることを意味する。
- *Any* — 形式は上記のもの，あるいはそれ以外の任意のもの。形態は実装時の定義に委ねる。
- *xxx のリスト* — 指定された要素項目が1個または複数個並んだものの型（ここの xxx は要素となるデータ型）。

6.2.6 *関連の記載*

6.2.6.1 関連は表 2 の形式を使って記述される。この表にはたどって行ける注目クラスのみがこの表に記載される。たどり付ける関連についての説明は¶ 6.2.7 を参照のこと。

6.2.6.2 各関連について，表では型，説明，関連クラス，役割，個数が行単位で示す。

- *型* — 関連の型。構成物，集約，一般関連が使われる。この型の説明は UML 標準の¶4.3 参照のこと。
- *説明* — 文字による関連の説明。
- *関連クラス* — この関連によりつながっている先のクラス名。
- *役割* — 関連において関連するクラスのの役割名は UML 図から抽出する。集約されたオブジェクトである構成要素を普通は意図するため，構成物の関連は本書では役割名としては記入しない。
- *個数* — 関連クラスがこのクラスに幾つ係わるか – UML の書き方で記入する。

表 2 クラス xxx のたどり着ける関連

型	*説明*	*関連クラス*	*役割*	*個数*

6.2.7 *関連の可航性*

6.2.7.1 UML の関連には「可航性」の概念が含まれる。関連がたどり着けるものであるならば，その関連は関連の相手先のクラスインスタンスにたどって行き着ける（ターゲットオブジェクト）。

6.2.7.2 原則として関連は両方向からたどって行ける。矢印が関連の片方の端のみであれば，可航性はその方向のみである。

6.2.7.3 いかなるオブジェクトも可航性のある関連に関して，ターゲットオブジェクトへの参照を得ることが可能なはずである。本書で定義されたいかなる可航性のある関連に関しても，このことは実装者への要求である。ターゲットオブジェクトの参照方法は実装に依存し，本書では規定しない。

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

7 RaP の概要

7.1 RaP の規定は装置レシピの定義，管理，伝達に焦点を絞るものである。RaP ではレシピの構成要素は ProcessDefinitionElements（プロセス定義要素）即ち *PDE* と呼ばれる。本章では §1 で示した本規定の目的に沿うように，用いられる大まかな観点で概念と要件を説明するものである。

7.2 この RaP の概念では本書で後程定義する要求事項に対する参照を可能にするものである。しかしながら，この概要の章で要求事項の定義をすることはない。この概要は後述の詳細な要求事項がより楽に理解できる様，一般的な RaP の概念の理解を意図したものである。

7.3 *RaP に携わる関係者*

7.4 RaP ではレシピの管理において 3 種の関係者が存在するものと想定する – 即ち FICS，装置，*PDEeditor*。この 3 つが RaP サービスを提供するときに，これらは "*RaPnodes*" となる – FICS は *FICSnode* となり，Equipment は *EquipmentNode* となり，*PDEeditor* は *EditorNode* となる。

7.5 レシピ管理の目標を達成するためにこの 3 者は力を合わせて動作しなければならない。この 3 者は（*RaPnode* に）規定されたサービスを利用する独立した実体として存在するかも知れないし，ある場合には（多くの場合エディタが装置上に載せられる様に）一体であることもある。一体になった場合はその塊は標準で定める通信は必要ないため，RaP では図 2 に示す様に，この関係者が独立に存在する場合に必要なサービスとデータを規定するものである。

7.6 *RaP の環境*

7.6.1 レシピ管理の議論する際，論理的に 3 つの別々の行為に分けて考えられる:

- レシピの実行
- レシピの作成と編集
- レシピの追跡と保管

7.6.2 RaP では上記 3 種の何れの行為をも，その実施方法を明確に規定するものではない。しかしながら，上記の行為は RaP の定義するサービスや概念を適用する際の前後の関係を定めるものである。レシピ自体がこの行為の観点の一部を反映する必要がある。

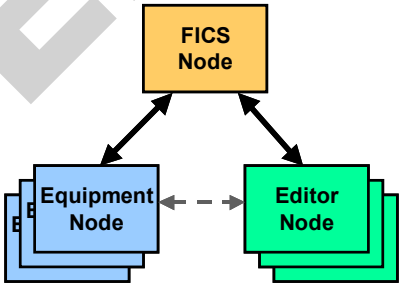

図 2
RaPnodes

7.7 *レシピの実行*

7.7.1 実行行為を装置上で行う場合，動作の起こる順序にその動作が指示されなければならない。RaP は主要な仕様（レシピ），ならびにレシピを調整用に設定し得るパラメータを説明するものである。レシピの実行過程を説明することはしない。

7.7.2 RaP では装置または装置の何らかの構成要素がレシピの実行先であるものと仮定する。本書では実行先の部位をプロセスモジュール（Process Module）と呼ぶこととする。RaP はプロセスモジュールを特定する手段があるものと仮定し，プロセスモジュールが備える個性は工場側にとって既知であるものとする（製造，型式，供給者など）。

SUPERSEDED
AVA Documents Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

7.7.3 RaP は SEMI E120 と共用性がある。SEMI E120 での言い方を用いると，プロセスモジュールは “ExecutionElement”クラスに対応する。ExecutionElement クラスは装置の構造で，材料を処理する部分をモデル化したものである。その属性には，その ExecutionElement が実行できるレシピの種類を指定する “recipeType” がある。RaP が管理をするレシピがそこにある。RaP は SEMI E120 の使用，あるいは準拠を要求するものではない。しかしながらもし SEMI E120 準拠であるならば，RaP に関する多くのデータ（入出力何れも）の源である EquipmentElement がより明確にわかることとなる。

7.7.4 FICS の観点からは，レシピに関して個々の状況で正しいレシピが選択（機械と基板，プロセスの段階などの組み合わせ）できる様，十分なことが分かっていなければならない。

7.8 *レシピの作成と編集*

7.8.1 RaP ではレシピの作成と編集は別個の概念と考える。このことは編集能力は装置上に，工場管理系の内部に，その何れでもない独立系に，あるいはそのすべての何れかの組み合わせとしても良いことを意味する。この何れにも利点と欠点があり，それも時が経つにつれて変化する。それ故，RaP は何処に *PDEeditor* を置かなければならないなどと要求したりはしない。

7.9 *レシピの追跡と保管*

7.9.1 少なくとも装置上にある程度置かれるレシピの保管は最低限，プロセス遂行上必要である。*PDEeditor* でのレシピの保管も同様に必要である。例えば，複合分散レシピの作成が可能となる様にするためである。大抵の工場では，ほとんどのレシピの複製を安全に保管したいと考えている。レシピ管理の関係者は何れも，少なくとも一部のレシピを保管したいと概ね考えている。

7.9.2 追跡や妥当性（ID，チェックサム，作成日時など）確認に使われる，レシピの特質をも RaP は利用できる様にすることとする。

7.10 *通信の概要*

7.10.1 通信に関しては上記定義済みの 3 種の “*RaPnodes*” 間に RaP では注目することとする: そのノードは *FICSnode*，*EquipmentNode*，*EditorNode* である。Ra インタフェースを使って実行できる機能には 3 種類ある:

- *PDE* の移動 — ある *RaP ノード*に *PDE* を要求，あるいはある *RaP ノード*へ *PDE* を送る。
- *PDE* の管理 — ある *RaP ノード*にある *PDE* の情報の取得や，必要な場合の *PDE* の消去。
- 照合 — どの *PDE* があるジョブに対して必要となるかを判断すること，ならびに必要な部分全部が存在するか，正しい書式であるかについてチェックをする。

7.10.2 上記 3 種類が，RaP が定義する 8 個のサービスにつながる。通信サービスの詳細は¶8.4 に示すこととする。その章ではどの *RaPnodes* が上記 3 種のいずれを用意すべきかも示す。¶8.5 では *RaPnodes* 追加要求事項を示す。

7.11 *レシピの主要概念*

7.11.1 与えられた目的を遂行するために，RaP は単純なレシピの移動サービス以上のものを提供する必要がある。堅牢な「レシピとパラメータの管理」の能力を作り上げるために，上記の基本機能を改変するに当たって，本章は本規定で使用する主要な概念を説明するものである。

7.12 *ProcessDefinitionElement（プロセス定義要素）*

7.12.1 レシピは全体が一つの構成要素でできていても，複数個の構成要素から成っていても良い。レシピの構成要素は ProcessDefinitionElement（プロセス定義要素），即ち *PDE* と呼ばれる。*PDE* は RaP の中心となる構造である。本章は *PDE* を説明するものである。

7.12.1.1 *PDE – 二部構成*

7.12.1.1.1 レシピ取り扱いの自動化の障害の一つに，内容の不透明性がある。いくつかの理由によりサプライヤはユーザに対して（ひいては自動化システムに対しても）レシピの内容を見せようとはしたがらない。レシピの構成要

SUPERSEDED
Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

素（即ち PDE と紹介したものであるが）を 2 分することで RaP はその問題を収めようとするものである: 資料部と実行部。これを図 3 に図示する。

7.12.1.1.2 *PDE* の資料部には *PDEheader* が含まれる。この部分はいかなるときも公開され，ユーザは利用可能である。*PDEheader* には名称，説明，*PDE* の作者，派生元（基にした *PDE*），利用を想定したプロセスモジュールなどのような情報を収容するものである。*PDEheader* の内容に関するこれ以上の詳細は ¶8.3.2 に譲るものとする。

7.12.1.1.3 *PDE* の内のもう片方の部分には設定や実行に関する指示を収容する。*PDEbody* の書式は規定せず，非公開であっても良い（即ち，専用で不透明）。ジョブ実行時に装置は PDEheader ではなく PDEbody の内容に依存することに注意すること。

7.12.1.2 *同一性の保証*

7.12.1.2.1 *PDEheader* 情報に関して重要な点は，この *PDE* は工場側がそうであると考えているものに間違いがないことを保証できることである。これは元来，2 つのデータ値で達成されるものである: *uid* とチェックサム。*uid* は参考の章に示した ISO スタンダード（¶4.2 参照）の定義する uuid 値をもつ，普遍的にユニークな識別子である。*PDE* が修正される度に，PDE の *uid* は違うものにされる必要がある。それ故，*uid* が合っていれば，その *PDE* が求めているものであることが保証される。

7.12.1.2.2 PDE に変更が加わらなかったことを更に保証するものとして *PDE* に対するチェックサム値がある。対象となる項目に対するチェックサムは IETF（§10 参照）の定める MD5 技術を使い計算される。変更が施されなかったことを確認するため予め計算し，*PDE* 内に書きもまれた値と必要な時にこの計算値は比較される。

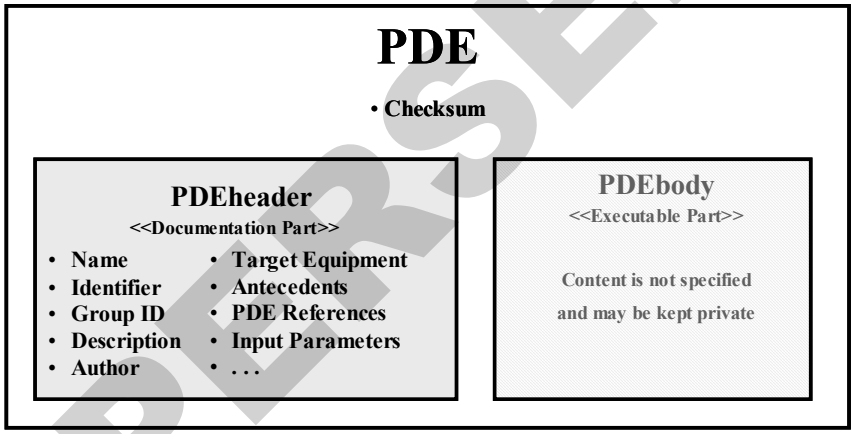

図 3
***PDE* の内容**

7.12.1.3 *PDEbody の分離*

7.12.1.3.1 *PDE* 内部に置かれた *PDEbody* は *PDEheader* と同等の書式[4] をとり，内容は普通，ユーザにとって理解可能なものである。*PDEbody* がユーザにとって不透明であったり，書式がヘッダ情報と共通でなかった場合，*PDE* とは分離して置くことになる。このことを図 4 に図示するが，この場合，分離した *PDEbody* のチェックサムは *PDE* 内部に保存される。

7.12.1.3.2 *PDEbody* 自体だけからは同一性は得られないため，分離 *PDEbodies* の追跡は，ある場合は FICS にとって必要なことである。それ故，*PDE* とそのボディの関係を保持する方法を FICS は採らなければならない。*PDEbody* はその *PDEheader* から分離して移動されることはない。

4 *PDEheader* の書式の定義は実装の問題であり本書の枝番の規定で定める予定である。.

SUPERSEDED
AVA Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

7.12.1.3.3 装置，ならびに *PDEeditor* は *PDEbody* の内容を解釈でき，それにより周到な追跡することなく関係を知ることができる。装置や *PDEeditor* が *PDEheader* と *PDEbody* の間を関係付けられる様，*uid* の様な鍵となる情報を *PDEbody* は備える必要がある（¶8.5.4.5 参照）。

7.12.1.4 *入力パラメータ*

7.12.1.4.1 *PDEbody* はまた，実行時に設定可能な入力パラメータを定義することもある。*PDEheader* はこの *PDEparameters* を記載することとする。パラメータ化については更に¶7.12.3 で議論する。

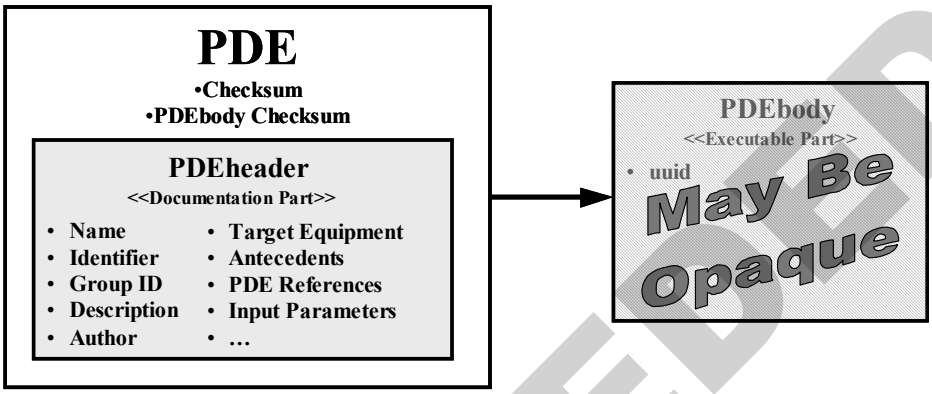

図 4
分離 *PDEbody* を持つ *PDE*

7.12.1.5 *他の PDE の参照*

7.12.1.5.1 実行中に *PDE* は他の *PDE* にアクセスする必要があることがある（サブルーチン呼び出しに例えられる）。*PDEheader* はこの参照について記載するものである。*PDE* を参照するには 2 つの方法がある。最初に紹介するのはユニークな識別子（*uid*）によるものである。*uid* は個々の *PDE* に対する参照先となる。2 つ目の参照方法はグループ識別子（*gid*）によるものである。あるグループ内の *PDE* はお互いに代役を務めることができ，それ故，同一の組み合わせの入力パラメータを共用せざるを得ない。この種の参照は実行前にグループのある特定のメンバに対してたどって行うものである。参照をどの様にたどるかについて更に理解するには¶7.12.2.5 を参照のこと。

7.12.1.6 *RaP への転換*

7.12.1.6.1 *PDE* を定義していく過程は従来の独自のレシピから転換するよい道筋となる。古い形態のレシピは RaP 適用時には *PDEbody* になり得る。"*PDEbody*" は分離物とすることも可能であるため（独自形態の内容である場合），元のレシピをほとんど，あるいは全く変更してはならない場合も数多く存在する。*PDEheader* を含め，*PDE* の構造はラッパーとして作用する。

7.12.1.6.2 RaP の要求条件によって既存のレシピ構造が結果として変更される可能性がある 2 つの領域がある。1 つの領域は，EquipmentNode が PDEheader をその外部の PDEbody と一致させることができる要求条件である。これは，PDEbody に uid 値を挿入することによって実行するのが最良であろう。しかし，PDEheader には PDEbody を変更しなくても使用できる"bodyChecksum"値がある。しかし，これには PDEbody のチェックサムの再計算が必要となる。もう 1 つの領域は，存在する PDEparameter をユーザが指定できる要求条件である。PDEbody には，ユーザ指定の PDEparameter を格納しなければならない。しかし，ユーザ指定の PDEparameter がない既存のレシピの場合，ユーザが PDE を編集してそのようなパラメータを追加するまで変更は不要である。例えば，実装者は既存のレシピのための古いレシピフォーマットをサポートするように選択できるが，新しいすべてのレシピを PDEparameter のサポートを含む新しいフォーマットに転換する必要がある。

7.12.2 *レシピ構造*

7.12.2.1 レシピが複数の構成要素からできている場合，RaP はその構成要素間の関係を記録化する（¶7.12.1.5 参照）。結果として図 5 の様な階層レシピ構造となる。

SUPERSEDED
Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

7.12.2.2 レシピの階層は唯一つの *PDE* を頂点とする構造となる。この *PDE* は "*Master PDE*" と呼ばれる。これはジョブが指定されたときに参照される *PDE* である。この場合，SEMI E40 のプロセス管理を例にとると，RecID（即ち RCPSPEC）には *Master PDE* の *uid* が入る。SEMI E40 や SEMI E30 と併用する際の RaP に関するこれ以上の議論は関連情報 §R2-2 参照のこと。代わりに *gid* を用いることがあり，*uid* は後から決まることになる。

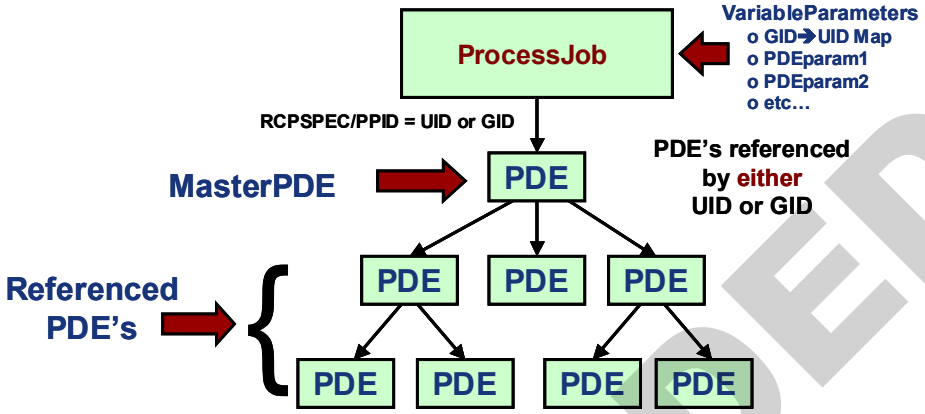

図 5
レシピ構造の図解

7.12.2.3 *Master PDE* が最初に実行される *PDE* であることをレシピ構造は意味する。*Master PDE* の他には，レシピ構造は *PDE* 実行の順序や装置内でのウェーハの運ばれる経路を明記しない。装置サプライヤはウェーハの移動経路がわかる形態（人が読める）の *PDEbody* をもった *PDE* を提供する様，望まれる。

7.12.2.3.1 ここでその他の補足事項を紹介する:

- *Master PDE* は指示名称で *PDE* の種類ではない。*Master PDE* は「実行可能」となっていなければならない（表 5 参照）。
- 同一 *PDE* に対して複数からの参照には，制約は特にはない。
- レシピの階層数に特に制限はない。しかしながら，大抵の場合十分な，2 つか 3 つの階層と考えるのが望ましい。*Master PDE* に全部のレシピが入っている場合，1 階層だけで良いことになる。
- ある 1 つの *PDE* がある階層の複数のレシピに現れることがある。その様なレシピ構成要素の再利用は望ましいものであり，推奨されるものである。

7.12.2.4 *レシピの保守に関する考察*

7.12.2.4.1 グループ ID（*gid*）は複合レシピを維持しやすくするために存在するものである。*PDE* の参照をすべて一々，*uid* で行うレシピを作ることは可能ではある。しかし，構成要素の 1 つに変更を加える際に問題につながる可能性がある。

7.12.2.4.2 *PDE* に何らかの変更を加えると，それは新たな *uid* を持つ新たな *PDE* にならなければならないことを思い出して頂けるだろうか。かつての *PDE* の，新たなバージョンを使わんがために，元々の *PDE* を参照していた一部（またはすべての）*PDE* は更新された方を参照し直さなければならない。新規バージョンを参照することで，参照する側の *PDE* が変更されると，参照する側の *PDE* はどれも新たな *PDE* となってしまう。これにより更新された *PDE* を参照する *PDE* はどれも同じ問題を抱えることになる。この問題は上記の最初の *PDE* のどの階層にも数珠繋ぎに波及する。

7.12.2.4.3 *PDE* の参照の中である *gid* をもつものは何れも，グループのいかなるメンバをも満足する。*PDE* が変更された場合，*uid* が変化するにもかかわらず同一の *gid* を持ち続けることができる。それ故，元々の *PDE* を参照する *PDE* を変更する必要がない。このことで階層を貫いて繰り返し連なる参照の問題を解決できることになる。

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

7.12.2.4.4 しかし，*gid* を *PDE* の参照に利用することは負担を増やすものではない。何れかの時点で *gid* の参照関係のつながる先は確定する。

7.12.2.5 *PDE 参照先の確定*

7.12.2.5.1 ¶7.12.1.5.1 で *PDE* 参照は *uid* か *gid* の何れかで与えられると論じた。*gid* が参照に用いられる場合，参照を行っている *PDE* が実行に移される前に最終的な *uid* が明らかにならなければならない。*uid* の確定作業は1階層ずつ *Master PDE* から行っていかなければならない。今注目している階層が参照する 1 個または複数の *PDE* が明らかな時に限り，それら *PDE* に関して次の階層を確定させることを行う。

7.12.2.5.2 *gid* を *uid* に確定させる作業は装置か装置に作業を依頼する側が行うことができる。その様に設定されているとすれば，装置は依頼側が行わない *gid* の確定作業を行うことになる。装置がある *gid* の確定作業を行う必要がある場合，装置に現在あるグループの最新の *createDate* を持つ構成要素を選び出す。

7.12.2.5.3 依頼側は処理予定のジョブに関して *gid* を確定しても良く，そこに設定されているパラメータをそのジョブの入力条件とすることができる。ここのパラメータ "*PDEmap*" で依頼側は *gid* のリストに対応する各々の *uid* を指定する。装置はそれによりそのリストの内容を使ってレシピの確定作業を進めることになる。この確定作業の過程で装置が *PDEmap* にない *gid* に遭遇した場合は，上に示した方法で *gid* の確定作業が進められる。依頼側が *PDEmap* を指定しない場合，装置は自ら，現れる *gid* の確定作業を行う。上位の管理下でないところで装置に最終的に PDE を確定させることは薦められない。

7.12.2.5.4 依頼側は最良となる *PDEmap* の値を確定するため，装置とやり取りする。*resolvePDE*() サービスを使うことにより依頼側は *PDE* を準備することができ，対応する（完全に確定した）"*結果のマップ*"を装置から受け取ることができる。結果のマップは確定作業の結果による *gid – uid* の対応リストを表す—*PDEmap* に等価。依頼側は結果のマップに優先的に反映を望む *gid – uid* の対応リストを必要に応じ示すこともできる。このやり取りの例に関しては関連情報 1 に示したシナリオを参照のこと。

7.12.3 *パラメータの抽出*

7.12.3.1 プロセスジョブのパラメータ抽出作業はプロセス制御に関する重要な概念である。プロセスに影響が出る様にするために，装置にどの様にパラメータを渡すことができるかと言った規定は他の SEMI スタンダードにもある（例，SEMI E40 や SEMI E30 参照）。RaP は上記パラメータの定義の支援を行うものである。図 6 に RaP レシピのパラメータの流れを示す。

7.12.3.2 RaP ではレシピはジョブに関する上位の規定の一部として実行される。「プロセスジョブ」が実行される段になると，その規定を満たすべく，設定値をジョブに渡すことができる。RaP ではこの設定値のことを変数パラメータと呼ぶ。図 6 に変数パラメータがプロセスジョブに渡るところを示す。

7.12.3.3 プロセスジョブは *Master PDE* をプロセスレシピとして参照する。プロセスジョブ実行時に，装置は変数パラメータの値を *Master PDE* の *PDEparameters* に合う形にして *Master PDE* に渡す。

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

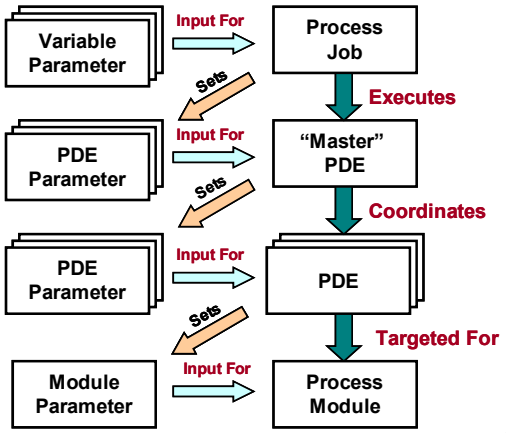

図 6
パラメータの流れの図解

7.12.3.4 *Master PDE* とそれに隣接する階層の *PDE* は続く下位の階層の *PDE* の動作を調整する。

7.12.3.5 各 *PDE* はまた「モジュールパラメータ」を直接セットでき，その設定値はプロセスモジュール（または装置の他の構成要素）に影響を与える。各プロセスモジュールにはレシピから内部でセットできる入力パラメータの固定セットがある。RaP はモジュールパラメータを公式に定義はしないことを注意しておく。このパラメータはレシピが取り扱う装置内部の設定値を現すために使われるものである。

7.12.3.6 ある *PDE* に対して適用される *PDEparameter* の値が外に作用する値として，これとは別の *PDE* の *PDEparameter* あるいはモジュールパラメータの値に影響を及ぼすとことがある。*PDE* 外からもたらされた値が *PDE* 外に作用する値として使われる場合，この効果は直接的である。*PDE* 外に作用する値を決める際にある種の変換が外からもたらされた値になされる場合，この効果は間接的と言える。間接的効果の例としては，*PDE* の内部の経過時間の設定値があり，どのモジュールパラメータを何時セットするのかを決めるのに使われるものである。

7.12.3.7 必要性に応じ，プロセス進行中に *PDE* モジュールパラメータを別の値にセットすることもできることを指摘しておく。例えば，プロセスチャンバが "ChamberTemperatureSetpoint" をモジュールパラメータとして持っているような場合。ある *PDE* はこの値をプロセスの初期段階では 450°C とする様に求め，次の段階では 500°C とする様に求めたりする。そこで *PDE* は直接，最適な時間にその値をセットすることもできる。別の方法として *PDEparameters* をこのために 2 つ，定義することもできる：FirstStageTemp と SecondStageTemp。各パラメータは ChamberTemperatureSetpoint にマッピングされ，それを既定値として設定しておくようにすることもできる。その上で，*PDE* の変更に依らずに実行中にパラメータのセットする方法で，最初の段階の温度を 430°C とすることが定まった時点で変更を加えることができる。

7.12.4 *TransferContainer*

7.12.4.1 ¶7.10 で *PDE* の移動について触れた。RaP は複合した相互関係のある *PDE* にまで及ぶ，プロセスに係わる動作についての定義を行うものである。メッセージ通信の有効性と *PDE* の構造化を進めて行くために RaP では *TransferContainer* を定義することとする。*TransferContainer* は移動用に複数の *PDE* を互いにつなぎ留めるものである。

7.12.4.2 *TransferContainer* 1 個に *PDE* や *PDEbody*（何れも複数も可），ならびに *Manifest* が含まれるものである。*TransferContainer* を保存用ファイル（例えば移動の効率化を考え圧縮を施す "zip" ファイル）に，*PDE* や *PDEbodies* はその内容物のファイルの概念の如くに考えた方が，都合が良い。

7.12.4.3 *TransferContainer* に収納された *PDE* 同士が何らかの関連を持たなければならないと言う訳ではない。若しくは，ある装置の動作に関して必要な *PDE* がすべて一緒に移動するようにしなければいけない訳でもない。しかしながら RaP では分離して存在する *PDEbody* はその元の *PDE*（ならびにその *PDEheader*）と一緒とすることを強く求めるものである。

SUPERSEDED

AVA Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

7.12.4.4 *Manifest* は *TransferContainer* に収められている *PDE* をならべ，各 *PDE* を分離されて外部に存在する *PDEbody*（もしあれば）と関連付ける。*Manifest* は装置上での各レシピの収納場所（例，ディレクトリ）を指定することも可能である。

7.12.4.5 更に詳細については，¶8.4.2.13 で示す。

7.13 *目的の実現*

7.13.1 RaP の目的については §1 で議論した。そこでは目的の主要 6 項目を紹介した。本章ではその 6 項目を再度確認し，§7 が定義した概念でその各々が満足できるかを示すこととする。

- 装置上，装置外部でのレシピの管理 — 各々*PDE* をユニークに識別する能力を RaP は提供するものである。これはホストまたは装置のレシピ管理機能が使用することになる，バージョン管理システムにより独立に行われる。装置上での管理情報（*バージョン*，複写元など）は *PDE* ヘッダに組み込まれるが，ホストはこの情報を取捨選択してながら無視し，自らの管理の仕方に則っても構わない。*PDE* の管理がどの様に行われるかについて，ホストまたは装置に求められる要件は特に指定しないものとする。
- レシピの完全管理 — レシピの完全性を保障するためには，各 *PDE* をユニークに識別でき，何か変更があると，別の違った PDE になるものと捉えられていなければならない。ユニークな識別に関する要件は uuid 値として収容されるユニークな識別子（*uid*），ならびに *PDE* に対するいかなる変更によっても *uid* が変更されると言う要件で達成されるものである。チャックサムは変更が無かったことを示す確認を補足的に示すこととなる。
- プロセスの完全管理 — モジュールパラメータすべてを直接，あるいは実行時に使われる入力パラメータとして *PDE* によりセット可能とすることで，RaP はこの要件を処理するものである。使われそうなパラメータすべてが *PDE* で使用可能であるとは，RaP は保障するものではない。このことはサプライヤやユーザの協力関係で確認する様，委ねるものである。
- 変数パラメータの定義 — RaP は SEMI スタンダード（例，SEMI E40）に沿った方法でプロセスジョブ用にパラメータを定義する，柔軟性のあるやり方を提供する。¶7.12.3 参照。
- 装置上，装置外部でのレシピの作成と編集 — RaP はレシピの作成と編集の機能を備えるよう要求するものである。RaP は装置外の編集機能との通信を標準化するインタフェースを規定する。RaP では装置外部の編集機能を必須とはしない。
- 情報のアクセス — *PDEheader* はユーザがアクセス可能で，個々の PDE に関する情報，ならびにそれら同士がお互いにどう関係するかの情報を多く含むものである。

8 要求事項

8.1 *要件概要*

8.1.1 本章は RaP が規定する要求事項すべてを記載するものである。この要求事項は 3 つに分類される。

- 最初に，レシピの構成要素が収容するデータである内容物を記述するレシピオブジェクトモデル。
- 次の部分は，レシピ管理にかかわる関係者の間で行われる通信を定義する RaP サービス。
- 3 番目は，*RaPnode* の要求事項と詳細説明であり，レシピ管理の関係者各々の責務を定めるものである。

8.2 *レシピオブジェクトモデル*

8.2.1 RaP は階層構造をしたレシピ構成要素から構成される。各レシピの構成要素は *PDE* である。この階層構造で，ある *PDE* はその仕事を遂行するために他の *PDE* を利用することがある。階層構造中，複数回，同一の *PDE* が現れることがあることを注意しておく。その *PDE* は出現する毎に，使われ方は異なる。

8.2.2 階層構造の頂点には単一の *PDE* が存在し，レシピ全体を代表する。この最上位の *PDE* を“*Master PDE*”と呼ぶ。*Master PDE* がそうであるために RaP が PDE に求めるもうひとつのことは *Master PDE* は「実行」属性が「真」であることだけである（表 5 参照）。

SUPERSEDED
Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)

8.2.3　レシピオブジェクトモデルは *PDE* の内容に関して，RaP が課す要求事項を規定するものである。この *PDE* は RaP サービスの対象となるものである（¶8.4 参照）。*PDE* に関するものは *PDE* サービス（例，*getPDEdirectory()*参照）から直接アクセス可能である。この情報はレシピオブジェクトモデルで定義されるものである。

8.2.4　レシピオブジェクトモデルは特に移動中の *PDE* の形態に対して適用するものである。*PDE* の意味が失われることがなく，厳格な移動の形態により，後の移動に対しても再構築可能である限り，RaP はいかなる PDE に対しても保存用の形態についてはいかなる要求事項も規定するものではない。

8.3　*PDE のクラス図*

8.3.1　*PDE* のクラス図（図 7 参照）は *PDE* が収容する情報を表現するものである。実行指示の他に *PDE* はそれ自体の公式な情報を用意する。装置サプライヤが自らに完全性の確保を最終的に任せ課し，実際のプロセス時の指示に関する企業秘密となる値を保護することを認める一方で，このことは *PDE* に関する重要な情報のアクセス権をユーザに与えることを意味するものである。

8.3.1.1　*PDE クラス*

8.3.1.1.1 *PDE* は *PDE* を説明する *PDEheader* と，実行時の指示が格納される *PDEbody* を持つ。*PDEheader* は *PDE* の目的や履歴の様な *PDE* に関する重要な情報を保持する。

8.3.1.1.2 *PDEbody* は *PDE* の内部（*PDEbody* クラスで表す）にあるか，あるいは外部に分離して（*PDEbodyReference* クラスで表す）いるかの何れでも良い。*PDEbody* は *PDE* の内部に置かれなければならないが，その形態が *PDEheader* と同一の書式化の規則に対応している場合に限る。書式化の規則は本書の枝番の規定で定義されるものである。

8.3.1.1.3 *PDE* には状態はない。従って，*PDE* の状態変化に従って報告する様なイベントも定義されない。 但し，装置のレシピの収蔵に変化があった際のイベントに対する議論に関しては¶8.5.4.7 を参照のこと。

8.3.1.1.4 *PDE* およびその *PDEbody* のサイズに制限はない。

8.3.2　*PDEheader クラス*

8.3.2.1　*PDEheader* は *PDEbody* の内容の記録をもつ。*PDEheader* には実行用指示自身は含まれないが，ただ単に *PDEbody* に収められている実行時の指示が反映され，重要な前後の状況と説明用情報が収められる。

8.3.2.2　*PDEheader* は *PDE* が作られる度に *PDEeditor* によって作成されるものである。*PDEheader* と *PDEbody* は一致した組み合わせであって，独立に変更されてはならない。

8.3.2.3　*PDEheader* の属性を表 5 に定義する。*PDEheader* は更に，*AntecedentData*， *ReferencedPDEs*，*PDEparameters*，*ExecutionTargets* の各々のリストで構成される。これらの関連は表 5 に記載し，関連クラスとしてその下で説明する。

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E139-0708 (Japanese)