SEMI E81-0600
PROVISIONAL SPECIFICATION FOR CIM FRAMEWORK DOMAIN ARCHITECTURE

This provisional specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 2, 2000. Initially available at www.semi.org April 1999; to be published June 2000. Originally published June 1999.

1 Purpose

1.1 This document is an overview of the structure and contents of a suite of documents representing an application framework for the Computer Integrated Manufacturing (CIM) systems as used in semiconductor factories. A framework is a software infrastructure that creates a common environment for integrating applications and sharing information in a given domain. The purpose of this framework is to establish an industry standard architecture for complex manufacturing systems, leading to an open, multisupplier CIM system environment. The framework described in this specification is called the CIM Framework.

2 Scope

2.1 The intent of this document is to describe the Manufacturing Execution Systems (MES) domain that is the subject of the CIM Framework and to provide a reference for concepts that are common to the set of documents that specify the CIM Framework. The *Provisional Specification for CIM Framework Domain Architecture* defines the structure, relationships and interworkings of the components that together comprise the CIM Framework. This architecture defines the partitioning of the CIM Framework components and the responsibilities of each of those components. It also specifies the common abstractions for manufacturing jobs, material, and factory resources that are used consistently throughout the CIM Framework as unifying themes.

2.2 The CIM Framework Domain Architecture does not address the dependencies on computing technologies needed to implement these components. These aspects apply more to the realization of the components as software artifacts than to their functionality in terms of semiconductor manufacturing concepts. The technical aspects of the CIM Framework architecture are captured in a separate document, SEMI E96, Guide for CIM Framework Technical Architecture.

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

3 Limitations

3.1 The CIM Framework Specification must continue to evolve to meet the needs of a competitive and vital industry. The content of this framework represents a significant amount of real development experience from a number of commercial software suppliers and their customers. These specifications reflect the product architectures of those companies, as well as the requirements of their customers.

3.2 As a SEMI Provisional Standard, the Specification for CIM Framework Domain Architecture has specific deficiencies that must be addressed before it may be upgraded to full SEMI Standard status. These deficiencies are:

- Ensuring consistency with the details of subsequent related specifications that are based on this domain architecture.
- Evolving from coarse-grained component partitions to fine-grained components that provide substitutability of smaller components.
- Expanding interfaces to include build-time configuration functions.
- Providing fully validated models using the standard Unified Modeling Language (UML) notation.
- Aligning the CIM Framework representation of equipment and interfaces for interactions with equipment automation software with emerging standards in areas such as Object-Based Equipment Model (OBEM) and Automated Material Handling Systems (AMHS).
- Modifying the CIM Framework use of the "in" parameter mode and operation return value to include also the "out" and "inout" modes to better accommodate implementations based on Microsoft DCOM and IDL enhancements for pass-by-value of objects.

- Adjusting the functional partitioning of the Domain Architecture to reflect the final positioning of sub-components in the anticipated revisions of the other CIM Framework specifications.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E10 — Standard for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM)

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

SEMI E32 — Material Movement Management (MMM)

SEMI E42 — Recipe Management Standard: Concepts, Behavior, and Message Services

SEMI E58 — Automated Reliability, Availability, and Maintainability Standard (ARAMS): Concepts, Behavior, and Services

SEMI E86 — Provisional Specification for CIM Framework Factory Labor Component

SEMI E93 — Provisional Specification for CIM Framework Advanced Process Control Component

SEMI E96 — Guide for CIM Framework Technical Architecture

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

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

4.2 *OMG Documents*[1]

CORBA — Common Object Request Broker Architecture, Version 2.3.1 (OMG Document formal/99-10-07).

MfgDTF — Manufacturing Domain Task Force Roadmap, Version 3.1 (OMG Document mfg/98-06-11).

OMA — Object Management Architecture Guide, Version 3.0 (OMG Document ab/97-05-05).

UML — UML Notation Guide, Version 1.1 (OMG Document ad/97-08-05).

Workflow — Joint Workflow RFP Revised Submission (OMG Document bom/98-06-07).

1 Object Management Group, 492 Old Connecticut Path, Framingham, MA 01701, USA

4.3 *SEMATECH Documents*[2]

CIMArch — Computer Integrated Manufacturing (CIM) Framework Architecture Concepts, Principles, and Guidelines, Version 1.0 (SEMATECH-Technology Transfer #97103379A-ENG).

CIMFW — Computer Integrated Manufacturing (CIM) Application Framework 2.0 (SEMATECH Technology Transfer #93061697J-ENG).

4.4 *Other References*

ALBUS — J.S. Albus and A.M. Meystel, *A reference model architecture for design and implementation of intelligent control in large and complex systems*, International Journal of Intelligent Control and Systems vol. 1, no.1 p.15–30, World Scientific: Singapore, March 1996.[3]

ANSI — ANSI Standard ANSI/ISA-S88.01-1995, Batch Control Part 1: Models and Terminology[4]

COM+ — http://www.microsoft.com/msj/1197/complus.htm; http://www.microsoft.com/com/[5]

DCOM — http://www.microsoft.com/windows/downloads/bin/nts/dcom_architecture.exe.[5]

JAVA — http://www.javasoft.com.[6]

WfMC — http://www.wfmc.org.[7]

NOTE 1: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *AMHS* — Automated Material Handling System

5.1.2 *APC* — Advanced Process Control

5.1.3 *APCFI* — Advanced Process Control Framework Initiative

5.1.4 *API* — Application Programming Interface

5.1.5 *CIM* — Computer Integrated Manufacturing

5.1.6 *MES* — Manufacturing Execution System

2 SEMATECH, 2706 Montopolis Dr., Austin, TX 78741, USA

3 World Scientific Publishing Co., 1060 Main St., River Edge, NJ 07661, USA

4 American National Standards Institute, 11 West 42nd St., New York, NY 10036, USA

5 Microsoft, 10500 NE 8th St., Ste. 1300, Bellevue, WA 98004, USA

6 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, CA 94303, USA

7 Workflow Management Coalition Office, 2 Crown Walk, Winchester, Hampshire, S022 5XE, United Kingdom

5.1.7 *MMMS* — Material Movement Management Services (SEMI)

5.1.8 *OBEM* — Object Based Equipment Model

5.1.9 *OMA* — Object Management Architecture

5.1.10 *PFC* — Process Flow Context

5.1.11 *PFI* — Process Flow Iterator

5.1.12 *RFP* — Request for Proposal

5.1.13 *RMS* — Recipe Management System

5.1.14 *UI* — User Interface

5.1.15 *WIP* — Work In Process

5.2 *Definitions*

5.2.1 *abstract interface* — an interface defined outside any component that generalizes common features of the CIM Framework. The abstract interfaces are intended for use in multiple components via interface inheritance mechanisms.

5.2.2 *application* — 1. One or more programs consisting of a collection of interoperating objects which provide domain specific functionality to an end user or other applications. 2. Functionality provided by one or more programs consisting of a collection of interoperating objects.

5.2.3 *application framework* — a framework that constitutes an application or a set of applications for a domain area.

5.2.4 *application interface* — the interface provided by an application or application program.

5.2.5 *application object* — an object implementing an application interface.

5.2.6 *architecture* — the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time.

5.2.7 *attribute* — an identifiable association between an object and a value. An attribute may have functions to set and retrieve its value.

5.2.8 *behavior* — the effects of performing a requested service including its results.

5.2.9 *binding* — a specific choice of platform technologies and other implementation-specific criteria.

5.2.10 *class* — the shared common structure and common behavior of a set of objects. Class often implies an implementation of the common structure and behavior while interface represents a specification of those common features.

5.2.11 *client* — an object that uses the services of another object by operating upon it or referencing its state.

5.2.12 *collection* — an object containing references to (collections of) other objects with services for managing them and providing access to them as a related group of objects.

5.2.13 *component* — a reusable package of encapsulated objects and/or other components with well-specified interfaces. The component is the element of standardization and substitutability in the CIM Framework.

5.2.14 *Computer Integrated Manufacturing (CIM)* — an approach that leverages the information handling capability of computers to manage manufacturing information and support or automate the execution of manufacturing operations.

5.2.15 *conformance* — adherence to a standard or specification in the implementation of a product, process, or service.

5.2.16 *conformance requirement* — identification in the specification of behavior and/or capabilities required by an implementation for it to conform to that specification.

5.2.17 *conforming implementation* — an implementation that satisfies all relevant specified conformance requirements.

5.2.18 *distributed system* — an integrated collection of several processing and memory components whose distribution is transparent to the user so that the system appears to be local.

5.2.19 *domain interface* — an interface specific to an application subject area.

5.2.20 *domain object* — an object implementing a domain interface.

5.2.21 *events* — an asynchronous message denoting the occurrence of some incident of importance. For example, state change or new object created.

5.2.22 *event channel* — the intermediate object that forwards published events to interested subscribers.

5.2.23 *exception* — an infrastructure mechanism used to notify a calling client of an operation that an unusual condition occurred in carrying out the operation.

5.2.24 *extensibility* — the ability to extend or specialize existing components and add new object classes or components while preserving architectural integrity and component conformance to standards.

5.2.25 *framework* — a collection of classes or components that provide a set of services and functionality for a particular domain.

5.2.26 *implementation* — the internal view of a class, object or module, including any non-public behavior. The specific code and functionality that implements an interface.

5.2.27 *infrastructure* — the services, facilities, and communications mechanisms that support the collaboration between and lifecycle of distributed objects.

5.2.28 *inheritance* — a relationship among classes wherein one class (a subclass) shares the structure or behavior defined in one or more other classes (superclass). A subclass typically specializes its superclasses by augmenting or redefining existing structure and behavior.

5.2.29 *instance* — a software entity that has state, behavior and identity. The terms instance and object are interchangeable. An object is an instance of an interface if it provides the operations, signatures and semantics specified by that interface. An object is an instance of an implementation if its behavior is provided by that implementation.

5.2.30 *interface* — the external view of a class, object, or module that emphasizes its abstraction while hiding its structure and internal behavior. An interface definition ideally includes the semantics.

5.2.31 *interface inheritance* — the construction of an interface by incremental modification of other interfaces (see implementation inheritance). The CIM Framework specifies interface inheritance but not implementation inheritance.

5.2.32 *interoperability* — the ability for two applications or the parts of an application to cooperate. In the CIM Framework, interoperability requires that application components be able to share data, invoke each others' behavior (services), exchange events, and publish service exceptions.

5.2.33 *job* — some system level operation whose execution may be requested by an entity whose responsibility it is to manage jobs. The job concept is analogous to operations performed on the "factory floor" in a physical factory. There, operators are requested to perform operations (jobs) requested by their managing supervisors or some other managing source. A job often spans a significant amount of time and multiple resources within the system. In the CIM Framework, the job construct is intended for specialization to enable specific job supervisors and jobs to provide system solutions.

5.2.34 *lifecycle* — the life of an object, including creation, deletion, copy, and equivalence.

5.2.35 *method* — an operation upon an object defined as part of the declaration of a class. In general, the terms message, method and operation can be used interchangeably. Technically, a method is defined within a class and an operation is defined within the IDL. An operation is implemented by a method.

5.2.36 *object* — an identifiable encapsulated entity that implements one or more services that can be requested by a client. An instance of a class.

5.2.37 *object services* — interfaces for general services that are likely to be used in any program based on distributed objects.

5.2.38 *Object Management Group (OMG)* — an international consortium dedicated to the development of open specifications for distributed, heterogeneous, object-oriented systems.

5.2.39 *operation* — an operation is an entity, identified by an operation identifier that denotes a service that can be requested. An operation has a signature that describes the legitimate values of request parameters and returned results, including any exceptions.

5.2.40 *persistent object* — an object that can survive the process or thread that created it. A persistent object exists until it is explicitly deleted.

5.2.41 *process definition* — information characterizing manufacturing processes including an estimate for the time a process resource will be engaged in the process; process resource settings; and the process capabilities required for the process.

5.2.42 *process flow* — the part of a product specification that defines the sequence of process steps for the manufacturing of a specific product. The data structure for representing a process flow is the directed graph; specifically, a tree structure. The nodes of the tree are called process flow nodes (see below). Services are required to navigate the process flow.

5.2.43 *process flow context* — navigational information pertaining to a product's progress as it traverses its context process flow.

5.2.44 *process step* — the smallest unit of processing activity that can be defined in a process flow. One or more process steps are sequenced to define an operation set.

5.2.45 *recipe* — the pre-planned and reusable portion of the set of instructions, settings and parameters that determine how a job is to be performed. For example, recipes are used to describe Process Steps and are typically contained within a Product Specification.

They determine the processing environment seen by a manufactured product (e.g., wafer). Processing recipes may be subject to change between product runs or processing cycles.

5.2.46 *sub-component* — a component that is fully contained within a larger component. The interfaces of the sub-component may be exposed or hidden by the encapsulating component.

5.2.47 *substitutability* — the ability to replace a given component from one supplier with a functionally equivalent component from another supplier without impacting the other components or its clients in the system.

5.2.48 *type* — a declaration that describes the common properties and behavior for a collection of objects. Types classify objects according to a common interface; classes classify objects according to a common implementation.

6 Overview

6.1 This section provides background information that will help readers get the most from the content of this specification.

6.2 *Intended Audience*

6.2.1 The framework specification is intended to address the needs of the following CIM technologists:

- Technical CIM managers.
- System architects and engineers.
- Application developers and integrators.
- Standards developers.

6.2.2 These groups may be found in a variety of organizations, including semiconductor manufacturers, software product suppliers, system integrators, equipment suppliers, standards organizations, universities, national laboratories, and other research organizations.

6.2.3 *Technical CIM Managers*

6.2.3.1 Technical CIM managers are responsible for managing the development, delivery, and integration of complex manufacturing software applications. They can use the CIM Framework specification to plan and organize the development activities and guide component testing and validation. Moreover, those who buy some of their software from external sources can use it as a purchasing guide when discussing system architecture and integration requirements with potential suppliers.

6.2.4 *System Architects and Engineers*

6.2.4.1 System architects and engineers are responsible for overall system design, including selection of industry standards for computing and communications infrastructure, software development processes, product roadmaps, and related topics. They can make extensive use of the CIM Framework as a starting point for many of their activities, including the

- partitioning and allocation of application functions to specific modules,
- definition of the boundary between the distributed system infrastructure and the rest of the system, and
- specification of open interfaces between the portions of the system they are designing and the external environment.

6.2.4.2 They can also use the CIM Framework specifications to define a strategic system roadmap for migration to an open, distributed system environment.

6.2.5 *Application Developers and Integrators*

6.2.5.1 Application developers and integrators must produce, install, and support software applications for semiconductor manufacturing. The CIM Framework specification, in conjunction with a specific framework "binding" (i.e., target computer system hardware and software technologies), represents a set of detailed design requirements for the application developer. At a minimum, the CIM Framework defines the scope and boundaries of the essential standard components of a manufacturing execution system, and can be used principally as an interface specification. The object models can also be used in the internal design of new applications and/or legacy integration "wrappers," accelerating the development process even further. Finally, the specification can form the basis for creating an independent set of tests necessary to verify conformance.

6.2.6 *Standards Developers*

6.2.6.1 Developers of CIM software standards are responsible for specifying the public interfaces and shared information models that allow the many software products found in a modern semiconductor factory to work together. They can use the CIM Framework as an open source of information for establishing precise definitions for the many items in a factory that must be represented in multiple suppliers' products, including

- standards for partitioning and communicating with complex equipment,
- product and raw material attributes and relationships,

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.