SEMI E133-1108
PROVISIONAL SPECIFICATION FOR AUTOMATED PROCESS CONTROL SYSTEMS INTERFACE

This standard was technically approved by the global Information & Control Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on August 29, 2008. It was available at www.semi.org in October 2008 and on CD-ROM in November 2008. Originally published March 2004; previously published July 2008.

1 Purpose

1.1 In order to more efficiently facilitate the integration of Process Control Systems (PCS) such as run-to-run (R2R) control, fault detection (FD), fault classification (FC), fault prediction (FP), statistical process control (SPC), etc. into current and future fabs, there is a need to define interfaces for Process Control Systems that enable them to interact effectively and share data (1) among themselves and (2) with the other interdependent factory systems (including equipment data collection). This standard is intended for use by equipment suppliers, semiconductor manufacturers, equipment subsystem suppliers and control system suppliers who will utilize the PCS standards in future developed products.

2 Scope

2.1 This standard focuses on defining PCS capabilities, functional groups, and functional group interfaces. The standard is structured so that new functional groups can be added to the standard through future balloting. The components in this document include:

- Definitions of key terms and concepts pertinent to this domain,
- Description of the specification conventions used,
- Identification of the major functional groups within the scope of PCS,
- Description of the capabilities expected in each functional group (required and optional), and
- Definition of interfaces for these functional groups, including:
  - Data that is available across these interfaces. This includes the interface data descriptions (inputs and outputs). Future enhancements to this specification will include detailed data structures in the form of inheritance and aggregation models that are common to all PCS functional groups, as well as specific models for each,
  - Services supported and Interface behavior assumed, and
  - Description of an approach for compliance verification testing.

2.2 The specific format and protocol implementation of these interface specifications, which are technology specific, will be delineated in supporting "dot" specifications to this standard. As an example a possible format is XML (which would include data type definitions and metamodel schemas).

2.3 In addition to the information embodied in this standard, the PCS TF will describe a validation process (prototyping) for the standard to identify and address real implementation issues as early as possible; results of this work will be included in future versions.

2.4 The provisional status of this specification will be removed when the following enhancements are added:

- Interface data structures and services specifications for the R2R, FD, FC, FP and SPC functional groups, and
- Additional compliance verification procedures for the R2R, FD, FC, FP and SPC functional groups.

2.5 Note that conformance with this standard as defined in § 10 is not enforceable until the complete data structures have been defined for at least one functional group.

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

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

3 Limitations

3.1 The proposed PCS standards will **not** focus on:

- Low-level communication mechanisms (e.g., hardware connection schemes, transport layer software, etc.) between PCS functional groups,
- Methodology for implementing actual equipment or factory level control strategies,
- Specifying internal structure or implementation methods of PCS functional groups,
- Capabilities of complementary applications (recipe management, data collection, engineering analysis, data storage, scheduling, change control, …),
- The management of information and collaboration between functional groups. This is left to implementation,
- Communications within equipment or within factory systems,
- Determination of the proper control algorithm, model, or runtime selectable behavior. It is optional for a PCS function to contain "context matching,"
- The creation and management of Process Control System Jobs, or
- Defining test procedures required for validating the standard.

4 Referenced Standards and Documents

4.1 *SEMI Standards*

SEMI E81 — Provisional Specification for CIM Framework Domain Architecture

SEMI E121 — Guide for Style and Usage of XML for Semiconductor Manufacturing Applications

SEMI E125 — Specification for Equipment Self Description (ESD)

SEMI E134 — Specification for Data Collection Management

4.2 *IEC Standard*[1]

IEC 61158-5 — Digital Data Communications for Measurement and Control – Fieldbus for Use in Industrial Control Systems – Part 5: Application Layer Service Definition

4.3 *ISO Standard*[2]

ISO 8601 — Data Elements and Interchange Formats – Information Interchange – Representation of Dates and Times

4.4 *Other Documents*

The Unified Modeling Language Reference Manual, James Rumbaugh, Ivar Jacobson, Grady Booch, 1999/0-201-30988-X/Addison Wesley Professional

The Unified Modeling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson, 1999/0-201-57168-4/Addison Wesley Professional

Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1991 by James Rumbaugh, *et al*

Equipment Engineering Capabilities (EEC) Guidelines (Phase 2.0), Version 2.5, SEMATECH, July 2002

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *XML* — eXtensible Markup Language

1 International Electrotechnical Commission, 3 rue de Varembé, Case Postale 131, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.919.02.11; Fax: 41.22.919.03.00; http://www.iec.ch

2 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

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

5.2 *Definitions*

5.2.1 *Advanced Process Control (APC)*[3] — the manufacturing discipline for applying control strategies and/or employing analysis and computation mechanisms to recommend optimized machine settings and detect faults and determine their cause.

5.2.2 *Analysis Engine (AE)* — a process that utilizes data and possibly operational instructions to produce a response. In the context of this standard, this term is used to encompass the characteristics common to all PCS functional groups.

5.2.3 *context* — a series of attributes that uniquely identifies a manufacturing entity (e.g., wafer, lot, module, tool, reticle) and its status in the manufacturing operation.

5.2.4 *Context Matching (CM)* — the process of comparing and matching the values of a set of attributes that represent the state of a system (e.g., process, product and equipment) to a set of stored or computed values. This is usually done so that a unique action can be specified by the context matching system.

5.2.5 *Fault Classification (FC)* — the technique of determining the cause of a fault once it has been detected.

5.2.6 *Fault Detection (FD)* — the technique of monitoring and analyzing variations in tool and/or process data to detect anomalies. Fault detection includes both univariate and multivariate statistical analysis techniques.

5.2.7 *Fault Detection and Classification (FDC)* — combination of FD and FC.

5.2.8 *Fault Prediction (or Prognosis) (FP)* — the technique of monitoring and analyzing variations in process data to predict anomalies.

5.2.9 *Functional Group (FG)* — a collection of closely related software capabilities that one would expect to be provided as an integrated product.

5.2.10 *Process Control System Job (PCS job)* — a unit of work that could be tracked and have data associated with it. In the context of this standard a process control system job is associated with a unit of work that a PCS Analysis Engine performs; a unique identifier such as an ID number is often utilized to track the process control system job.

5.2.11 *Process Control System (PCS)*[3] — a system capable of performing Process Control, which includes one or more of R2R Control, FD, FC, FP, SPC, or any future Process Control functionality defined in this standard for a PCS functional group.

5.2.12 *Run-to-Run (R2R) Control* — the technique of modifying recipe parameters or the selection of control parameters between runs to improve processing performance. A "run" can be a batch, lot, or an individual wafer.

5.2.13 *Statistical Process Control (SPC)* — the technique of using statistical methods to analyze process or product metrics to take appropriate actions to achieve and maintain a state of statistical control and continuously improve the process capability.

6 Background

6.1 With the development of process control system applications at the factory level and inside the equipment, it is now apparent that the lack of interface standards for these applications has inhibited widespread adoption of process control at the factory level. Moreover, adding process control and integrated metrology capabilities inside the tool requires the ability for the factory level applications to collaborate with tool level applications.

7 Conventions

7.1 This section defines conventions followed in this document.

7.2 *Object Modeling*

7.2.1 *Unified Modeling Language (UML)* — This specification uses UML notation for all class diagrams and for object diagrams provided as examples. No other types of diagrams are used in this specification.

7.2.2 UML class diagrams have clearly defined meanings and are a part of this specification. Detail contained in these diagrams is not always repeated in the text.

3 APC and PCS terms are used interchangeably in some companies. Both can be seen as the umbrella of components for process control.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

7.2.3 *Class Diagrams* — UML Class Charts.

7.2.4 *Name of a Class* — The text capitalizes class names.

7.2.5 *Abstract and Concrete Classes* — Each class is specified as Abstract or Concrete. Abstract classes are not directly implemented. In this specification "implemented" means represented to the factory through the communications interface. All classes defined as concrete may be directly implemented. This specification does not attempt to define equipment control system implementation. In UML class diagrams, abstract class names are shown in italics.

7.2.6 *Class Attribute Definition* — The attributes of a class are defined in table format as illustrated in the table below. Note that the "+" sign in front of attributes in UML class diagrams indicates "public" attributes. All attributes defined in this specification are public.

Table 1 Attribute Table Format

Attribute Name	*Definition*	*Access*	*Reqd*	*Form*
		RO, RW, or NA	Y or N	See list below.

7.2.6.1 *Access* — Attributes may be settable (ReadWrite or RW) or not settable (ReadOnly or RO) through the interface services. If the attribute refers to a component of a message (e.g., that is communicated to/from a PCS analysis engine), the concept of access does not apply and "Not Applicable" (NA) is used in the Access field.

7.2.6.2 *Reqd* — Is this attribute required? Y — Yes, or N — No.

7.2.6.3 *Form* — Defines the data type of the attribute. Data types specified in the Process Control Systems specification are described in the subsections that follow. Data types in this specification are high-level definitions and should be mapped to the data types of a specific technology as appropriate. Extensions to the Process Control System interface specifications are not limited to the list of data types defined in this specification. See the definition of "Form" in the SEMI Compilation of Terms for other Form values commonly used in SEMI documents.

7.2.6.4 *Constraints*

7.2.6.4.1 Constraints in UML are shown with dotted lines, typically with a note attached to explain the constraint. Constraints shown in this way tend to clutter large UML diagrams. Therefore, in this specification, constraints are shown only on the diagrams focusing on individual classes.

7.2.7 *UML Associations*

7.2.7.1 The mechanism used for representing associations in UML between classes is implementation dependent. This document is abstract in nature, and does not specify or imply any such mechanism. Any adjunct standard that provides an implementation of this specification must include a description of the mechanism used for representing UML associations used in this document.

7.2.8 *Class Association Table* — This table provides an example of the tables used to list and describe associations between classes defined in this specification.

Table 2 Association Table Format

Association Role Name	*Role Definition*	*Comments*

7.2.8.1 *Association Role Name* — The name of the association role being defined.

7.2.8.2 *Role Definition* — Describes the function or purpose of the association.

7.2.8.3 *Comments* — Any additional comments or notes regarding the association.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

7.3 *Data Types*

7.3.1 The following data types may be used in this specification to describe the form of an attribute or other similar high level definitions. Note that extensions to the Process Control System interface specifications are not limited to the list of data types defined in this specification.

- *Any* — The format may be any of the others listed in this section. Format is determined by the implementation.
- *Binary* — A sequence of bytes that may have any value. Binary values are sometimes called "unformatted" because their structure is not apparent.
- *Boolean* — Takes the value of "true" or "false."
- *Checksum* — A checksum value calculated on a specific stream of data using a specific method of calculation.
- *Enumeration* — A format that allows a specified list of possible values. In this document, these values are represented as text strings, but may be implemented differently (e.g., as integers that correspond to the named values).
- *Status* — The Status type is a structure that contains information about whether or not an error that has occurred in a transmission and information on the type of error. The form of the structure is given in § 9.
- *Integer* — A numeric value, Integers are always whole numbers. The form and number of bytes is left to the implementation definition.
- *List of xxx* — An item that can hold multiple instances of a specified type (where xxx is the data type).
- *Real* — A numeric value that may represent any whole or fractional number. The form and number of bytes is left to the implementation definition.
- *Text* — A text string. Limitations on length are left to the implementation technology unless otherwise specified in this document. Strings are recommended to be implemented as UTF-8.
- *Time* — Representation of the date and time of the occurrence of interest. The structure and form of items of this type is left to the implementation.

7.4 *Service Message Representation*

7.4.1 *Service Resource Definition*

7.4.1.1 The service resource definition table defines the specific set of messages for a given service group as shown in the following table:

Service	*Reqd*	*Type*	*Description*
Service name	Y/N	N or R	The intent of the service

7.4.1.2 Required can be Yes (Y) or No (N). Type can be either Notification (N) or Request (R). A notification service consists of messages initiated by the service provider. No response is expected. A request service consists of a request message initiated by the service user which generally asks for data or for an operation to be performed, and a related response message returned by the service provider.

7.4.2 *Service Parameter Definition*

7.4.2.1 The service parameter definition table defines parameter detail for each service in which parameters are explicitly specified, as shown in the following table:

Parameter	*Request*	*Response*	*Form*	*Description*
Service parameter name	(see below)	(see below)	Data type	The intent of the parameter

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

7.4.2.2 The Request and Response entries can take on the following values:

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 value of the other parameter.
-	The parameter is not used.
=	The entries M and C in the response can be modified with (=) to indicate that the value in the response must match the request.

7.5 *Behavior Representation*

7.5.1 *Sequence Behavior* — Where appropriate, this document uses the Harel State Chart notation to describe the dynamic behavior of the Analysis Engines. An overview of this notation is presented in an appendix of SEMI E30. The formal definition of this notation is presented in Science of Computer Programming 8, 'Statecharts: A Visual Formalism for Complex Systems," by D. Harel, 1987.

7.5.2 *State Behavior* — Where appropriate, transition tables are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. Each transition table contains columns for transition #, "Previous State," "Trigger," "New State," "Actions," and "Comments." The trigger for the action occurs while in the "Previous State." The "Actions" include a combination of: (1) actions taken upon exit of the "Previous State," (2) Actions taken upon entry to the "New State," and (3) actions taken which are most closely associated with the transition. No differentiation is made.

8 Overview

8.1 Using the simple definition of a Functional Group given in the Terminology section, the initial set of PCS functional groups has been identified. To identify the proper scope (and therefore the boundaries) for these groups, the following criteria are applied to test whether or not a given set of capabilities constitutes a valid PCS functional group. Specifically, a functional group should satisfy one or more of the following requirements:

- Support a coherent set of tasks
- Share common set of data and must use it under similar constraints (performance, retention, etc.)
- Internal objects are tightly coupled
- Legacy implementation/integration constraints exist (e.g., the capabilities are inseparable because an external system must interact with them as an integrated whole)
- Strong associations exist in the market (i.e., one would expect these capabilities to be provided as an integrated product)

8.2 Based on these criteria, and the stated implementation priorities of the semiconductor manufacturers and suppliers involved on the task force, the following initial PCS functional groups have been identified:

- R2R (Run-to-Run) Control
- FD (Fault Detection)
- FC (Fault Classification)
- FP (Fault Prediction)
- SPC (Statistical Process Control)

8.3 Note that it is understood that other PCS functional groups can be added as needed. Future versions of the document will describe procedures for adding a new group.

8.4 The aforementioned groups can be implemented at the tool or factory level or both. As a requirement, functional group interfaces should be defined in a consistent way. The following diagram represents the proposed functional groups and an example of their interactions within the factory and tool environments.

8.4.1 Process Control Systems may consist of a variety of capabilities or functions at both the factory and equipment level as shown in Figure 1. Moreover, this may include multiple instances of the same function type from

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

different suppliers. In the example above, several run-to-run controllers are running at the factory level with differences that are only visible as internal special capabilities. This interface standard will facilitate easier collaboration between the tool level functions and factory level systems independent of the supplier that developed the internal function's capabilities.

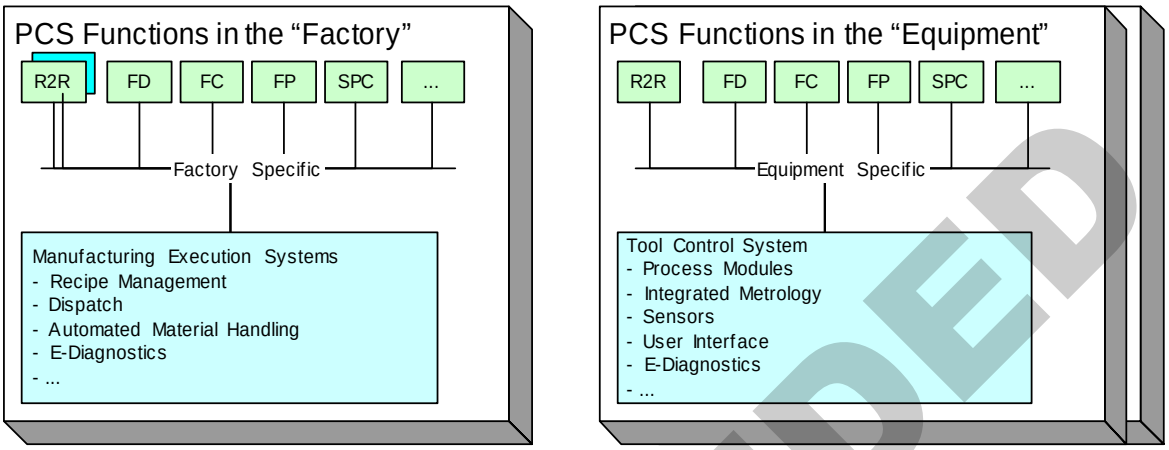

Figure 1
PCS Functions

SUPERSEDED

9 Process Control Systems Functional Group Capabilities and Interface Data Structures

9.1 *Functional Group Partitioning*

9.1.1 Process control systems are partitioned into functional groups using the criteria defined in § 8. This standard defines the interface data structures, services and behavior associated with these functional groups. In this section the capabilities of the PCS functional groups are defined in more detail. First of all, the basic capabilities common to all functional groups are presented. Then the additional basic capabilities unique to each functional group are presented. A high level description of the specifications of data structures, services and behavior is provided in this section.

9.1.2 It is important to note that the specific implementations of these various functional groups may differ greatly in the level of capability delivered, performance, quality of service, and internal architecture. Moreover, in a given factory, there may be multiple instances of the same functional group running concurrently, from one or more suppliers. Some functional groups may be completely independent of other PCS functional groups (in other words, they can operate in relative isolation, depending only on information from systems external to PCS); others may depend heavily on other functional groups, collaborating closely to provide a higher level of capability than they could on their own.

9.1.3 The goal of the PCS interface standards is to enable all of these implementation scenarios (and more) without dictating internal product architectures.

9.2 *PCS Analysis Engine*

9.2.1 As defined in § 8, all functional groups can be thought of as specializations of a PCS Analysis Engine and therefore inherit the structure and behavior of a PCS Analysis Engine. This section defines the common capabilities of a PCS Analysis Engine.

9.2.2 *General Capability Description*

9.2.2.1 All PCS Analysis Engines shall have inputs and outputs as shown in Figure 2. This means that the PCS Analysis Engines shall support the consumption of the specified inputs and production of the specified outputs as necessary to achieve the required capabilities of that analysis engine as specified in this standard. These inputs and outputs are specified as aggregate data structures. The structures are aggregated into "collection types" according to the category of data. A definition of these collection types is provided in Table 4. The PCS Analysis Engine shall

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

have the general behavior of utilizing specified inputs, performing tasks based to some degree on these inputs, and generating outputs. Any additional structure and behavior of various PCS Analysis Engines are specified in the reminder of this section.

9.2.2.2 Note that, in addition to limitations of specification defined in § 4, the following are beyond the scope of definition of the PCS Analysis Engine:

- The level of synchronization of inputs with performance of tasks and/or generation of outputs, and
- The specification of any internal states of the PCS Analysis Engine.

9.2.2.3 All PCS Analysis Engines belonging to this functional group shall have the capabilities listed in Table 5.

Figure 2
PCS Analysis Engine Common Input/Output Structure

9.2.2.4 Note that based on the simple data model shown in Figure 2, Inputs are available as an Output from an Analysis Engine. Data provided as input to an Analysis Engine should be accessible as output from the Analysis Engine for historical purposes.

Table 3 PCS Analysis Engine Inputs and Outputs Collection Type Definitions

Collection Type	*Definition*	*Comments*
Globals	Information that applies globally across an entire job.	A lot id, for example could be specified once as a global so that it would not have to be specified with many other data sets.
AnalysisParameters	Information that identifies numeric data used by the Analysis Engine to perform various calculations.	Equation coefficients, feedforward data, etc.
CalculatedOutputs	Data representing the results of analysis from an Analysis Engine.	A calculation derived from various inputs to the Analysis Engine indicating wafer state, process state or model state information, recommended machine settings, etc.
LogMessages	Data describing the activity of an Analysis Engine as it executes its tasks.	Data describing the activities of the Analysis Engine. Note that this does not include log information on the communication protocol with the AE. An example that is acceptable is “Algorithm #3 with model parameters ‘A’, ‘B’, and ‘C’.”

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

Collection Type	*Definition*	*Comments*
ModuleData	Information representing the current or last known physical state of a process module that will be used by an Analysis Engine to process material.	Machine, or module E10 state, RF hours, number of wafers processed, etc.
ProcessData	Information representing the process itself, such as the name and type of the process and the current or last known process state based on measurements from sensors of the processing environment.	Name of the process (CMP), in-situ process measurements, trace data, and summary statistics etc.
SubstrateData	Information about the substrate being associated with the Analysis Engine. This could include data representing a single wafer, multiple wafers, or one or more lots, as well as attributes of the material.	Profile or topology data, and other characteristics measured physically or electrically.

Table 4 PCS Analysis Engine Functional Group Capabilities

#	*Description*	*Reqd*	*Comments*
3.1	**Provide a PCS Analysis Engine capability** PCS Analysis Engine shall be capable of performing one or more of R2R control, FD, FC, FP, SPC, or any future functionality defined in this standard as a PCS functional group.	Y	
3.2	**Provide the required PCS Analysis Engine methods** The PCS analysis engine shall provide the required interface services as defined in this standard.	Y	These include PCSAnalyze and PCS UpdateModel.
3.3	**Maintain models of selected process control system environment** The PCS Analysis Engine shall maintain explicit or implicit models of the selected process control environment. These models shall be utilized to provide the analysis or control capability of the Analysis Engine.	Y	The specific form and behavior of these models is beyond the scope of this standard.
3.4	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action** The PCS Analysis Engine shall support an input mechanism whereby the analysis engine can be instructed to take a specific unique action.	Y	The mechanism could be a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model and generation of a PCS result, update of a model, or communication of model parameters.
3.5	**Interface with multiple external systems** The PCS Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.
3.6	**Edit configuration of models** The PCS Analysis Engine shall provide the capability to create/add, edit configuration, and delete models.	Y	This capability will be supported through the Analysis Engine interface.
3.7	**Interface with other Analysis Engines** The PCS Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	May be a requirement of Analysis Engines belonging to a specific functional group.
3.8	**Advanced Model Management** The PCS Analysis Engine may provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

9.2.3 *Common Interface Data Structures*

9.2.3.1 *Common Data Model Overview* — The common interface data structures from which all PCS functional groups may inherit are shown in Figure 3. This figure shows the basic types of information ("classes" in object terminology) that can be supported by the interface. This model applies to the structure of inputs as well as outputs of a PCS analysis engine as shown in Figure 2. This model applies to a complete message associated with a PCS job. That is it applies to the complete set of information communicated to or from a PCS AE for a PCS job. Note that a complete message into or out of a PCS AE may be broken into multiple application-to-application communications as defined in ¶ 9.2.3.5.

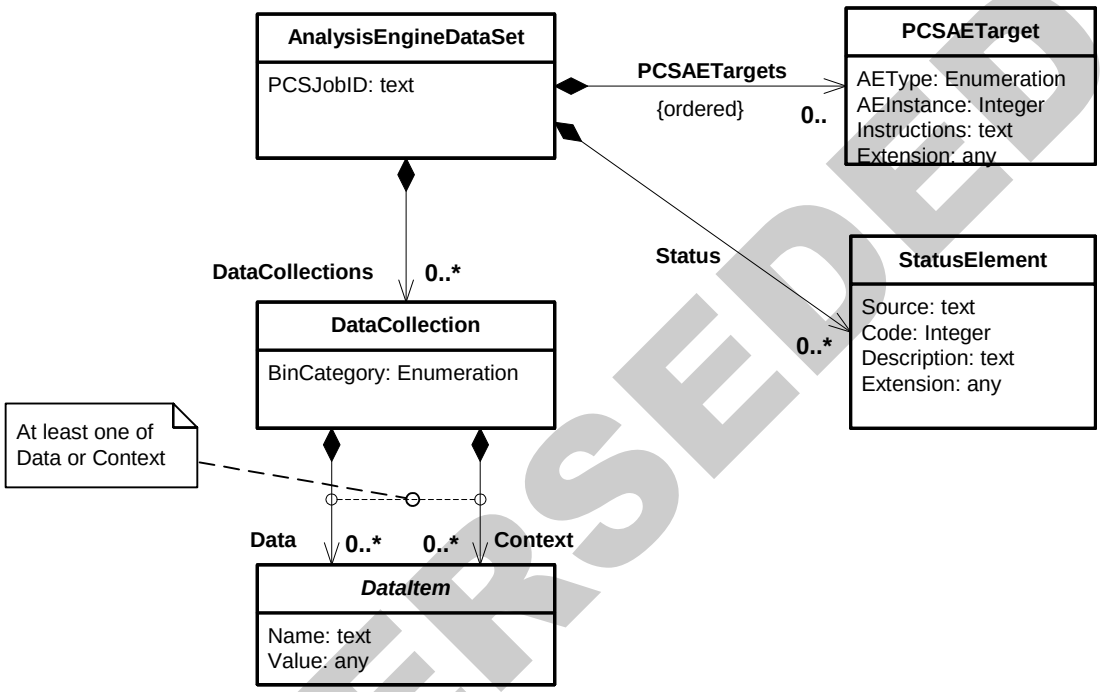

Figure 3
Analysis Engine Data Structure Model for a Complete Message Associated with a PCS Job

9.2.3.2 *Class Definitions* — Class definitions consist of a brief description of each class, a table defining any attributes belonging to that class, and further text as necessary to fully define the class.

9.2.3.2.1 *AnalysisEngineDataSet* — A composite data structure that represents the data associated with a single input or output message of an AE.

Table 5 Attributes of AnalysisEngineDataSet

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
PCSJobID	A unique identifier of a Process Control System Job.	NA	C	ObjID — reference SEMI E39.
PCSAETargets	An attribute that indicates a list of AE types targeted by the AnalysisEngineDataSet.	NA	Y	An ordered list of structured data of type PCSAETarget; see below.

SUPERSEDED
Documents
Document Sharing Hub
Full document details: SEMI E133-1108

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
Status	An attribute formatted as an extensible data structure that relates properties of the success or failure of a PCS AE, identified in PCSAETargets, to execute a PCS job.	NA	C	An unordered list of structured data of type StatusElement; see below.

9.2.3.2.1.1 *PCSJobID* — The specification that PCSJobID is a required attribute of AnalysisEngineDataSet along with the specification that it is a unique identifier of a PCS job means that a unique PCSJobID shall be associated with each PCS job. The management of the uniqueness of the PCSJobID can be performed by an AE or by an application that wishes to utilize the capabilities of an AE. The application communicating with the AE is assumed to generate and manage PCSJobIDs for all PCS jobs, however the AE is required to provide this functionality in the event that the application does not wish to provide it. The existence of the PCSJobID attribute is conditional upon the following:

- If the application communicating with the AE is managing a PCSJobID for a PCS job, then that PCSJobID shall be provided as the value of the PCSJobID attribute for all interface data structures of type AnalysisEngineDataSet associated with that PCS job.
- If the AE is providing the functionality of generating a PCSJobID for a PCS job, then the PCSJobID shall be provided in the first communication from the AE (to any consuming application) for this job and all subsequent communications associated with this job.
- The PCSJobID is not required for transactions that do not involve the AE executing a Process Control Job such as status checks, or pre-processing queries.

9.2.3.2.1.2 *PCSAETargets* — This attribute indicates a list of AE types targeted by the AnalysisEngineDataSet; it is structured as an ordered list of type PCSAETarget (see below). At most one PCSAETargets value shall be associated with Each PCSJobID. Thus any two messages received by a PCS AE with the same PCSJobID will have at most one value for PCSAETargets, which could be provided in either or both messages. Each AE that receives this parameter in an input AnalysisEngineDataSet will echo the parameter and its value for all output AnalysisEngineDataSet messages with that PCSJobID.

9.2.3.2.1.2.1 *PCSAETarget* — This attribute class represents an AE targeted by the current PCS job. The PCSAETarget attribute is used to differentiate PCS AE targets when a single message travels between multiple AE's (often referred to as a "daisy chain"). In this case the AEType and AEInstance attributes of the PCSAETarget (see below) can be utilized to identify a particular consumer for the AnalysisEngineDataSet data. Note that in many cases, such as point-to-point communications between an AE client and a single AE, the PCSAETarget attribute may not be necessary.

9.2.3.2.1.2.1.1 This attribute has the following form:

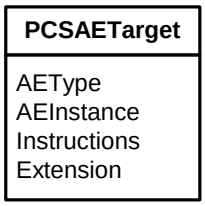

Figure 4
PCSAETarget Data Structure

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

Table 6 PCSAETarget Attribute Definition

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
AEType	An enumeration that represents the analysis engine type to which the instruction is targeted.	NA	N	Enumeration, with possible values being ("FDD" — Fault Detection, "FCC" — Fault Classification, "FPP" — Fault Prediction, "R2R" — Run-to-Run Control, "SPC" — Statistical Process Control, "PCS" — Process Control System).
AEInstance	A number indicating the specific instance of an analysis engine within a type that is targeted.	NA	N	Integer.
Instructions	An ordered list of Instructions, where each instruction is of type text.	NA	Y	Ordered list of elements of type text.
Extension	This attribute allows the supplier to include additional information regarding the targeted AE instance.	NA	N	Any.

9.2.3.2.1.2.1.2 The specification of how the AEs will consume the information is beyond the scope of this standard.

9.2.3.2.1.2.1.3 The AEType enumeration of "PCS" is used to indicate a generic PCS type, or is an unspecified type.

9.2.3.2.1.2.1.4 An AEInstance value is not specified indicates that the instance of AEType is unspecified.

9.2.3.2.1.2.1.5 When multiple PCSAETarget instance values are specified, the target PCS application to which these values refer should execute the PCS job associated with the PCSJobID in sequence. The mechanism for ensuring this behavior is beyond the scope of this standard.

9.2.3.2.1.2.1.6 Instructions is an ordered list of elements, each of which is an instruction and is of type text. Note that no two elements in the ordered list can have the same value. An *instruction* is an attribute that relates a specific task that is being requested to be carried out by a targeted AEInstance.

9.2.3.2.1.3 *Status* — This conditional attribute class relates properties of the success or failure of a PCS AE, identified in PCSAETargets, to execute a PCS job. The attribute has the following form:

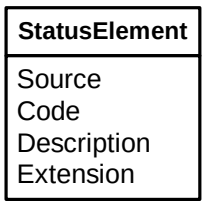

Figure 5
StatusElement Data Structure

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

Table 7 StatusElement Attribute Definition

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
Source	Represents the document from where the attribute code value is defined. For example "Source" could identify a SEMI standards document. For status codes defined by SEMI standards, the content of Source is the urn representation of the standard number. For status codes defined by a supplier, it is the urn representation of the supplier's name. (e.g., for SEMI E132 errors, the content of "Source" is: urn:semi-org:E132; for supplier company errors, for instance, where company name is "Acme," the content is: urn:Acme-com) Additional information may be included by the supplier by adding such information after the ":" colon symbol and separating the fields with a "." period symbol. Please refer to SEMI E121 for additional information.	NA	N	Text.
Code	A value that directly maps to a particular specification status code list. The source of this list is beyond the specification of this document. A status code of zero indicates that no error has occurred, while any non-zero status indicates an error has occurred. Specific error codes defined for this standard are provided in Table 8.	NA	Y	Integer.
Description	A human readable description of the corresponding code attribute being reported. It maps directly to the definition of the Code attribute.	NA	N	An unordered list of structured data; see below.
Extension	This attribute allows the supplier to include additional information regarding the status code encountered. Additional information may be supplied in this attribute to provide more detail about any error (e.g., Common Component, Code 5001, "Insufficient Arguments Provided:Argument "SessionID" missing").	NA	N	Any.

Table 8 SEMI E133 Code Attribute Value Definitions

StatusElement Parameter			
Source	*Code*	*Description*	*Extension*
SEMI E133	0	No error has occurred	<manufacturer specific>
SEMI E133	≠ 0	An error has occurred	<manufacturer specific>
SEMI E133	−1	Message syntax error (e.g., XML validation error, see SEMI E133.1)	<manufacturer specific>
SEMI E133	−2	Unknown or unexpected PCSJobID	<manufacturer specific>
SEMI E133	−3	Unknown or unexpected service	<manufacturer specific>
SEMI E133	−4	Unknown or unexpected PCSAETarget	<manufacturer specific>
SEMI E133	−5	Service not supported	<manufacturer specific>
SEMI E133	−6	Service unavailable at this time	<manufacturer specific>
SEMI E133	−7	Unknown or unexpected data item	<manufacturer specific>
SEMI E133	−8 to −31	<Not specified: reserved for future generic AE use>	<manufacturer specific>
SEMI E133	−32 to −99	<Reserved for specific AE type use; reserved unless otherwise indicated in AE functional group specification>	<manufacturer specific>
#1	<−99	Manufacturer specified	<manufacturer specific>
#1	>0	Manufacturer specified	<manufacturer specific>

#1 Manufacturer Specific.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

9.2.3.2.1.3.1 The existence of the StatusElement attribute is conditional on the properties of the message containing a data structure of type AnalysisEngineDataSet. The StatusElement attribute shall exist if and only if the following conditions apply:

- The message is an output from an AE.
- The message is a response to a request for service(s) from that AE (e.g., a reply message).

9.2.3.2.2 *DataCollection* — A set of data and associated context for that data that conforms to the PCS AE common input/output structure as defined in ¶ 9.2.2.1.

Table 9 Attributes of DataCollection

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
BinCategory	An enumerated identifier of the type of data collection among the possible types defined in the collection type enumeration, listed in Table 4.	NA	Y	CollectionType — As defined.

9.2.3.2.2.1 *CollectionType* — An enumeration of BinCategory as defined in Table 4.

9.2.3.2.3 *DataItem* — An individual component of a data collection that has meaning in the PCS environment. Each data item has a role of either "context" or "data" as determined by the client of the PCS AE.

Table 10 Attributes of DataItem

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
Name	Human-readable name for DataItem.	NA	Y	StringValue — See rules for naming below.
Value	An assigned or calculated quantity.	NA	Y	AbstractValue — As defined in SEMI E134.

9.2.3.2.3.1 *Name* — This attribute is used to provide a level of understanding of the meaning and purpose of the DataItem. Names shall consist of alphanumeric characters, spaces, hyphens and underscores. Names shall start with an alpha character, and shall not contain leading or trailing spaces.

9.2.3.2.3.2 *Value* — AbstractValue (as defined in SEMI E134) is an abstract base class that is used to represent standard types including: Integer, Real, Boolean, String, Binary, Structure, Array, and Enumerations. In this way, parameter values collected via SEMI E134 may be passed directly into the PCS compliant AE via the value attribute of a DataItem. Refer to SEMI E134 for further definition of the AbstractValue type. AbstractValue base class allows for arbitrarily complex data items to be passed into the AE.

9.2.3.2.3.3 A DataItem that belongs to a DataCollection whose CollectionType is "Globals" may not serve a role of data (i.e., it can only serve the role of context).

9.2.3.2.3.4 It is not required that the PCS AE utilize a data item in the same role (Data or Context) as is specified by the PCS AE client, however the PCS AE is required to maintain the role representation in I/O data structures produced or consumed across a PCS AE interface.

9.2.3.3 *Ordering of Elements within an AnalysisEngineDataSet* — In addition to compliance with a data structure model, this standard also defines rules governing ordering of classes and attributes within each AnalysisEngineDataSet instance, corresponding to the data content of an AE I/O message. Figure 4 provides a general illustration of this ordering. The following are rules governing that ordering.

9.2.3.3.1 The first attribute of any AnalysisEngineDataSet shall be the Process Control System Job ID (PCSJobID), if the PCSJobID value has been determined as defined in ¶ 9.2.3.2.1.

9.2.3.3.2 The first attribute of any DataCollection shall be BinCategory.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E133-1108

9.2.3.3.3 The first DataCollection shall be of BinCategory "Globals" if the AnalysisEngineDataSet contains a DataCollection of that kind. At most one DataCollection shall be of BinCategory "Globals" in any AnalysisEngineDataSet.

9.2.3.3.4 In each DataCollection all DataItems whose role is "Context" shall occur before the first DataItem whose role is "Data" occurs.

9.2.3.4 *Additional Requirements of an AnalysisEngineDataSet*

9.2.3.4.1 If multiple AnalysisEngineDataSet instances are "related," as defined in ¶ 9.2.3.2.1 and both contain the same DataCollection with the same value of BinCategory, and further contain the same DataItem (with the same "Name") within that DataCollection, the last DataItem shall be assumed to be correct. The interpretation of ordering of information as communicated by the underlying protocol is beyond the scope of this document.

9.2.3.4.2 Within an AnalysisEngineDataSet instance or a group of instances that are "related," as defined in ¶ 9.2.2.1, a DataItem may exist within multiple DataCollections that has a role of context. If this condition exists, the following rules shall apply.

9.2.3.4.2.1 Any Context DataItems defined in a DataCollection with BinCategory = "Globals" should be used as Context DataItems for all other BinCategorys in any related message except as defined in ¶ 9.2.3.3.2.2 below.

9.2.3.4.2.2 If a DataCollection with a BinCategory other than "Globals" defines a Context DataItem that is also defined in the Global BinCategory of the current message or a related message, the local BinCategory DataItem value should be used as context (i.e., the local Context DataItem overrides the Global value inside this DataCollection).

9.2.3.5 *Utilizing Multiple Application-to-Applications to Communicate a PCS* — A PCS message may be broken into a number of application-to-application communications or partitions, however it is not required that a PCS AE support this partitioning capability. Each of these communication partitions must conform to the data model of Figure 3, with the following modifications:

9.2.3.5.1 PCSAETarget minimum cardinality is zero.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E133-1108

- PCSJobID == “[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}”
- PCSAETargets
  - APCSAETarget: PCSAETarget
    - AEType == “R2R”
    - AEInstance == 1
    - Instructions
      - Instruction == “PCSAnalyze”
    - Extension == “execute serially”
- Status
  - AStatusElement: StatusElement
    - Source == “urn:semi-org:E132”
    - Code == 0
- ADataCollection: DataCollection
  - BinCategory == “Globals”
    - Role == “Context”
      - ADataItem: DataItem
        - AName: (type == text)
        - AValue: (type == any)
      - ADataItem
        - AName: (type == text)
        - AValue: (type == any)
  - BinCategory == “ModuleData”
    - Role == “Context”
      - ADataItem: DataItem
        - AName: (type == text)
        - AValue: (type == any)
      - …
    - Role == “Data”
      - ADataItem: DataItem
        - AName: (type == text)
        - AValue: (type == any)
      - …
  - BinCategory == “ProcessData”
    - Role == “Context”
      - ADataItem: DataItem
        - AName: (type == text)
        - AValue: (type == any)
      - …
    - Role == “Data”
      - ADataItem: DataItem
        - AName: (type == text)
        - AValue: (type == any)
      - …
  - …

Figure 6
Illustration of Ordering of Information in an AnalysisEngineDataSet Instance
(Partial example; “…” is used to indicate possible continuation in the listed format;
PCSJobID in this example is aligned with specifications in SEMI E120.1)

SUPERSEDED

9.2.4 *Common Interface Services*

9.2.4.1 Except as specified in ¶ 9.2.4.1.1 PCS AEs shall provide services by accepting and producing messages that are compliant with the common interface data structure as defined in § 9.2.3. PCS AEs shall support both Request and Notification services as specified herein. For Request services, both the Request and Response messages associated with the service shall comply with the common interface data structure specified in § 9.2.3, which includes a specification of required and optional elements. The success or failure of performing this service is conveyed in the Response message utilizing the “Code” attribute (see ¶ 9.2.3.2.1.3). The “Code” attribute is required in all response messages associated with request services. For Notification services, the message associated with the notification, whether produced or consumed by an AE, shall comply with the common interface data structure. Each

Documents
Document Sharing Hub
Full document details: SEMI E133-1108