

SEMI E125-0703

PROVISIONAL SPECIFICATION FOR EQUIPMENT SELF DESCRIPTION (ESD)

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 April 11, 2003. Initially available at www.semi.org May 2003; to be published July 2003.

Contents

1 Purpose	2
2 Scope	2
2.1 In-scope	2
2.2 Out of Scope	2
3 Limitations	2
3.1 Provisional Specification	2
3.2 Abstract Model	2
4 Referenced Standards	2
4.1 SEMI Standards	2
4.2 ISO Standards	2
4.3 Other Standards	3
5 Terminology	3
5.1 Abbreviations and Acronyms	3
5.2 Definitions	3
6 Conventions	3
6.1 Document and Notation Conventions	3
7 Background	5
7.1 Motivation	5
8 Overview	5
8.1 Equipment Metadata Organization	5
8.2 Accessing Equipment Metadata	7
9 Interface for Requesting Equipment Metadata	7
9.1 Equipment Metadata Manager	7
10 Equipment Metadata	9
10.1 Management of Equipment Metadata	9
10.2 Equipment Structure	9
10.3 Associations	9
10.4 Parameters	13
10.5 Types	16
10.6 Units	23
10.7 Exceptions	24
10.8 Software Modules	25
10.9 StateMachines	25
10.10 ObjTypes	27
11 Use of Equipment Metadata for Data Acquisition	29
11.2 Identifying Information Sources	29
11.3 Identifying Information Items	29
11.4 Parameter Values	30
12 Requirements for Compliance	30

1 Purpose

1.1 This specification describes a method for allowing equipment suppliers to provide a description of the variables, events, exceptions, and physical equipment configuration available from their equipment. With this information available for consumption by software systems, it can be used as a tool to aid the process of integrating equipment into a factory's automation system.

2 Scope

2.1 In-scope

2.1.1 This document specifies the classes that suppliers are to use to describe essential data, events, and exceptions provided by their equipment. The specification only describes information that is static in nature (that is, information that does not change dynamically while the equipment is running). This document also specifies an interface that clients can use to access this information.

2.1.2 This specification applies to all semiconductor manufacturing equipment that supports the data acquisition interface defined in the SEMI specification for Data Collection Management.

2.2 Out of Scope

2.2.1 This specification does not define any new behavior required of the equipment other than that necessary for retrieving information describing equipment configuration, interfaces, and available data, and keeping this information current.

2.2.2 The details of any underlying concepts and behavioral models (for example, carrier management, process/control job, etc.) that can be described by metadata are to be separately specified in a document dedicated to those concepts. Only the ability to describe the fact that a supplier has implemented such a concept and that a client can discover this implementation and any data it can produce is in scope for this specification.

2.2.3 This specification does not require that the metadata provided by the equipment be directly human readable. It is expected that applications will be written to organize and present this information to human users in a form that is easier for end users to digest.

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.

3 Limitations

3.1 Provisional Specification

3.1.1 This is a provisional specification. In order to have the provisional status removed, the following items must be completed:

- Analysis of requirements for describing material, processing, equipment, or other contextual information for use with data collection capabilities. Results of this analysis may or may not result in new metadata classes.
- Definition of mechanism for version management of metadata (for example, mechanisms for clients to detect changes made to equipment metadata).
- Definition of mechanism for describing SEMI standardized events that have no corresponding standardized state model.
- Creation of Related Information to document the representation of SEMI standard state models using the StateMachine classes in Section 10.9.
- Further refinement of the model for describing units, to accommodate conversion factors and/or other conventions that will aid the ability of software to understand the use of units in equipment metadata.

3.2 Abstract Model

3.2.1 This specification is an abstract model only. Adjunct specifications must be developed to bind this specification to an implementation technology.

4 Referenced Standards

4.1 SEMI Standards

SEMI E39 — Object Services Standard: Concept, Behavior, and Services

SEMI E120 — Provisional Specification for the Common Equipment Model (CEM)

4.2 ISO Standards¹

ISO 639:1988 — Code for the representation of names of languages

ISO 3166-1:1997 — Codes for the representation of names of countries and their subdivisions – Part 1: Country codes

¹ International Organization for Standardization

ISO Central Secretariat, 1, rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland. Telephone: 41.22.749.01.11; Fax: 41.22.733.34.30

Website: /www.iso.ch

4.3 Other Standards

American National Standards Institute (ANSI) X3.135-1992 – Information Systems – Database Language – SQL (<http://www.ansi.org>)²

Internet Engineering Task Force (IETF) RFC 1776 – Tags for the Identification of Languages, ed. H. Alvestrand. 1995 (<http://www.ietf.org/rfc/rfc1766.txt>)

Unified Modeling Language (UML) Specification, Version 1.4, OMG Specification 01-09-67, available from http://www.omg.org/technology/documents/modeling_spec_catalog.htm

Uniform Resource Name (URN) Syntax, IETF RFC 2141, May 1997 (<http://www.ietf.org/rfc/rfc2141.txt>)

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

5 Terminology

5.1 Abbreviations and Acronyms

5.1.1 Descriptions of many of the abbreviations and acronyms used in this specification may be found in the SEMI Compilation of Terms, available on the SEMI web site, <http://www.semi.org/>. In most cases, these terms are not included in this section.

5.1.2 *IDL* — Interface Definition Language

5.1.3 *OMG* — Object Management Group

5.1.4 *SQL* — Standard Query Language

5.1.5 *UML* — Unified Modeling Language

5.1.6 *URN* — Uniform Resource Name

5.1.7 *XML* — eXtensible Markup Language

5.2 Definitions

5.2.1 Definitions or descriptions of many of the terms used in this specification may be found in the SEMI Compilation of Terms, available on the SEMI web site, <http://www.semi.org/>. In most cases, these terms are not included in this section.

5.2.2 Related Information 8 contains useful definitions of UML terms taken directly from the UML standard. UML terms used in this document conform to these definitions. Please refer to this section as needed.

5.2.3 *metadata* — data used to describe data. For example, if a tool can report an event with several associated variables under certain conditions, the metadata for that event would provide a description of what condition will produce the event, what the type and units are of each variable, and the id of the event itself.

6 Conventions

6.1 Document and Notation Conventions

6.1.1 *Unified Modeling Language (UML)* — This specification uses the UML conventions for representing all class, state, object, and sequence diagrams. Refer to the UML specification for a detailed description of these conventions.

6.1.2 *Attribute Tables* — The table below provides an example of the tables used to list and describe attributes of classes defined in this specification.

² American National Standards Institute

Headquarters: 1819 L Street, NW, Washington, DC 20036, USA.
Telephone: 202.293.8020; Fax: 202.293.9287

New York Office: 11 West 42nd Street, New York, NY 10036, USA.
Telephone: 212.642.4900; Fax: 212.398.0023

Website: www.ansi.org

Table 1 Attribute Table Format

<i>Attribute Name</i>	<i>Definition</i>	<i>Required</i>	<i>Form</i>
		Y or N	See list below

6.1.2.1 *Required* — Specifies whether or not an attribute must be present for the described class. Y – Yes, or N – No.

6.1.2.2 *Form* — Defines the data type of the attribute. The terms used to describe data types in this column are defined in the SEMI Compilation of Terms, or are included as part of the specification. Refer to the compilation of terms for the definition of SEMI type name meanings.

6.1.3 *UML Associations* — The mechanism used for representing UML associations 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 the UML associations shown in this document.

6.1.4 *Association Tables* — The table below provides an example of the tables used to list and describe associations between classes defined in this specification.

Table 2 Association Table Format

<i>Association Role Name</i>	<i>Definition</i>	<i>Comments</i>

6.1.4.1 *Association Role Name* — The name of the association role being specified.

6.1.4.2 *Definition* — Describes the function or purpose of the association.

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

6.1.5 *Operation Definition Tables* — the table below provides an example of the tables used to list and describe the interface operations defined in this specification.

Table 3 Operation Definition

<i>Operation</i>	<i>Description</i>	<i>Type</i>
		See list below

6.1.5.1 *Operation* — Specifies the name of the operation.

6.1.5.2 *Type* — Specifies the messaging semantics of the operation. Only Request-Reply (RR) semantics are used in this specification. Request-Reply messages are messages that involve an initiator and a receiver. In a Request-Reply exchange, the initiator sends a single request message to the receiver, and the receiver sends a single reply message to that request back to the initiator.

6.1.6 *Operation Argument Definition Table* — The table below provides an example of the tables used to list and describe arguments for interface operations defined in this specification.

Table 4 Operation Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
		See list below	See list below

6.1.6.1 *Argument* — Specifies the name of the argument.

6.1.6.2 *Kind* — Specifies whether the argument is an ‘in’, ‘out’, or ‘error’ argument for the operation. ‘Error’ arguments always function as ‘out’ arguments, but indicate that the operation did not complete successfully.

6.1.6.3 *Form* — Defines the data type of the argument. The terms used to describe data types in this column are defined in the SEMI Compilation of Terms, or are included as part of the specification. Refer to the compilation of terms for the definition of SEMI type name meanings.

7 Background

7.1 Motivation

7.1.1 This specification is designed to help address the problem of integrating semiconductor equipment from a variety of suppliers into an automated factory environment, and to provide a standard mechanism for describing information and data that the equipment can provide for a wide variety of uses. See Related Information 1 for an overview of some example use cases.

7.1.2 This specification defines the information that is needed to communicate essential features of the equipment to software applications, and provides extensibility features to allow suppliers to communicate information unique to their equipment or that may not be specified in a SEMI standard.

7.1.3 Equipment metadata is designed to be useful to applications that provide an interface to human end users for the purposes of understanding and browsing equipment configuration, available data items, events, exceptions, etc. In many cases, these users will be referring to equipment metadata so they can perform data collection for a variety of objectives (diagnostics, troubleshooting, process control, etc.).

7.1.4 Equipment metadata is also designed to be useful to applications written to take advantage of standardized state models and other data so that discovery and usage of this information can be automated, without requiring human intervention. Diagnostics, health monitoring, or other supplier-provided applications that understand the details of specific equipment types, for example, can also discover and use common equipment-specific configuration and data without requiring manual assistance.

8 Overview

8.1 Equipment Metadata Organization

8.1.1 Figure 1 shows the concepts that are included in this specification, organized as UML classes, and the dependencies between them. Not all classes in the specification are shown, for clarity. Classes that are grayed out belong to the SEMI E120 Common Equipment Model, and are not specified in this document. Each class that is not grayed out represents a primary concept included this specification. A brief description of the purpose of each class is provided in this section only as an overview. For the detailed specification of each class, refer to Section 10.

8.1.2 Dotted lines between classes indicate a dependency between the classes. The direction of the dependency is indicated by the direction of the arrow in each dependency. Dependencies point from a “dependent” class to a “target” class. In the diagram, a “defines” dependency between two classes shows that the dependent class uses the target class to define unique reference-able metadata items (for example, a specific parameter). A “refers to” dependency shows that the dependent class refers to unique metadata items that are defined using the target class.

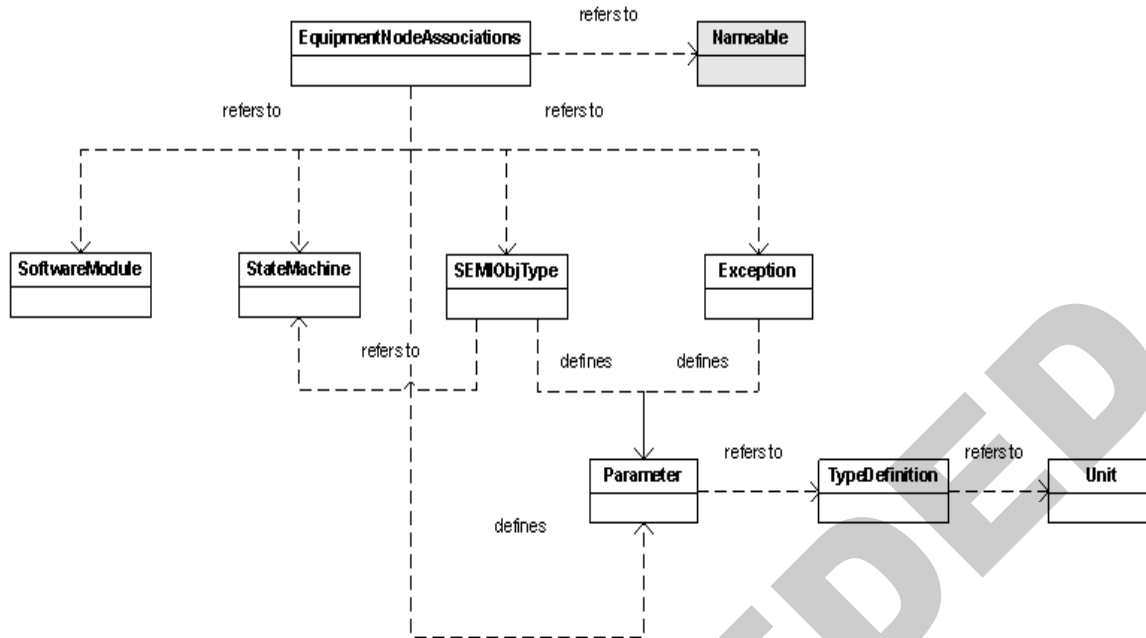


Figure 1
Organization of Metadata Concepts

8.1.3 EquipmentNodeAssociations

8.1.3.1 This class, together with other related classes, describes the relationships between specific elements from the physical equipment structure and exceptions, equipment system software modules, SEMI ObjTypes, and events originating from implemented state models. This class also defines the Parameters that are provided by a specific equipment element. For example, if a load port on a specific tool implements the SEMI E87 Load Port Transfer state model, this class would define an association between that load port and the SEMI E87 state model as well as the corresponding events and parameters that can be reported by the load port. This class and other related classes are described in Section 10.3 on page 9.

8.1.4 Parameter

8.1.4.1 This class, together with other related classes, describes named, typed variables and their purpose or function. Parameters can be used to model any concept that can be represented as a data type. For example, the SEMI E30 concepts of equipment constants and status variables can both be represented by a parameter. Any type of data that the equipment can produce (for example, measurement results on metrology equipment) can be represented by a parameter. Parameters can only be defined by an equipment node, a SEMI ObjType, or an exception. This class and other related classes are described in Section 10.4 on page 13.

8.1.5 TypeDefinition

8.1.5.1 This class, together with other related classes, is used to describe the data types used on the equipment to represent parameters. For example, if the equipment utilizes several parameters that have a structured type consisting of an integer, a string, and an array of Booleans, this class would describe the structure of that type. For numeric types, the metadata can describe the units that a value of that type would represent. This class and other related classes are described in Section 10.5 on page 16.

8.1.6 Unit

8.1.6.1 This class is used to describe the system of units that is used by the various parameter types. For example, if some parameter represents a capacitance in microfarads, the Unit class would describe how the microfarad unit is represented so the parameter of that type can indicate what its units are. This class is described in Section 10.6 on page 23.

8.1.7 Nameable

8.1.7.1 This is the base class used by the SEMI E120 Common Equipment Model to describe the physical organization of the equipment. Each physical equipment component is described by a class (Link, Equipment, Module, Subsystem, MaterialLocation, and IODevice) that derives from Nameable. Please see that specification for a complete description of these classes.

Additional description is provided in Section 10.2 on page 9.

8.1.8 SoftwareModule

8.1.8.1 This class is used to describe the existence and version of equipment system software that is in use on the tool. For example, this could be used to describe the version of the control system software modules for the equipment or those of important subsystems. It is common when troubleshooting equipment problems to need to understand what software versions are currently in use on the equipment. This can influence the behavior of the equipment, as well as its ability to interface to factory systems, so it is often the first line of inquiry in determining where the source of a problem may be. This class is described in Section 10.8 on page 25.

8.1.9 StateMachine

8.1.9.1 This class, together with other related classes, describes the state machines (and the corresponding transitions and events) that are implemented by the equipment. This class may be used to describe any SEMI-defined state machine or any supplier-defined state machine. This class and other related classes are described in Section 10.9 on page 25.

8.1.10 SEMIObjType

8.1.10.1 This class, together with other related classes, is used to describe implemented SEMI ObjTypes from any SEMI standard in which an ObjType is defined. An ObjType provides a state machine described by the StateMachine class, and has two or more attributes associated with the ObjType, in accordance with the SEMI definition of the ObjType's attributes. This class and other related classes are described in Section 10.10 on page 27.

8.1.11 Exception

8.1.11.1 This class describes the existence of error conditions that the equipment can detect and report. An exception can represent any SEMI E30 alarm, SEMI E41 exception, or equipment-supplier-defined exception that can be reported to clients. This class is described in Section 10.7 on page 24.

8.2 Accessing Equipment Metadata

8.2.1 Equipment metadata is directly associated with each unique equipment installation within a factory. Access to metadata is provided by a collection of operations available from a specific equipment installation. This interface is described in Section 9.1 on page 7.

9 Interface for Requesting Equipment Metadata

9.1 Equipment Metadata Manager

9.1.1 Figure 2 shows the interface used to access equipment metadata descriptions.



Figure 2
Interface for Accessing Equipment Metadata

9.1.1.1 EquipmentMetadataManager Operations

Table 5 EquipmentMetadataManager Operation Definition

Operation	Description	Type
GetUnits	Retrieves all unit metadata provided by the equipment.	RR
GetTypes	Retrieves all type definitions provided by the equipment.	RR
GetSoftwareModules	Retrieves all software module metadata provided by the equipment.	RR
GetStateMachines	Retrieves all state machine metadata provided by the equipment.	RR
GetSEMIObjTypes	Retrieves all SEMI ObjTypes metadata provided by the equipment.	RR
GetExceptions	Retrieves all exception metadata provided by the equipment.	RR
GetEquipmentStructure	Retrieves all equipment structural metadata provided by the equipment.	RR
GetEquipmentAssociations	Retrieves all equipment node association metadata provided by the equipment.	RR

9.1.1.2 *GetUnits* — Upon receiving this request, the equipment shall return all units metadata provided by the equipment.

9.1.1.2.1 *GetUnits Operation Arguments*

Table 6 GetUnits Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
units	All units defined for the equipment.	out	Unordered list of structured data, of type Unit, described in Section 10.6.

9.1.1.3 *GetTypes* — Upon receiving this request, the equipment shall return all type definition metadata provided by the equipment.

9.1.1.3.1 *GetTypes Operation Arguments*

Table 7 GetTypes Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
types	All type definitions provided by the equipment.	out	Unordered list of structured data, of type TypeDefinition, described in Section 10.5.

9.1.1.4 *GetSoftwareModules* — Upon receiving this request, the equipment shall return all software module metadata provided for the equipment.

9.1.1.4.1 *GetSoftwareModules Operation Arguments*

Table 8 GetSoftwareModules Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
softwareModules	All software module metadata provided by the equipment.	out	Unordered list of structured data, of type SoftwareModule, described in Section 10.8.

9.1.1.5 *GetStateMachines* — Upon receiving this request, the equipment shall return all state machine metadata provided by the equipment.

9.1.1.5.1 *GetStateMachines Operation Arguments*

Table 9 GetStateMachines Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
stateMachines	All state machine metadata provided by the equipment.	out	Unordered list of structured data, of type StateMachine, described in Section 10.9.

9.1.1.6 *GetSEMIObjTypes* — Upon receiving this request, the equipment shall return all OSS ObjType metadata provided by the equipment.

9.1.1.6.1 *GetSEMIObjTypes Operation Arguments*

Table 10 GetSEMIObjTypes Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
objTypes	All SEMI ObjType metadata provided by the equipment.	out	Unordered list of structured data, of type SEMIObjType, described in Section 10.10 . If no SEMI ObjTypes are provided, this list shall be empty.

9.1.1.7 *GetExceptions* — Upon receiving this request, the equipment shall return all exception metadata provided the equipment.

9.1.1.7.1 *GetExceptions Operation Arguments*

Table 11 GetExceptions Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
exceptions	All exception metadata provided by the equipment.	out	Unordered list of structured data, of type Exception, described in Section 10.7.

9.1.1.8 *GetEquipmentStructure* — Upon receiving this request, the equipment shall return all equipment structural metadata provided by the equipment.

9.1.1.8.1 *GetEquipmentStructure Operation Arguments*

Table 12 GetEquipmentStructure Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
equipmentStructure	All equipment structural metadata provided by the equipment.	out	Structured data, of type Link or Equipment, as defined by the SEMI Common Equipment Model.

9.1.1.9 *GetEquipmentAssociations* — Upon receiving this request, the equipment shall return all equipment node association metadata defined for the equipment.

9.1.1.9.1 *GetEquipmentAssociations Operation Arguments*

Table 13 GetEquipmentAssociations Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
equipmentAssociations	All equipment association metadata provided by the equipment.	out	Unordered list of structured data, of type EquipmentNodeAssociations, as described in Section 10.3.

10 Equipment Metadata

10.1 Management of Equipment Metadata

10.1.1 When equipment configuration changes result in a change to any information described using the classes in Section 10 of this document or by the SEMI Common Equipment Model, the supplier shall update the metadata description accordingly at the time of the change. This includes reconfiguration of the equipment after its introduction into the factory. Ensuring that all aspects of equipment metadata are kept up-to-date and available to the factory is a critical aspect of maintaining equipment metadata.

10.2 Equipment Structure

10.2.1 The physical equipment structure shall be described according to the SEMI Common Equipment Model. Suppliers shall provide a model of those elements of their equipment for which it is necessary and practical to facilitate (at least) utilization tracking, process control, and technical support objectives.

10.3 Associations

10.3.1 Equipment Node Associations

10.3.1.1 Figure 3 shows how to describe the associations between an element (or node) of the equipment hierarchy described by the equipment physical structure metadata, and software modules, parameters, implemented statecharts, objTypes, and exceptions. Providing these descriptions indicates that the specified node acts as the source of all state machine events, exceptions, objTypes, and parameters that are included in the description. All parameters, exceptions, objTypes, and events associated with a given equipment node are independent of the events, exceptions, objTypes, and parameters associated with other equipment nodes.

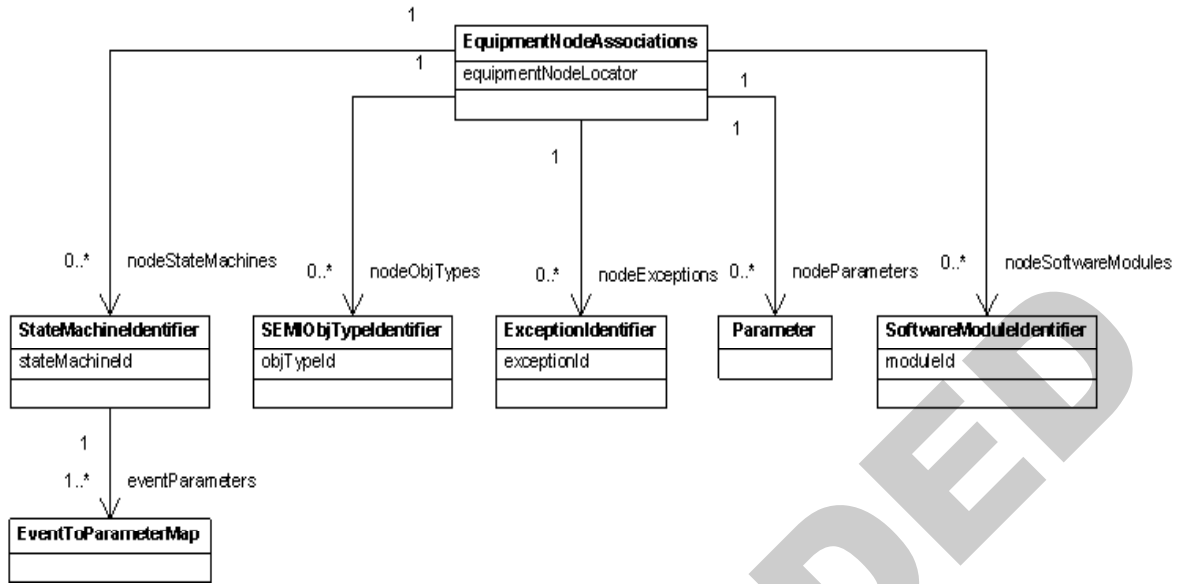


Figure 3
Describing Equipment Node Associations

10.3.1.2 *EquipmentNodeAssociations* — This class is used to identify a specific element of the equipment hierarchy. For the specified element, this class identifies the software modules, parameters, exceptions, objTypes, and events for which this node is the source. This is achieved through the StateMachineIdentifier, SEMIObjTypeIdentifier, SoftwareIdentifier, ExceptionIdentifier, and Parameter types. The EventToParameterMap class is described in Section 10.3.1.15. The Parameter class is described in Section 10.4. Only one EquipmentNodeAssociation description can be provided for a given equipment element. Equipment elements may be at the Link, Equipment, Module, Subsystem, MaterialLocation, or IODevice level, as described by the SEMI E120 standard, Common Equipment Model.

10.3.1.3 *EquipmentNodeAssociations* Attribute Definition Table

Table 14 *EquipmentNodeAssociations* Attribute Definition

Attribute Name	Definition	Required	Form
equipmentNodeLocator	Identifies a specific equipment element.	Y	Text, formatted according to the 'locator' (or 'alias') field of the Nameable class in the SEMI Common Equipment Model specification. Must be equal to a valid locator from the equipment physical structure metadata.

10.3.1.4 *EquipmentNodeAssociations* Association Definition Table

Table 15 *EquipmentNodeAssociations* Association Definition

Association Role Name	Definition	Comments
nodeStateMachines	An unordered list of state machines implemented by this equipment node.	List of elements of type StateMachineIdentifier.
nodeObjTypes	An unordered list of the objTypes that are associated with this equipment node.	List of elements of type SEMIObjTypeIdentifier.
nodeExceptions	An unordered list of exceptions that can be reported by this equipment node.	List of elements of type ExceptionIdentifier.

Association Role Name	Definition	Comments
nodeParameters	An unordered list of typed parameters provided by this equipment node.	List of elements of type Parameter, described in Section 10.4. Any parameters that represent variables from SEMI standards shall be typed and named according to Sections 10.5.7 and 10.4.11. Each parameter name for this node shall be unique. Parameter names need not be unique across equipment nodes.
nodeSoftwareModules	An unordered list of software modules that are related to this equipment node.	List of elements of type SoftwareIdentifier.

10.3.1.5 *StateMachineIdentifier* — This class can be used to identify a specific state machine that is implemented by this equipment node.

10.3.1.6 *StateMachineIdentifier Attribute Definition Table*

Table 16 StateMachineIdentifier Attribute Definition

Attribute Name	Definition	Required	Form
stateMachineId	Identifies a state machine.	Y	Text, must be equal to the 'id' attribute of a top-level StateMachine (see Section 10.9 described elsewhere in the metadata.

10.3.1.7 *StateMachineIdentifier Association Definition Table*

Table 17 StateMachineIdentifier Association Definition

Association Role Name	Definition	Comments
eventParameters	Unordered list of events and associated parameters.	List of elements of type EventToParameterMap, described in Section 10.3.1.15.

10.3.1.8 *SEMIObjTypeIdentifier* — This class can be used to identify a specific SEMI OSS ObjType that is provided by this equipment node.

10.3.1.9 *SEMIObjTypeIdentifier Attribute Definition Table*

Table 18 SEMIObjTypeIdentifier Attribute Definition

Attribute Name	Definition	Required	Form
objTypeId	Identifies an ObjType supported by the equipment.	Y	Text, must be equal to the 'id' attribute of a SEMIObjType (see Section 10.10) defined elsewhere in the metadata.

10.3.1.10 *ExceptionIdentifier* — This class can be used to describe the exceptions that can be reported by the identified equipment node.

10.3.1.11 *ExceptionIdentifier Attribute Definition Table*

Table 19 ExceptionIdentifier Attribute Definition

Attribute Name	Definition	Required	Form
exceptionId	Exception identifier.	Y	Text, must be equal to the 'id' attribute of an Exception (see Section 10.7) defined elsewhere in the metadata.

10.3.1.12 *SoftwareModuleIdentifier* — This class can be used to identify a specific software module that is in use by, or associated with the operation of, this equipment node.

10.3.1.13 *SoftwareModuleIdentifier* Attribute Definition Table

Table 20 *SoftwareModuleIdentifier* Attribute Definition

Attribute Name	Definition	Required	Form
moduleId	Identifier of a software module.	Y	Text, must be equal to the 'id' attribute of a SoftwareModule (see Section 10.8) defined elsewhere in the metadata.

10.3.1.14 Figure 4 shows how to describe the parameters that are available for reporting when a specific event occurs.

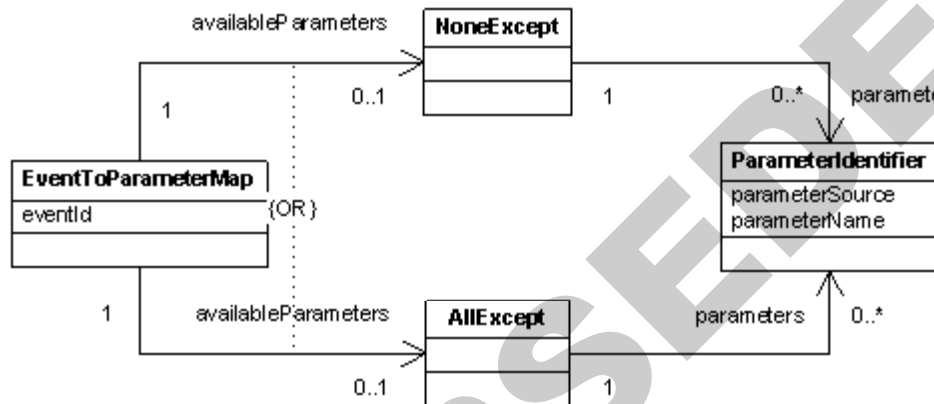


Figure 4
Describing Parameters Available for Events

10.3.1.15 *EventToParameterMap* — This class is used to provide a description of the parameters available for a specific event from a given source. It can provide this description through either the *NoneExcept* or *AllExcept* classes.

10.3.1.16 *EventToParameterMap* Attribute Definition Table

Table 21 *EventToParameterMap* Attribute Definition

Attribute Name	Definition	Required	Form
eventId	Event identifier.	Y	Text, must be equal to the 'eventId' attribute of a Transition (see Section 10.9.4) defined elsewhere in the metadata.
availableParameters	Describes which parameters are restricted or available for reporting with this event.	Y	Structured data of type <i>NoneExcept</i> or <i>AllExcept</i> .

10.3.1.17 *EventToParameterMap* Association Definition Table

Table 22 *EventToParameterMap* Association Definition

Association Role Name	Definition	Comments
availableParameters	Describes which parameters are restricted or available for reporting with this event.	Structured data of type <i>NoneExcept</i> or <i>AllExcept</i> .

10.3.1.18 *NoneExcept* — This class can be used to describe the parameters that can be reported for this event.

10.3.1.19 *NoneExcept Association Definition Table*

Table 23 NoneExcept Association Definition

Association Role Name	Definition	Comments
parameters	Unordered list of parameters that can be reported with this event.	List of 0 or more elements of type <i>ParameterIdentifier</i> . If no elements are present, this indicates that no parameters can be reported with this event. Parameters not included in this list cannot be reported with this event.

10.3.1.20 *AllExcept* — This class can be used to describe the parameters that cannot be reported for this event.

10.3.1.21 *AllExcept Association Definition Table*

Table 24 AllExcept Association Definition

Association Role Name	Definition	Comments
parameters	Unordered list of parameters that cannot be reported with this event.	List of 0 or more elements of type <i>ParameterIdentifier</i> . If no elements are present, this indicates that all parameters can be reported with this event. All parameters not included in this list can be reported with this event.

10.3.1.22 *ParameterIdentifier* — This class is used to identify any parameter associated with any equipment node.

10.3.1.23 *ParameterIdentifier Attribute Definition Table*

Table 25 ParameterIdentifier Attribute Definition

Attribute Name	Definition	Required	Form
parameterSource	Identifier for the source of a parameter.	Y	Text, must be either empty, or formatted according to Section 11.2. If empty, the current source is implied.
parameterName	Name of a parameter defined for the specified equipment node.	Y	Text, must be equal to a valid parameter name owned by the source identified by 'parameterSource'.

10.4 Parameters

10.4.1 Parameters can represent any concept that can be described by any of the primitive or composite types described in Section 10.5. A parameter can generally be thought of as a named, typed field that has a specific meaning, utility, or other information value. Examples are process control variables, equipment configuration settings, measurement results, wafer maps, data variables, etc. Parameters can only be defined by an equipment node (Section 10.3.1), a SEMI *ObjType* (Section 10.10), or an exception (Section 10.7).

10.4.2 Figure 5 shows how to describe named, typed parameters with units, constraints, and associations with other parameters. Type and units information is contained in the *TypeDefinition* referred to by the *DefinedType* class. *TypeDefinition* is described in Section 10.5.1.3.

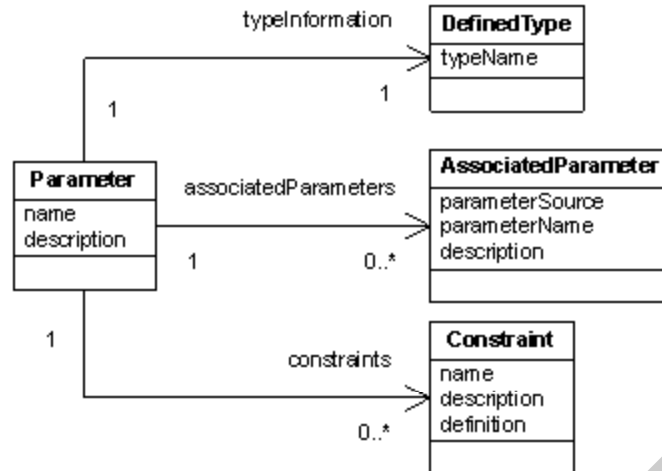


Figure 5
Describing Typed Parameters

10.4.3 Suppliers shall provide a model of those Parameters for which it is necessary and practical to facilitate (at least) utilization tracking, process control, and technical support objectives.

10.4.4 Parameter Attribute Definition Table

Table 26 Parameter Attribute Definition

Attribute Name	Definition	Required	Form
name	The name of the parameter.	Y	Text.
description	A human-readable description of the purpose or usage of the Parameter.	Y	Text.

10.4.5 Parameter Association Definition Table

Table 27 Parameter Association Definition

Association Role Name	Definition	Comments
constraints	A description of any restrictions associated with this parameter.	List of structured data, of type Constraint, described in Section 10.4.9. Logical AND semantics apply between constraints if more than one is described.
associatedParameters	List of other parameters in the system that are related to this parameter.	List of structured data, of type AssociatedParameter, described in Section 10.4.8.
typeInformation	Refers to a TypeDefinition that describes the type and units of this parameter.	One and only one element of type DefinedType.

10.4.6 *DefinedType* — Identifies a TypeDefinition (see Section 10.5.1.3) described elsewhere in the metadata.

10.4.6.1 DefinedType Attribute Definition Table

Table 28 DefinedType Attribute Definition

Attribute Name	Definition	Required	Form
typeName	The name of the defined type for this field.	Y	Text, must be equal to the “name” attribute of a TypeDefinition (see Section 10.5.1.3) described elsewhere in the metadata.

10.4.7 Each parameter can provide a list of zero or more associated parameters, and a description of what the relationship is. For example, the relationship between a temperature set point parameter and a parameter representing the actual temperature value. Or, the relationship between a time-varying parameter (for example, current, magnetic field, etc.) and a configuration parameter representing the maximum possible sampling rate for that variable.

10.4.8 *AssociatedParameter* — This class can be used to identify a specific parameter anywhere in the equipment hierarchy. If identified, this indicates that the Parameter being defined is related in some way to the parameter identified by this class. The meaning of the relationship is documented in the description of the association.

10.4.8.1 *AssociatedParameter Attribute Definition Table*

Table 29 AssociatedParameter Attribute Definition

Attribute Name	Definition	Required	Form
parameterSource	Identifier of a valid parameter source.	Y	Text, must be either empty, or formatted according to Section 11.2. If empty, the current source is implied.
parameterName	Name of a parameter defined for the specified source.	Y	Text, must be equal to a valid parameter name defined by the source identified by 'parameterSource'.
description	Description of the relationship between the defined parameter and the parameter identified by this association.	Y	Text.

10.4.9 *Constraint* — Represents a constraint associated with this parameter. A constraint can be used to describe limitations on the valid values of a parameter, the rate at which the parameter can be collected for off-tool reporting, or any other similar restriction on the value or usage of the parameter.

10.4.9.1 *Constraint Attribute Definition Table*

Table 30 Constraint Attribute Definition

Attribute Name	Definition	Required	Form
name	The name of the constraint.	Y	Text.
description	A description of the purpose or meaning of the constraint.	Y	Text.
definition	The definition of the constraint.	Y	Text, formatted according to Section 10.4.10.

10.4.10 *Constraint Definition*

10.4.10.1 Parameter constraints can describe restrictions on the valid values of parameters. A constraint may be documented as a propositional formula, which evaluates TRUE with valid parameter values. Relational algebra defines a rich syntax for propositional formulas; however, this syntax is not commonly used for commercial applications. Standard Query Language (SQL) is a well-documented commercial application of relational algebra well suited to this application. Specifically, the syntax of the **WHERE** clause (of the SQL **SELECT** statement) is suitable for describing constraints on parameter values. The **WHERE** clause of the SQL **SELECT** statement is a propositional-expression that consists of one or more comparisons of values with constants. It has the following form.

WHERE propositional-expression [nested-propositional-expression];

(The syntax of the SQL **SELECT** statement **WHERE** is discussed in Related Information, Section R2-1).

10.4.10.2 For example, a parameter named “TemperatureSetpoint” that represents a temperature set point that may not be less than 100 or greater than 200 may be described using the following SQL **WHERE** clause.

WHERE TemperatureSetpoint **BETWEEN** (100 **AND** 200);