```
BAKE;2.3     BakeTime                          undefined for unlinked recipe
```

R1-6 Applications of Object Services

Object Services are defined in SEMI E39 (Object Services Standard (OSS): Concepts, Behavior, and Services). This section provides a description of how these services may be applied for RMS.

R1-6.1 *Scope* — Scope may be used to specify the *namespace* of interest. It may also be used to specify a *namespace* or *recipe execution resource* belonging to a *component agent* of the service provider. For example, scope allows a factory host to ask a cluster tool for the attributes of a cluster module.

Scope may be used to point to one of several namespaces supported by an agent.

R1-6.2 *Filter* — A filter may be used when asking a *namespace* for a list of recipe *identifiers* to limit the number of *identifiers* returned. For example, the service user may ask a *namespace* for a list of recipe *identifiers* that were edited by "Tom" (Attribute name = "EditedBy", Attribute value = "Tom", Attribute Relation = "is equal to") and that are *linked* (Attribute name = "Linked", Attribute value = "TRUE", Attribute Relation = "is equal to").

R1-6.3 *Complex Attributes* — Object Services do not provide a way to change individual elements in an attribute with a structure or list form. To add, delete, or modify items in a list or structure, it is necessary to first get the current attribute value for the object of interest using the GetAttr service, to make the desired changes externally to the value, and then to write the new value using the SetAttr service.

R1-7 Examples of RMS Application

Figure R1-2 illustrates possible interactions between a *namespace* and *namespace manager* implemented on a factory system, a diskless equipment with a *recipe executor*, and an *operator*.

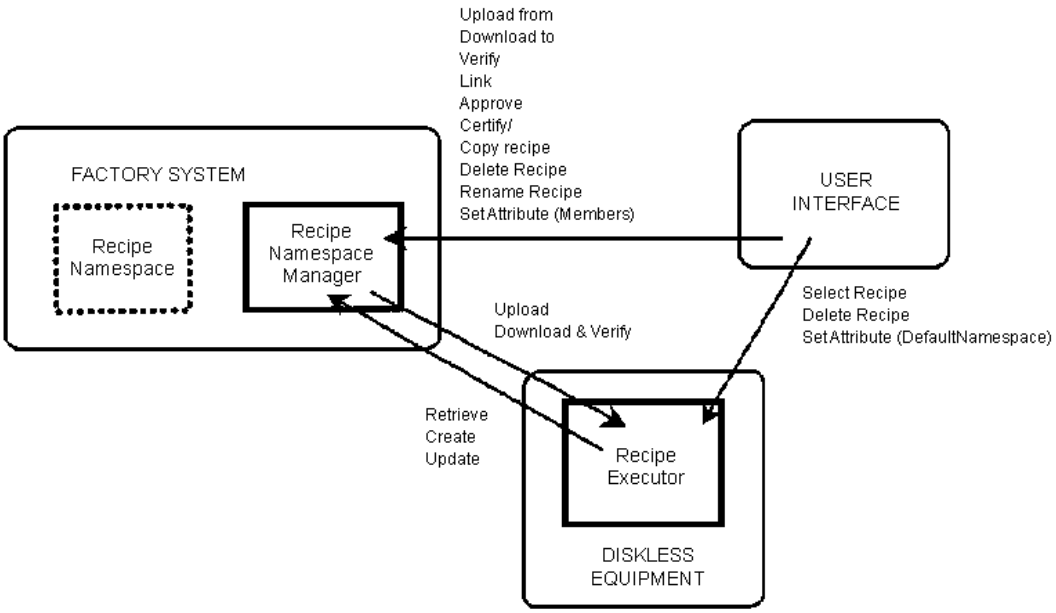

Figure R1-2
Example of RMS Applications

Other examples would be similar. Interactions are primarily with external service users of *namespace* and *recipe execution* capabilities, and between *namespace managers* and *recipe executors*.

Namespaces do not interact with one another. To copy a recipe from one *namespace* to another, an intermediary is required. This is not covered by RMS.

NOTICE: These standards do not purport to address safety issues, if any, associated with their use. It is the responsibility of the user of these standards to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use. SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E42.1-0704
STANDARD FOR SECS-II PROTOCOL FOR RECIPE MANAGEMENT STANDARD (RMS)

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the Japanese Information & Control Committee. Current edition approved by the Japanese Regional Standards Committee on April 30, 2004. Initially available at www.semi.org June 2004; to be published July 2004. Originally published in 1996; previously published September 1997.

1 Purpose

1.1 This document maps the services and data of the parent document to SECS-II streams and functions and data definitions.

2 Scope

2.1 This document applies to all implementations of Recipe Management that use the SECS-II message protocol (SEMI E5).

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 Referenced Standards

3.1 *SEMI Standards*

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

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

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

4 Terminology

None.

5 Service Mapping

5.1 Table 1 shows the specific SECS-II streams and functions that shall be used for SECS-II implementations of the services defined in RMS.

5.2 Request and notification messages are mapped to primary (odd-numbered) SECS-II functions and response messages are mapped to secondary (even-numbered) SECS-II functions.

5.3 In several cases, a common set of parameters allows more than one RMS services to be mapped to a single stream and function, with an additional SECS-II data item used to differentiate between the two services. RMNCreateNS and RMNDeleteNS both map to S15,F3/F4, RMNCreate and RMNUpdate both map to S15,F13/14, and RMNCopy and RMNRename both map to S15,F19/20.

5.4 In addition, the notification messages RMNComplete, RMEComplete, RMEChange, RMDComplete, and RMDNotify all map to S6,F25/F26.

Table 1 Services Mapping Table

Message Name	*Stream, Function*	*SECS-II Name*
RMNCreateNS	S15,F3/F4	Recipe Namespace Action Request/Acknowledge
RMNDeleteNS	S15,F3/F4	Recipe Namespace Action Request/Acknowledge
RMNRenameNS	S15,F5/F6	Recipe Namespace Rename Request/Acknowledge
RMNSpaceInquire	S15,F7/F8	Recipe Space Request/Data
RMNRcpStatInquire	S15,F9/F10	Recipe Status Request/Data
RMNVersionInquire	S15,F11/F12	Recipe Version Request/Data
RMNCreate	S15,F13/F14	Recipe Create Request/Acknowledge
RMNUpdate	S15,F13/F14	Recipe Create Request/Acknowledge
RMNStore	S15,F15/F16	Recipe Store Request/Acknowledge
RMNRetrieve	S15,F17/F18	Recipe Retrieve Request/Data
RMNCopy	S15,F19/F20	Recipe Rename Request/Acknowledge
RMNRename	S15,F19/F20	Recipe Rename Request/Acknowledge

Message Name	*Stream, Function*	*SECS-II Name*
RMNAction	S15,F21/F22	Recipe Action Request/Acknowledge
RMNGetDescriptor	S15,F23/F24	Recipe Descriptor Request/Data
RMNVarPar	S15,F25/F26	Recipe Parameter Update Request/Acknowledge
RMNComplete	S6,F25	Notification Report Send
RMEDnldVer	S15,F27/F28	Recipe Download
RMEVerify	S15,F29/F30	Recipe Verify
RMEUpload	S15,F31/F32	Recipe Upload
RMESpaceInquire	S15,F7/F8	Recipe Space Request/Data
RMERename	S15,F19/F20	Recipe Rename Request/Acknowledge
RMEDelete	S15,F35/F36	Recipe Delete Request/Acknowledge
RMESelect	S15,F33/F34	Recipe Select Request/Acknowledge
RMEDeselect	S15,F35/F36	Recipe Deselect Request/Acknowledge
RMEGetDescriptor	S15,F23/F24	Recipe Descriptor Request/Data
RMNAction	S15,F21/F22	Recipe Action Request/Acknowledge
RMEChange	S6,F25	Notification Report Send
RMEComplete	S6,F25	Notification Report Send
RMDSApproveAction	S15,F37/F38	DRNS Segment Approve Action Request/Acknowledge
RMDRAdd SegRecord	S15,F39/F40	DRNS Recorder Segment Request/Acknowledge
RMDRDelSegRecord	S15,F39/F40	DRNS Recorder Segment Request/Acknowledge
RMDRAddChgRecord	S15,F41/F42	DRNS Recorder Modify Request/Acknowledge
RMDRDelChgRecord	S15,F41/F42	DRNS Recorder Modify Request/Acknowledge
RMDRGetChgRecord	S15,F43/F44	DRNS Get Change Request/Data
RMDGetChgRequest	S15,F43/F44	DRNS Get Change Request/Data
RMDSegChange	S15,F45/F46	DRNS Manager Segment Change Approval Request/Grant
RMDRebuild	S15,F47/F48	DRNS Manager Rebuild Request/Acknowledge
RMDComplete	S6,F25/F26	Notification Report Send
RMDNotify	S6,F25/F26	Notification Report Send
Attach	S14F13,F14	Object Attach Request/Acknowledge (SEMI E39.1)
AttachSetAttr	S14F15,F16	Attached Object Action request/Acknowledge (SEMI E39.1)
AttachSupervisedObject	S14F17,F18	Supervised Object Action Request/Acknowledge (SEMI E39.1)
Create	S14F9,F10	Create Object Request/Acknowledge (SEMI E39.1)
Delete	S14F11,F12	Delete Object Request/Acknowledge (SEMI E39.1)
Detach	S14F15,F16	Attached Object Action Request/Acknowledge (SEMI E39.1)
DetachSupervisedObject	S14F17,F18	Supervised Object Action Request/Acknowledge (SEMI E39.1)
Reattach	S14F13,F14	Object Attach Request/Acknowledge (SEMI E39.1)

6 Service Parameter Mapping

6.1 Table 2 shows the mapping between message parameters defined by RMS and data items defined by SECS-II. For parameters specified in the definitions of an RMS service, either the parameters themselves, or individual elements of complex parameters, map to a specific data item.

Table 2 Service Parameters Item Mapping Table

Parameter Name	*SECS-II Data Item*
RMSectionsCode	RCPSECCODE
AgentSpec_DescLength	RCPDESCLTH
AgentSpec_DescName	RCPDESCNM = “ASDesc”
AgentSpec_DescTimestamp	RCPDESCTIME
AgentSpec_SectionName	RCPSECNM = “ASDS”

Parameter Name	*SECS-II Data Item*
AttrData	RCPDESCLTH
AttrName	ATTRID
BodyData	RCPBODY
BodyDescLength	RCPDESCLTH
BodyDescName	RCPDESCNM = “BodyDesc”
BodyDescTimestamp	RCPDESCTIME
BodySectionName	RCPSECNM = “Body”
ErrorCode	ERRCODE
ErrorText	ERRTEXT
GenAttrSectionName	RCPSECNM = “Generic”
GenDescLength	RCPDESCLTH
GenDescTimestamp	RCPDESCTIME
LinkID	LINKID
NSAck	RMACK
RcpClass	RCPCLASS
RcpName	RCPNAME
RcpVersion	RCPVERS
RMAction	RCPCMD
RMAgent	AGENT
RMChangeStatus	RMCHGSTAT
RMDestRecID	RCPNEWID
RMNewNS	RMNEWNS
RMNSSpec	RMNSSPEC
RMOpID	OPID
RMParamName	RCPPARNM
RMParamRule	RCPPARRULE
RMParamSetting	RCPPARVAL
RMRcpID	RCPID
RMRcpSpec	RCPSPEC
RMRcpStat	RCPSTAT
RMSectionsCode	RCPSECCODE
RMSpace	RMSPACE
RcpDerivedID	RCPID
REAck	RMACK
REAttrDescLength	RCPDESCLTH
REAttrDescName	RCPDESCNM = “REAttrDesc”
REAttrDescTimestamp	RCPDESCTIME
REAtrSectionName	RCPSECNM = “Attribute”
REChangeStatus	CHGCODE
REDerivedID	RCPID
REDestRcpID	RCPNEWID
REOpID	OPID
REWOCode	REOWCODE
REParamName	PARAMNAME
REParamSetting	PARAMVALUE
RERcpID	RCPID
RERcpSpec	RCPSPEC
RESpace	RMSPACE
RESpec	RESPEC

Parameter Name	*SECS-II Data Item*
AttachToken	OBJTOKEN
RMChgType	RMCHGTYPE
RMManagerSpec	OBJSPEC
RMPermit	RMGRNT
RMRecorderSpec	OBJSPEC (S15,F43) and RMRECSPEC otherwise
RMRequestor	RMREQUESTOR
RMSegSpec	RMSEGSPEC
RMTimestamp	TIMESTAMP
TargetSpec	TARGETSPEC

6.2 There are also several data items that are used in the SECS-II messages which do not map to specific services parameters. Services with the same set of parameters are mapped to the same SECS-II message by adding an additional data item to differentiate between the services. In addition, SECS-II restrictions require multi-block primary messages to be preceded by a multi-block inquire/grant transaction defined within the same stream. For RMS messages, this requirement is satisfied with the S15,F1 and F2 messages. Data items defined for these two messages have no counterpart in RMS service parameters.

6.3 Table 3 contains the SECS-II data items that have no corresponding RMS service parameter:

Table 3 Additional Data Item Requirements Table

Function	*SECS-II Data Item*
Used to satisfy SECS-II conventions for linking a multi-block inquiry with a subsequent multi-block message.	DATAID
Used for the name of any *non-identifier* recipe attribute.	RCPATTRID
Used for the value of any *non-identifier* recipe attribute.	RCPATTRDATA
Used by S15,F19 to differentiate between RMNCopy and RMNRename services.	RCPRENAME
Used to satisfy SECS-II multi-block requirements.	RMGRNT
Used by S15,F3 to differentiate between RMNCreateNS and RMNDeleteNS services. Used by S15,F39 to differentiate between RMDRAddSegRecord and RMDRDelSegRecord. Used by S15,F41 to differentiate between RMDRAddChgRecord and RMDRDelChgRecord.	RMNSCMD
Used to satisfy SECS-II multi-block requirements. Neither required nor specified by RMS.	RMDATASIZE
Used by S15,F13 to differentiate between RMNCreate and RMNUpdate services.	RCPUPDT
Used by S15,F35 to differentiate between RMEDelete and RMEDeselect services.	RCPDEL
Used by S15,Fy to differentiate between A and B services.	RCPDEL

RELATED INFORMATION 1

NOTE: This related information is not an official part of SEMI E42.1 and is not intended to modify or supercede the official standard. Publication was authorized by full letter ballot procedures. Determination of the suitability of the material is solely the responsibility of the user.

R1-1 RMS-Compliant Equipment with a Stream 7 Host

Semiconductor manufacturers with legacy systems, capable of communicating using Stream 7 messages only, may still take advantage of many of the Recipe Management concepts when used with equipment that is Recipe Management-compliant. This section contains guidelines and examples for maintaining recipe attributes within the body of a Stream 7 process program.

R1-1.1 *Recipe Identifier* — Both the process program identifier PPID and the object identifier, including the recipe identifier, have a maximum length of 80 characters. Recipe class, name, and version number may be encoded into the data item PPID using the format required for the recipe identifier.

"/$CLASS_1$/$CLASS_2$/.../$CLASS_n$/NAME;VERSION"

It is recommended that recipe class names be unique, allowing use of the form

"/CLASS/NAME;VERSION"

to ensure the length conforms to the maximum allowed in SECS-II.

R1-1.2 *Recipe Sections* — All sections of the recipe are transferred together as the data item PPBODY. This allows all non-default values of recipe attributes to be preserved when a recipe is uploaded to the host and later downloaded back to equipment.

There are two potential methods for formatting PPBODY. If the recipe body is a text file, it is recommended that all attribute values are translated into text, including numeric and boolean values, with boolean values represented by the strings ".T." or ".F.". The second recommended method, applicable to both source and object forms, uses the structure of one of the SECS-II messages containing the managed recipe, as illustrated in Figure 1, or of an execution recipe. A cluster tool supporting shared *namespaces*, for example, might require multiple agent-specific data sets, while single-process equipment types would not.

R1-1.2.1 *Structuring PPBODY Using Text* — For purposes of transfer, recipes are regarded as composed of sections: The generic section, the body section, and zero or more agent-specific data-set sections (RMS, Section 12.1). This convention should also be maintained within PPBODY. The SECS-II messages used to transfer recipes (Stream 15, Functions 15 and 18) transfer the data-set sections last, because they are variable in number and are not always present. However, in a text format within PPBODY, it is generally easier to put all attributes before the body.

Since RMS uses the end-of-line character for source form recipes, this convention is recommended to separate the logical components of the sections containing attributes as well. A few reserved keywords are sufficient as tags to identify significant components: SECTION, ATTRIBUTE, LIST, PARAMNAME, PARAMSETTING, and PARAMRULE are recommended for consistency. Keywords are in uppercase for visual impact and are not delimited except for use of whitespace and the line-feed character ($0A_{16}$) used as the end-of-line (eol) character.

Whitespace is used to separate fields within a line. The symbol ␣ is used here to represent whitespace, which is used both as a delimiter to separate fields within a line and also for optional visual effect.

Each section begins with a section declaration of the form

SECTION␣"text"(eol)

where text is a section name, one of "Generic", "ASDS", "Body". Two sections are separated with two eol characters together. The eol convention within the body section, however, must conform to the recipe language specifications provided by the equipment manufacturer.

Each attribute begins with

ATTRIBUTE␣AttributeName

followed by the attribute value, which may be a single item, a list of single items, or a list of complex items, such as variable parameters. If a list, the keyword LIST precedes the values, each item in the list terminated with either a comma or, for the last item, the eol character.

This becomes

ATTRIBUTE␣AttributeName␣ “text”(eol) (single-value attributes)

or

ATTRIBUTE␣AttributeName␣LIST␣“text”, ␣“text”,...(eol)

for attributes consisting of lists of single items (e.g., LinkList or ExtRef).

If the keyword LIST if followed by eol, this indicates a list of complex items. The eol character, in this case, also is used to terminate the individual items in the high-order list. An attribute consisting of a list of complex items should identify the individual fields of each complex item. Those attributes containing variable parameters (e.g., Parameters) are of the form

ATTRIBUTE␣AttributeName␣LIST(eol)

␣PARAMNAME␣“text”, ␣PARAMSETTING␣“text”, ␣PARAMRULE␣“text”(eol)
␣PARAMNAME␣“text”, ␣PARAMSETTING␣“text”, ␣PARAMRULE␣“text”(eol)

...

␣PARAMNAME␣“text”, ␣PARAMSETTING␣“text”, ␣PARAMRULE␣“text”(eol)

Each individual field within a complex attribute value is delimited with the quotation mark (“) character. The eol character is used to terminate a section declaration, an attribute declaration, and a complex item within an attribute.

The recipe body is preceded with

SECTION “Body”(eol)

followed by the recipe body in the form normally used by the equipment.

In comparison of the two suggested formats, the advantage of the text form described in this section is that the attributes are human-readable and printable and therefore the user is able to modify those attributes that are not designated as read-only with a standard text editor.

The advantage of the SECS-II form (Figure 1) is that the equipment is more easily able to support a host that is only capable of using Stream 7.

R1-2 RMS-Compliant Recipe Management with Stream 7 Equipment

For RMS-Compliant Recipe Management applications dealing with legacy Stream 7 equipment, the situation is more restricted. While an RMS-compliant equipment may preserve attributes by bundling them inside the PPBODY of a process program, this option is not available when the situation is reversed because the equipment determines the permissible format(s) within the recipe. Therefore, all attributes of the recipe must be completely maintained by the host. The host in this situation may provide a recipe namespace that is dedicated either to an individual equipment installation, to a group of equipment capable of sharing the same process programs, or both.

If the equipment is compliant to Process Program Management as defined in SEMI E30 (GEM), then it will provide a collection event when the process program is created, changed, or deleted. Hosts that provide a dedicated recipe namespace for non-RMS equipment are able to use this event to trigger a process program upload and update the corresponding recipe.

Most Stream 7 equipment use only a single type of process program. This defaults to the recipe class PROCESS. The equipment is unaware of the internal fields used by the recipe identifier. If it preserves the contents of PPID and does not impose its own restrictions, it may be desirable to use the format of the recipe identifier, in particular the version component. This would allow version control at the host.

Some equipment maintains more than one type of process program and may distinguish between the different types either through providing additional Stream 7 messages, through naming conventions such as a ".ext" extension, through program length, or other devices.

Downloading a process program in Stream 7 is equivalent to both the operation of "sending" a recipe to a different namespace (at the equipment) and to downloading a recipe to the recipe executor for execution.

R1-2.1 *Structuring PPBODY Using SECS-II Message Format* — The second method of sending attributes within PPBODY is to use a SECS-II message structure that is used to send and receive recipes. Figure 1 illustrates using the structure of a managed recipe form S15,F13 and F16.

Note that cluster tools may provide full recipe namespace capabilities for its modules, including shared namespaces for modules of a specific type. In this case, a cluster recipe may contain multiple agent-specific datasets (m > 1). Other equipment may be able to use the structure for the *execution recipe*.

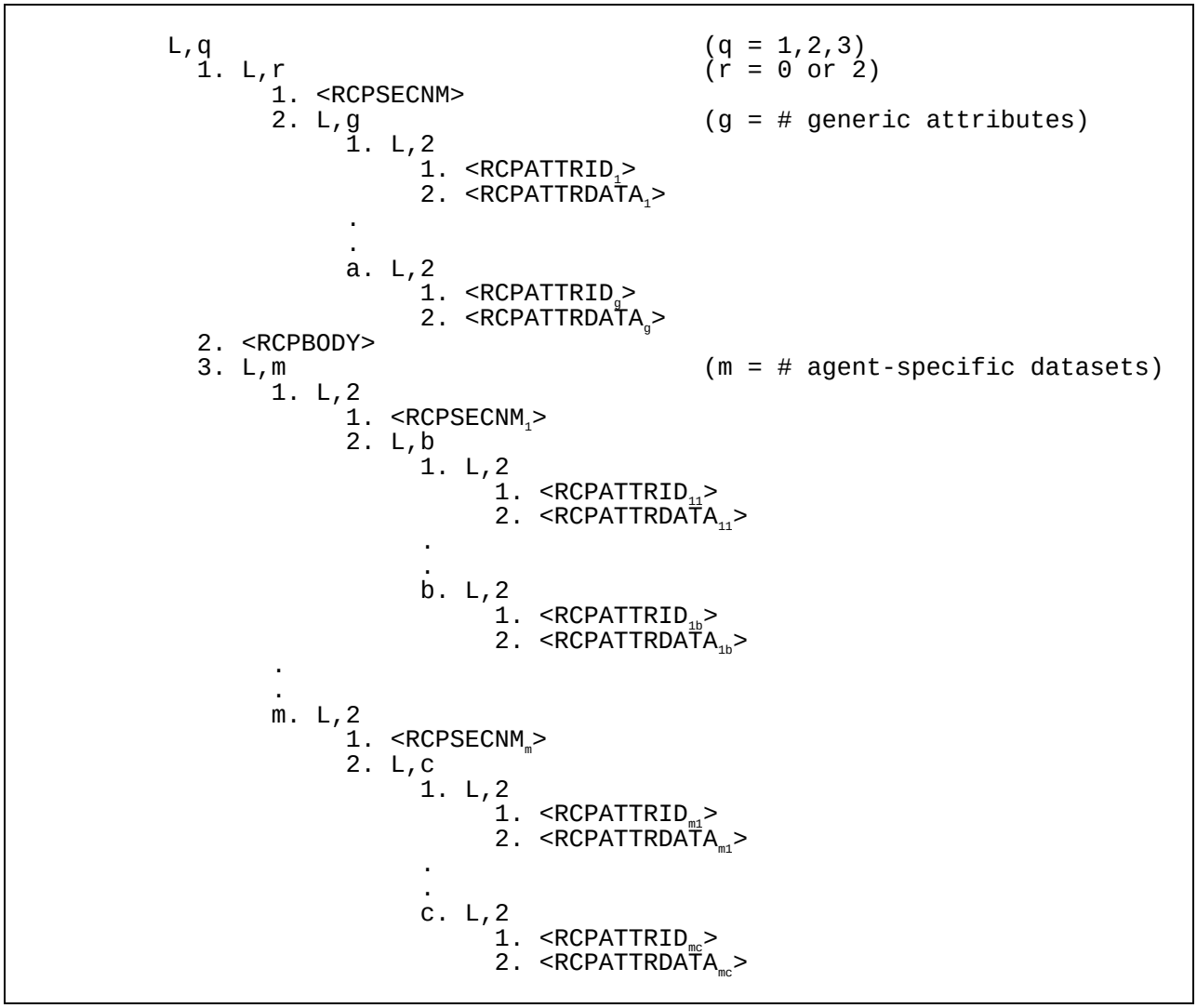

Figure 1
SECS-II Format for PPBODY

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E53-0704
EVENT REPORTING

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the Japanese Information & Control Committee. Current edition approved by the Japanese Regional Standards Committee on April 30, 2004. Initially available at www.semi.org June 2004; to be published July 2004. Originally published December 1996.

1 Purpose

1.1 Access to process data in equipment is crucial for effective process monitoring and control in a semiconductor manufacturing facility. This standard addresses the communication needs of semiconductor equipment and other factory objects, such as cell controllers or recipe servers, with respect to the timely collection and reporting of such data.

1.2 The purpose of this standard is to provide a general purpose set of event reporting services that may be offered by equipment suppliers. This document may be referenced, in whole or in part, by other standards addressing higher level application domains.

1.3 The communications services defined here will enable standards based inter-operability of independent systems. They shall allow application software to be developed which can assume the existence of these services and allow software products to be developed which offer them.

2 Scope

2.1 This standard is applicable to any stand-alone equipment, cluster module, cluster tool, or cell of automation in a factory. As such it addresses event reporting at all levels in the factory and equipment control hierarchy.

2.2 This standard requires significant communication and computational resources and is therefore not applicable at or below the level of I/O distribution (e.g., sensor bus) within the equipment.

2.3 This standard covers the reporting of data periodically and/or in response to events. Reports may also be requested on demand.

2.4 This standard presents a solution from the concepts and behavior down to the messaging services. It does not define the messaging protocol.

2.5 A messaging service includes the identification that a message shall be exchanged and definition of the data which is contained in that message. It does not include information on the structure of the message, how the data is represented within the message, or how the message is exchanged. This additional information is contained with the message protocol. The defined services may be applied to multiple protocols. Information on the mapping of these services to special protocols (e.g., SECS II) are added as adjunct standards.

2.6 The services assume a communications environment in which a reliable connection has been established between the user and the provider of the services. Establishing, maintaining, and releasing a connection is beyond the scope of this standard.

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 Referenced Documents

3.1 *SEMI Standards*

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

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

SEMI E38 — Cluster Tool Module Communications (CTMC)

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

SEMI E40 — Standard for Processing Management

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

4 Terminology

4.1 The following definitions are arranged in alphabetical order. Some terms are defined using terms defined elsewhere within this section. Other terms may be defined within SEMI's Compilation of Terms.

4.1.1 *attribute* — a data item associated with an object. An attribute may be referenced by zero or more data reports.

4.1.2 *behavior* — the manner in which something functions; how an object acts and reacts, in terms of its state changes and message passing.

4.1.3 *collection event* — an event that may be used to initiate the collection and reporting of data. A collection event may trigger an event report. A collection event may also start or stop one or more trace reports.

4.1.4 *data report* — a data report is a list of attribute names for a single object. Data reports may be pre-defined by a factory object or defined dynamically by the service user.

4.1.5 *default object* — the object assumed when no object specifier is supplied.

4.1.6 *event* — represents the occurrence of a change in the condition of a system (e.g., lot complete, temperature over range).

4.1.7 *event report* — a class of objects that has information related to an event and can be linked to user defined data reports and can send messages containing this information to a service user.

4.1.8 *factory object* — any identifiable object within the factory information and control architecture. Examples include equipment, a cluster process module, a cell controller, a recipe namespace server.

4.1.9 *object* — defined in the Object Services (SEMI E39).

4.1.10 *object specifier* — defined in the Object Services (SEMI E39).

4.1.11 *services* — a set of closely related messages.

4.1.12 *service provider* — a service provider is an application responsible for providing services to service users.

NOTE 1: There may be one or more service users concurrently accessing a single service provider. It is the responsibility of the service provider to provide its services transparently to each service user.

4.1.13 *service user* — a service user is any application that uses the services.

4.1.14 *trace report* — a class of objects which provides to a service user a means for collecting periodic readings of selected attributes of a system.

4.2 *Data Types*

4.2.1 *form* — type of data: float, positive integer, unsigned integer, integer, enumerated, boolean, text, formatted text, structure, list, ordered list.

4.2.2 *float* — a number represented by a mantissa and an exponent. It is used to represent numeric data which is continuous in value.

4.2.3 *positive integer* — May take the value of any positive whole number. Messaging protocol may impose a limit on the range of possible values.

4.2.4 *unsigned integer* — May take the value of any positive integer or zero. Messaging protocol may impose a limit on the range of possible values.

4.2.5 *integer* — May take the value of any negative or unsigned number. Messaging protocol may impose a limit on the range of possible values.

4.2.6 *enumerated* — May take on one of a limited set of possible values. These values may be given logical names, but they may be represented by any single-item data type.

4.2.7 *boolean* — May take on one of two possible values; TRUE or FALSE.

4.2.8 *Text* — A text string. Messaging protocol may impose restrictions, such as length or ASCII representation.

4.2.9 *formatted text* — a text string with an imposed format. This could be by position, by use of special characters, or both.

4.2.10 *structure* — a complex structure consisting of a specific set of items, of possibly mixed data types, in a specific arrangement.

4.2.11 *list* — set of one or more items that are all of the same form (one of the above forms).

4.2.12 *ordered list* — s list for which the order in which the items appear is significant.

5 Conventions

5.1 *Harel State Model* — This document uses the Harel State Chart notation to describe the dynamic behavior of the objects defined. 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.

5.1.1 Transition tables are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. A transition contains columns for transition #, current state, trigger, new state, action(s). The “trigger” (column 3) for the transition occurs while in the “current” state. The “actions” (column 5) include a combination of; 1) actions taken upon exit of the current state, 2) actions taken upon entry of the new state, and 3) actions taken which are most closely associated with the transition. No differentiation is made.

5.2 *OMT Object Information Model* — The object models are presented using the Object Modeling Technique developed by Rumbaugh, James, et al, in “Object-Oriented Modeling and Design,” Prentice Hall, Englewood Cliffs, NJ, ©1991. Overviews of this

notation are provided in an appendix of the Object Services SEMI E39.

5.3 *Object Attribute Representation* — The object information models for standardized objects will be supported by an attribute definition table with the following column headings:

Attribute	*Definition*	*Rqmt*	*Access*	*Form*
The formal text name of the attribute	Description of the information contained	Y or N	RO or RW	(see below)

5.3.1 The access column uses RO (read only) or RW (read and write) to indicate the access that users of object services have to the attribute.

5.3.2 A 'Y' or 'N' in the requirement (rqmt) column indicates if this attribute must be supported in order to meet fundamental compliance for the service.

5.3.3 The form column is used to indicates the format of the attribute. See Section 4.2 for definitions.

5.4 *Service Message Representation*

5.4.1 *Service Resource Definition* — The service resource definition table defines the specific set of messages for a given service group, as shown in the following table:

Service	*Type*	*Description*
Message name	N or R	The intent of the service

5.4.1.1 Type can be either notification (N) or request (R). Notification messages are initiated by the service provider. No response is expected. Request messages are initiated by the service user. Request messages ask for data or for an operation to be performed. Request messages expect a specific response (no presumption on the message content).

5.4.2 *Service Parameter Dictionary* — Each parameter should relate to either attributes from the object model or events of the dynamic model (Harel State Chart). All parameters to the services are listed in a single table or dictionary. The column headings for this parameter dictionary are as follows:

Parameter	*Description*	*Form*
Parameter X	Data type	A parameter

5.4.2.1 A row is provided in the table for each parameter of the service. The first column contains the name of the parameter. This is followed by columns describing the form and the contents of the parameter.

5.4.2.2 The description column describes the meaning of the parameter and interrelationships with other parameters.

5.4.2.3 The form column is used to indicate the type of data contained in the parameter (see Section 4.2 for definitions) and the possible values it may take on.

5.4.2.4 To prevent definition of numerous parameters named "XxxList," this document adopts the convention of referring to the list as "(list of) Xxx." In this case, the definition of the parameter "Xxx" will be given, not of the list. Where a list is used in both the request and the response, the list order in the request is retained in the response. A list must contain at least one element unless zero elements are specifically allowed.

5.4.3 *Service Message Definition* — There is a table for each service showing the parameter detail. The column headings for the service detail are as follows:

Parameter	*Req/ Ind*	*Rsp/ Cnf*	*Comment*
Parameter X	(see below)	(see below)	A description of the service

5.4.3.1 The columns labeled req/ind and rsp/cnf link the parameters to the direction of the message. The message sent by the initiator is called the "request" (req). The receiver terms the message the "indication" (ind). The receiver may then send a "response" (rsp), which the original sender terms the "confirmation" (cnf).

5.4.3.2 The request (req/ind) and response (rsp/cnf) 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 another parameter.
"U"	**User-defined 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.

6 Overview

6.1 Event reporting provides a dynamic and flexible means by which the service user can receive notification of events and data relating to objects defined here and in other SEMI standards.

6.2 Figure 1 illustrates a simple event reporting situation.

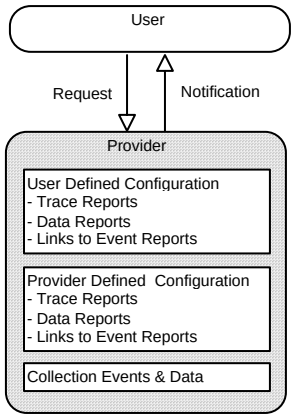

Figure 1
Simple Event Reporting

6.3 Event and variable identifiers may be defined by another standard with which equipment or an object complies. In either case, the definitions must be included in the documentation from the supplier.

6.4 The service user may define and access data reports using the event reporting request services. Data reports are sampled and sent by the service provider using the event reporting notification services.

6.5 Data reports may be pre-defined by the supplier. Such reports cannot be deleted and their report identifiers may not be used for dynamically created reports.

6.6 *Multiple Service Users* — In a more complex situation, several service users may require simultaneous access to the event reporting services (ERS). In this case, the service provider manages data report definitions independently for each service user as shown in Figure 2. A copy of each pre-defined report is made for each service user. Support for more than one simultaneous service user is optional.

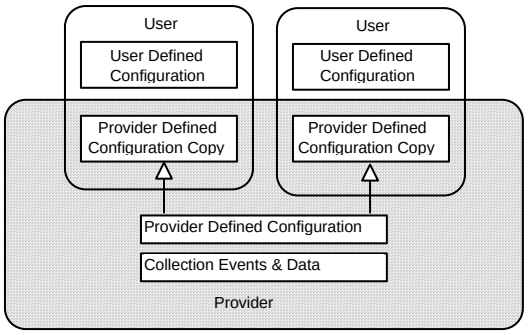

Figure 2
Multiple Service Users

6.7 *Using Object Services* — An event reporting service user can create reports dynamically by accessing attribute information using the object services (see SEMI E39) when they are provided (see Figure 3).

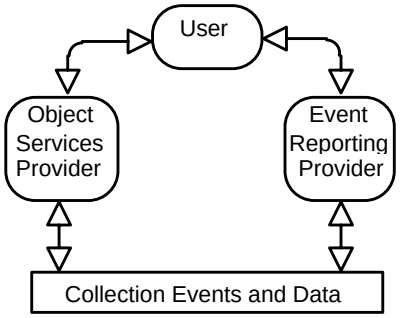

Figure 3
Using Object Services

7 Concepts

7.1 *Objects and Identifiers* — Much of the discussion and specification for event reporting services relies on an understanding of object technology and modeling techniques. The use of object technology for ERS implementations is not required. Further, it is not required that ERS implementations be compliant to the Object Services Standard (OSS). This standard does specifiy mechanisms by which ERS can take advantage of OSS when those services are provided.

7.1.1 Events are associated with an object and data is stored in object attributes. To uniquely identify an event or an attribute the object to which it relates must first be identified.

7.1.2 An object may be a physical entity such as stand-alone process equipment, a cluster tool or a cluster module. An object may be an abstraction such as a recipe, a process job or a transfer job. An object may be defined by the equipment supplier, or by a standard with which the equipment complies. An example of a cluster tool, containing process modules, is used in this section as the service provider. This example is for illustration purposes only. The same concepts apply for all factory objects as defined above.

7.1.3 Objects may be part of an object hierarchy or contained in other objects. The default object is the object which provides the service connection. In the case for batch processing equipment which is compliant to GEM, SEMI E30, the connection to the host is provided by the equipment and the equipment is the default object. Also, a cluster controller would be the default object of equipment containing process modules. However, from the point of view of the cluster controller, each connection to the service

providers in each connected module would be the default object. For example, the processing management service of PM:ETCH would be a default object for that module connection. But, the cluster controller would have differernt default objects on each of its other service connections.

7.1.4 An object specifier, (see SEMI E39), identifies an object within the scope of the service provider. An object specifier consists of one or more object type and object instance identifier pairs. For example, "PM:ETCH>" might be the object specifier of a process module (type=PM, identifier=ETCH) within a cluster tool. If the object specifier is omitted, then the default object is assumed.

7.1.5 Objects are considered local to the service provider even if they exist on a remote module. That means the service provider is responsible for detecting events and accessing data from those objects in a timely manner, by whatever means, to provide the specified services.

7.1.6 Equipment which implements OSS-compliant event reporting services allows the service user to reference equipment components by an object specifier; for example, "PM:ETCH>MFC:CCL4."

7.2 *Event Report Object Model* — Figure 4 shows the objects, attributes (variables), and relationships involved in event reporting. The notation used is defined in Section 5. The diagram illustrates the relationships between the service's messages and attributes (variables). The objects and their attributes are defined Section 8. Only the trace, data report, collection event, and data source objects' attributes and behaviors are standardized in that section. Note that this model presents a view of event reporting services from the perspective a single user. It is optional for the service provider to support simultaneous access from multiple service users. But for that option the provider shall present this view to each user independently.

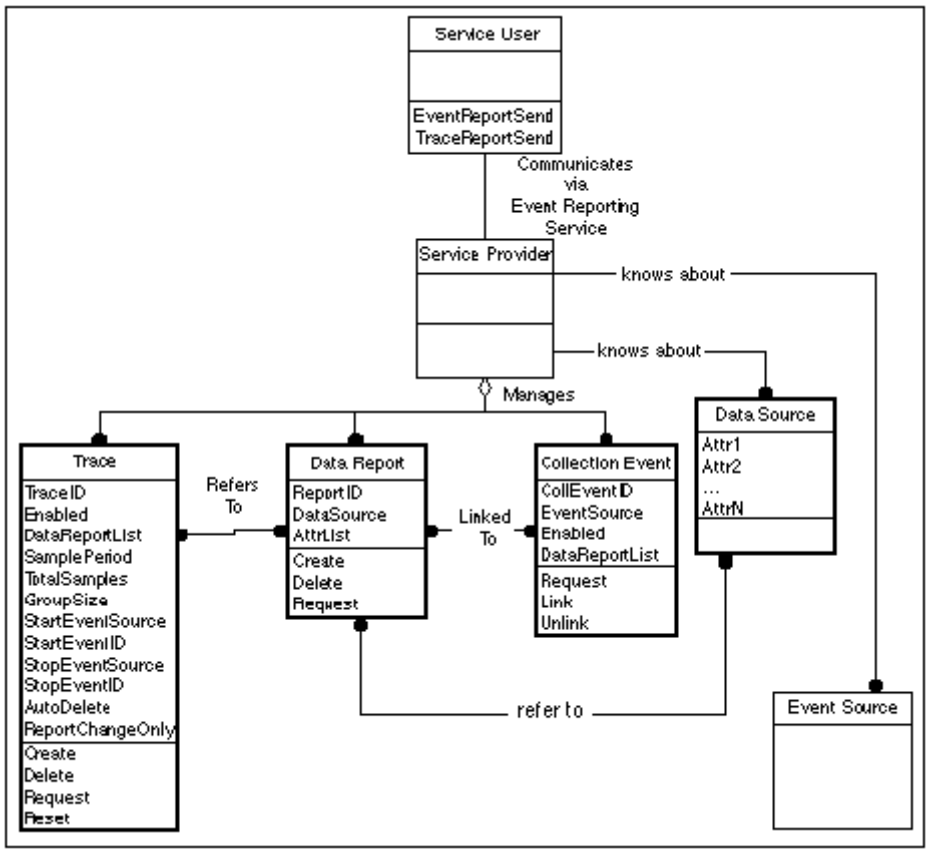

Figure 4
Event Report Object Model

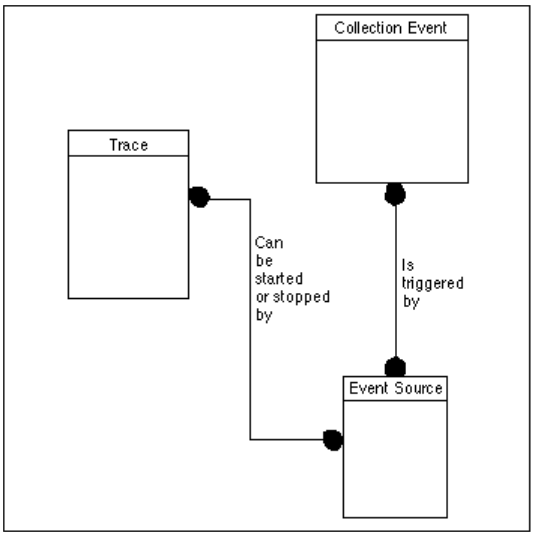

Figure 5
Trace Object Information Model

7.2.1 *Service Provider Object* — The service provider is responsible for compiling report samples according to each service user setup. It detects collection events, provides trace timing and accesses the value of data source object attributes (variables) in order to compile the data reports. ERS does not specify the attributes or behavior of this object. It is included in the event report object model to provide context for other objects within ERS.

7.2.2 *Service User Object* — The service user is responsible for configuring its own event reporting setup and receives reports from the service provider according to that setup. ERS does not specify the attributes or behavior of this object. It is included in the event report object model to provide context for other objects within ERS. However, the service user object shall be able to receive the event report send and trace report send messages from the service provider.

7.2.3 *Collection Event Object* — This object type or class is responsible to provide all the collection event identifiers for which event reports can be generated. Other factory services and objects may provide event notification messages which are important in the context of a service or functionality that they provide. But, in general, these notifications do not provide mechanisms to report information beyond their domain boundaries. For example, Processing Management, SEMI E40, includes an event notification for process complete (PRJobAlert (Milestone)). In order to collect data associated with this event, it must be "known" to the collection event object type. When it is "known" by the collection event object, then event reports can be generated using data or attributes from the data sources which are "known" by the event reporting service provider. A data source could provide information on process chamber pressure and temperature, which are attributes or variables important to the process but which are not known to the processing management service.

7.2.4 *Event and Data Source Objects* — The use of the event source and data source objects provides extensive capability to access the attributes of objects in a distributed environment. Where the event services provider has a domain which extends over multiple resources, such as a cell controller for a furnace bay or a cluster tool controller, these objects provide an unambiguous mechanism for addressing the data that the service user wishes to access.

7.2.4.1 The event and data source objects are to the event reporting services what directory object is in a file system of a computer's operating system. They should be thought of as containers of information which provide pointers (or names) of objects that are the actual sources of events or data.

7.2.4.2 The event source object is not directly accessible to the service user. Knowledge of events and event sources can be accumulated by interrogation of the collection event object using OSS.

7.2.4.3 Collection event identifiers shall be determined by either of two possible methods. One, the equipment supplier may define and supply with the equipment a list of collection event IDs (identifiers). Two, the equipment's collection event object can be interrogated with object services requests for collection event sources and identifiers. For example, using the second method, one could use object services to ask the collection event object for a list of collection event sources, which might include objects such as "PM:Etch>MFC:CCL4" or "TM>Aligner." Then using OSS, the collection event object can be queried for events associated with these sources. For the MFC a list of events might include "preslow," "overlimit," "valvemax," etc.

7.2.5 *Data Report Object* — A data report defines the variables (or attributes) of a specific object (data source) which are to be sampled. Data reports are either pre-defined or defined dynamically by the service user. A data report sample is a set of data containing the values of the variables defined for the data report. To sample a data report means that the current values of the variables at their data source object should be recorded. Data reports may be sampled and sent either with an event report and/or with a trace report. The service user may request:

- To sample an individual data report.
- Create a data report.

- Delete a data report.

7.2.6 *Event Report* — Event reports may be sent as messages to the service user whenever the associated collection event occurs. The collection event object is the source of the event report message. Event reporting is initially disabled. If the data to be sent with an event report has more than one data source, then there must be at least one data report for each data source to be included in the event report link definition. The service user may request:

- Event reporting to be enabled.
- Event reporting to be disabled.
- One or more data reports to be to linked (included in) the event report.

7.2.7 *Trace Report* — Trace reports are messages which are sent to the service user periodically. The trace report will contain the value(s) of the data item(s) specified in the data report(s) referred to by the trace object which is the source of the trace report.

7.2.7.1 The trace object contains the description of what to include in the trace report and trace objects are either pre-defined or defined dynamically by the service user. The service user may request:

- Trace reporting to be enabled.
- Trace reporting to be disabled.
- One or more trace reports to be reset at any time to their initial state.
- One or more trace reports to be deleted at any time.

7.2.7.2 When the enabled attribute of the trace object is set, then trace reporting may be started or in response to a start event. When the trace reporting starts, the sample count is reset to zero. A trace report may be stopped when a certain number of samples have been reported or when a stop event occurs.

7.2.7.3 The service user may specify a trace object to be automatically deleted once it has been stopped by whichever mechanism. Otherwise, the trace reporting simply stops. If no start event is defined, then the trace is restarted immediately. Otherwise, the next start event will restart the trace. Disabling or deleting a trace report stops reporting immediately.

7.3 *Scenarios* — The communications involved in event reporting are introduced below in the form of some representative scenarios. The full behavior and message set are covered in Sections 8.2 and 8.3 respectively.

7.3.1 *Event Reporting* — Figure 6 shows a scenario in which a data report is created by the service user and linked to an event report. Event reporting is then enabled. Subsequently, an event report is sent with the data reports attached whenever the collection event occurs. The service user then disables event reporting, preventing subsequent collection events from triggering the report. Data reports and links to event reports may be pre-defined by the equipment in which case the DataReportCreate and LinkCreate messages are omitted.

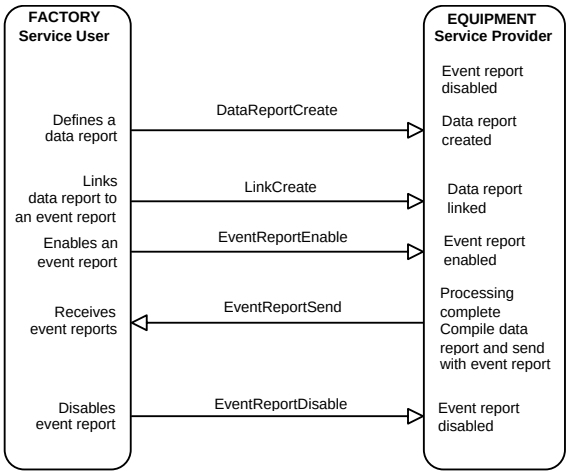

Figure 6
Event Reporting Scenario

7.3.2 *Trace Reporting* — Figure 7 shows a scenario in which a trace object is created with enabled and AutoDelete both set to TRUE. Since the trace object is enabled and there is no start event defined the data reports assigned to the trace object are sampled and sent immediately as a trace report message. The trace object timer is started and the sample counter set to zero. Each time the sample period elapses thereafter, a new trace report is sent, and the sample counter incremented. When the sample counter reaches the total samples parameter, then the trace report is automatically deleted.

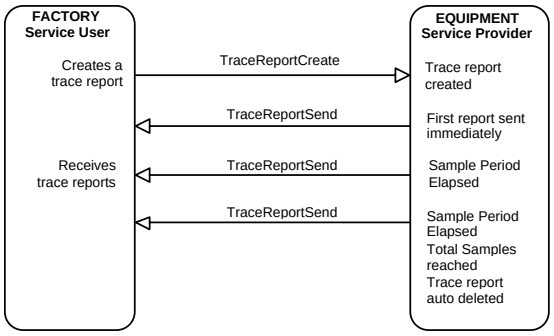

Figure 7
Trace Reporting Scenario

7.3.3 *Trace Reporting with Event Control* — Figure 8 shows a scenario in which a trace object is created with enabled and AutoDelete both set to FALSE. There is also a start event (ProcessingStarted) and a stop event (ProcessingComplete) defined. When the start event occurs, the data reports assigned to the trace are sampled and sent, the trace timer is started and the sample counter set to zero. Each time the sample period elapses thereafter, a report is sent and the sample counter incremented. Since a total samples parameter was not specified, the trace report stops only when the stop event occurs. Once the stop event has occurred, the service user disables the trace report preventing it from restarting when the next start event occurs.

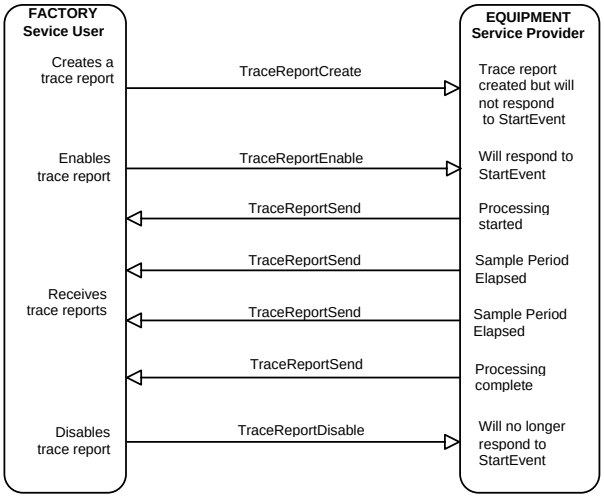

Figure 8
Trace Reporting with Event Control

7.3.4 *Event Reporting with Non-predefined CollectionEvent Object* — Additional capabilities of SEMI E39 (OSS) give Event Reporting standard more flexibility. For example, CollectionEvent objects can be created while equipment is running if the equipment has been installed the capability. If event sources are dynamically generated as process job, CollectionEvent objects must be created after the source is generated. If a user is not interested in such source, this capability is not required. However, if the user needs to get event report for such source, the capability is useful. This document doesn't recommend nor suppress that but it's depending on users.

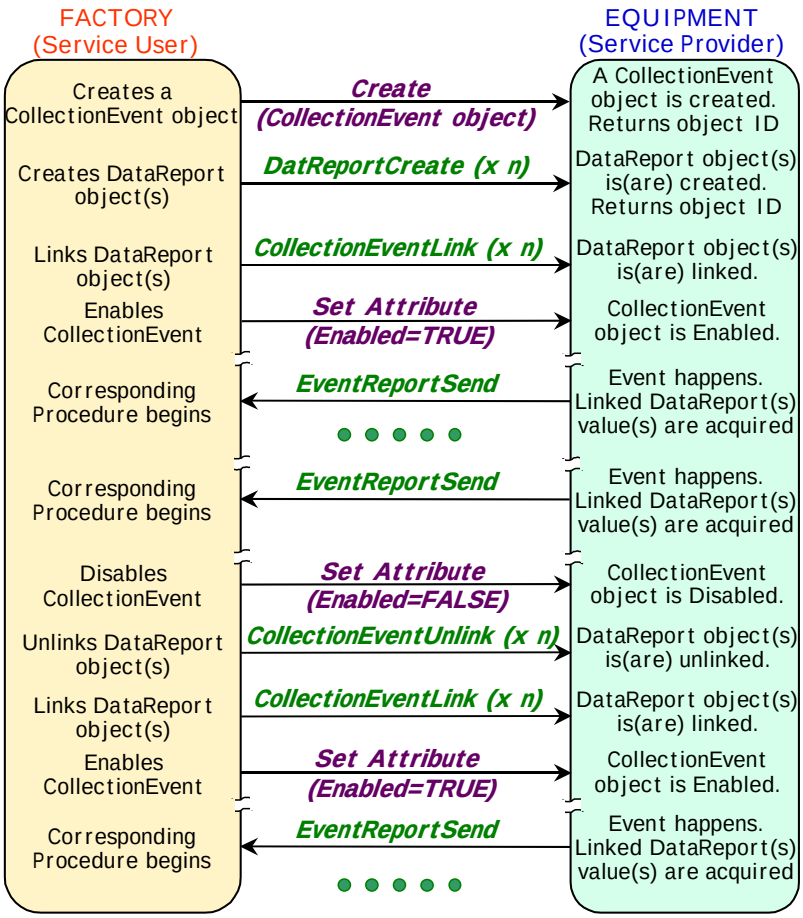

Figure 9
Non-predefined Collection Event Scenario

8 Specification

8.1 *Object Definitions*

8.1.1 *Collection Event Object* — A collection event object generates an event report each time it is triggered by a collection event from an event source. An event report contains the data samples from zero or more data reports to which it is linked by the collection event object.

Table 1 Collection Event Object Attribute Definitions

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ObjType	Specification of the Object Type of "Collection Event."	Y	RO	Text = "COLLEVENT"
ObjID	Object Identifier, within the scope of the event source object, for the Collection Event which is the trigger for the generation of an event report.	Y	RO	Text

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
EventSource	The object specifier which is the source of the event, see Section 7.1 and SEMI E39 (Object Services Standard).	N	RO	Formatted Text
DataReportList	Ordered list of data report object identifiers (text) to be sampled and sent with the event report.	Y	RO	List
Enabled	If this flag is cleared or false, then event reports will not be generated and sent when the associated Collection Event is detected.	Y	RW	Boolean

8.1.2 *Data Report Object* — The data report is part of the event reporting setup. It references zero or more attributes of a data source object and may be linked to zero or more event reports or trace reports.

Table 2 Data Report Object Attribute Definitions

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ObjType	Specification of the Object Type of "Data Report."	Y	RO	Text = "DATARPT"
ObjID	The object identifier of a pre-defined data report or one defined dynamically by the service user. ReportIDs are unique for eachservice user.	Y	RO	Text
DataSource	The object specifier of the object which has the attributes to be reported, see Section 7.1.	N	RO	Formatted Text
AttrList	An ordered list of attribute names from the Data Source object which will be sampled for inclusion in the data report.	Y	RW	List

8.1.3 *Trace Object* — The trace object generates trace reports based on the settings of its attributes. Trace object activities may be started and stopped by a collection event and may refer to one or more data reports.

Table 3 Trace Object Attribute Definitions

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ObjType	Specification of the Object Type.	Y	RO	Text = "TRACE"
ObjID	The identifier of a Trace Object. Trace reports contain this parameter to be used for reference purposes.	Y	RO	Text
Enabled	If this flag is cleared or false, then trace reports will not be generated and sent.	Y	RW	Boolean
DataReportList	Ordered list of data report object identifiers (text) to be sampled and sent with trace report.	Y	RO	List
SamplePeriod	The time delay between samples in seconds. Resolution may be in fractions of a second and should be specified by the service provider. The resolution available may also vary depending on the Data Source.	Y	RO	Float
TotalSamples	The number of samples to be taken before trace reporting stops automatically.	N	RO	Positive Integer
GroupSize	The number of samples taken before the report is sent to the service user.	N	RO	Positive Integer
StartEventSource	The object specifier of the object which is the source of the event which starts the trace reporting.	N	RO	Formatted Text
StartEventID	The event within the event source object which starts the trace reporting.	N	RO	Text
StopEventSource	The object specifier of the object which is the source of the event which stops the trace reporting.	N	RO	Formatted Text
StopEventID	The event within the event source object which stops the trace reporting.	N	RO	Text
AutoDelete	Specifies if trace report automatically deletes itself when its reporting cycle is complete.	N	RO	Boolean

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ReportChangeOnly	If TRUE, then only those data reports linked to the trace report which have changed since the last sample will be sent with the trace report. Otherwise, all data reports will be sent with the trace report.	N	RO	Boolean

8.1.4 *Data Source Object* — The data source is an entity, accessible to the service provider, which contains data (attributes) that may be of interest to the service user. The event reporting service can be composed of one or more data sources. In the case of only one data source, the use of its identifier in messages can be dropped. In this case, the data source is the default object. The attributes of a data source may be referenced by zero or more data reports.

Table 4 Data Source Object Attribute Definitions

Attribute Name	*Definition*	*Rqmt*	*Access*	*Form*
ObjType	The specification of the name for object type of " Data Source."	N	RO	Text = "DATASRC"
ObjID	The identifier of a Data Source Object. (This identifier is used in the object specifier when defining Data Report.)	N	RO	Text
AttrList	A list of the data source object attribute names. Names from this list may be used when defining a data report.	Y	RO	List

8.1.5 *Event Source Object* — This object represents components within a system that generate collection events. The service provider supplies mechanisms such that the when collection events are generated by the event source, then the collection event object, if enabled, will be triggered to generate an event report. ERS does not specify the attributes or behavior of the event source object. It is included in the event report object model to provide context for other objects within ERS.

8.2 *Object Behaviors* — This section specifies the behavior of the three objects in the information model which have standardized behaviors. The message detail for the service is addressed in a later section. Message flow diagrams presented in the previous section are useful to show simple situations. The data report, event report and trace report state models presented in this section provide the information necessary to extrapolate the message flow diagrams for all situations within the scope of this standard.

8.2.1 *Event Reporting State Models* — The behavior required for event reporting is fully specified by the service provider state model. All required service user behavior is inferred by this model.

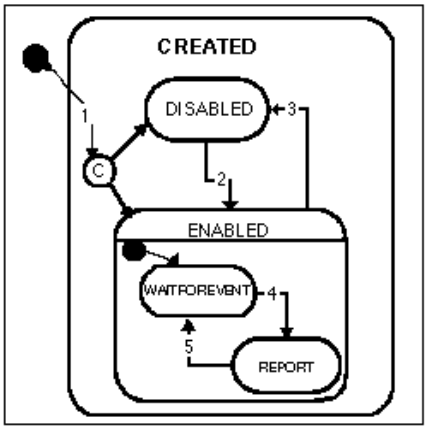

Figure 10
Collection Event Object State Diagram

8.2.1.1 Figure 10 shows the state diagram for a collection event object. The Harel State Model Notation used is described in Section 5.

Table 5 Collection Event Object State Definitions

State	*Definition*
CREATED	Each collection event is associated with a unique Collection Event ID.
ENABLED	The Service User will be notified by the sending of an event report when the collection event occurs.
DISABLED	Event reports will not be sent to the Service User.
WAITFOREVENT	The object is waiting for its associated Collection Event to occur.
REPORT	Send an Event Report to a Service User, sample Data Reports which have been linked.
notExist	User view of Collection Event Objects prior to connection (establishment of communications).

Table 6 Event Report Transition Table

#	*Current State*	*Trigger*	*New State*	*Action(s)*
1		Service user establishes connection.	CREATED and ENABLED or DISABLED	A Collection Event Object is created on behalf of the service user for every collection event. The event reporting Enable attribute setting is conditional. It may get its value from stored configuration information.
2	DISABLED	Enable attribute set to TRUE	ENABLED	Event reporting is enabled for a specific collection event.
3	ENABLED	Enable attribute set to FALSE	DISABLED	Event reporting is disabled for a specific collection event.
4	WAITFOREVENT	Collection Event occurs	REPORT	Sample linked data reports. Send event report with linked data reports.
5	REPORT	Reporting Complete	WAITFOREVENT	The object waits for the collection event.

8.2.2 *Data Report Object State Model* — Each data report shall follow the transitions as specified in Figure 11. Multiple instances of data report are managed by the event services. There are two basic types of data reports; pre-defined and user defined.

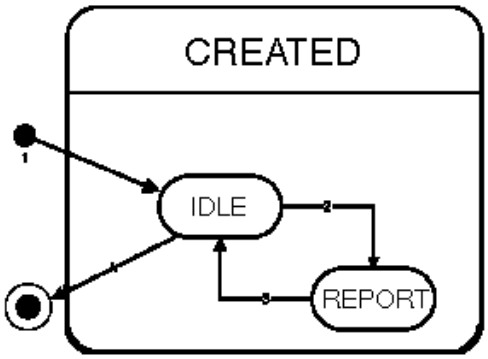

Figure 11
Data Report Object State Chart

Table 7 Data Report Object State Definitions

State	*Definition*
notExist	The state from which the service provider creates reports. NOTE: Some reports may be predefined, that is, they are not created by a user request.
IDLE	Reports are sampled and generated only when requested from an external source such as the Trace or Event Report object or the Service User directly.
CREATED	This is the super-state of Idle and Report. Variables and/or attributes to be sampled are defined.
REPORT	The variables or attributes specified in the Data Report Object are sampled and then sent to the report requestor.

Table 8 Data Report State Transition

#	*Current State*	*Trigger*	*New State*	*Action(s)*
1		CreateRequest	CREATED and IDLE	New ReportID is defined. Wait for an actionable event.
2	IDLE	ReportRequest	REPORT	Sample data in report definition.
3	REPORT	SamplingComplete	IDLE	Send data to report requestor.
4	IDLE	DeleteRequest and not pre-defined	notExists	ReportID is no longer valid.

8.2.3 *Trace Object State Model* — Figure 12 shows the state diagram for a trace object. The Harel State Model Notation used is described Section 5. The state transition table is shown in Table 10.

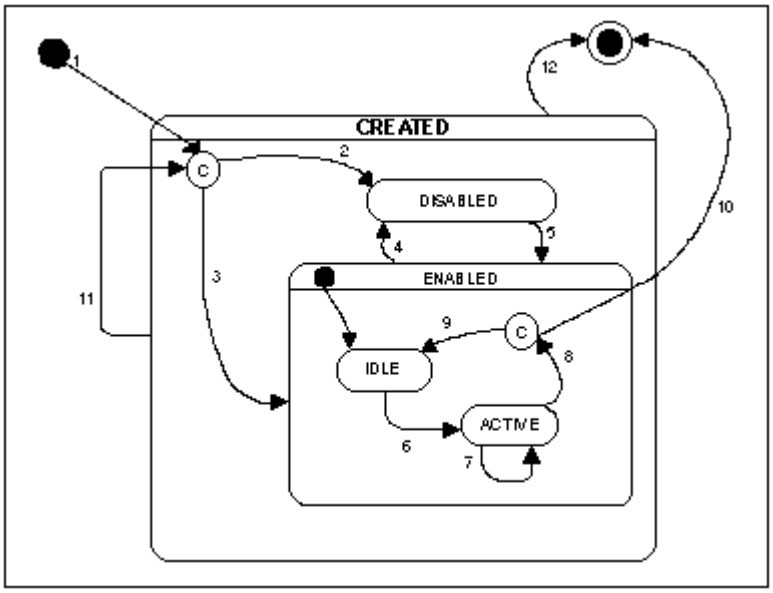

Figure 12
Trace Object State Diagram

Table 9 Trace Object State Definitions

State	*Definition*
CREATED	The Trace Object is created and ready to be accessed.
DISABLED	Trace report is created but will not be sampled or sent.
ENABLED	Trace object may begin generation of trace reports. See sub-states IDLE and ACTIVE.
IDLE	Trace reports are not being generated.
ACTIVE	The trace reports are being generated and sent periodically.

Table 10 Trace Object State Transition Table

#	*Current State*	*Trigger*	*New State*	*Action(s)*
1	(deleted)	Trace Object Create	(condition)	A Trace Object is created with Report ID, Enabled, Group Size, Total Samples, Sample Period, Start Event, Stop Event, and Auto Delete parameters.
2	(condition)	Enabled was initially FALSE.	DISABLED	Trace reporting is disabled.
3	(condition)	Enabled was initially TRUE.	ENABLED	Trace reporting is enabled.
4	ENABLED	Trace Report Disable	DISABLED	Trace reporting is disabled.
5	DISABLED	Trace Report Enable	ENABLED	Trace reporting is enabled.

#	*Current State*	*Trigger*	*New State*	*Action(s)*
6	IDLE	Start Event is not defined or Start Event is defined and occurs.	ACTIVE	Sample Data Reports and send. Start sample timer. Reset sample counter. Reset group counter. Wait for sample timer to elapse.
7	ACTIVE	Sample period elapses.	ACTIVE	Sample data reports into a trace report. If group sample counter equals group size, then send the trace report and reset group counter. Then prepare to generate the next trace report. Start sample time and wait for it to elapse.
8	ACTIVE	Stop Event is defined and occurs or… Total Samples > 0 and sample counter equals Total Samples.	(condition)	Sample data reports into a trace report. Send the trace report.
9	(condition)	Auto Delete is FALSE	IDLE	(none)
10	(condition)	Auto Delete is TRUE	(deleted)	Delete Trace Object.
11	CREATED	Event Report Reset	(condition)	Reset Trace Object state back to initial state when it was first created.
12	CREATED	Event Report Delete	(deleted)	Delete Trace Object.

8.3 *Messaging Services Detail* — This section specifies the messages required to implement the event reporting service. The messages were introduced in Section 7.3.1. These services are independent of the messaging protocol used.

The specification includes a list of required messages, the definition of the parameters contained within the messages, and data type of the parameters. Not defined here is the internal structure of the actual messages as transferred, including order of the parameters and how various data structures and data types are represented.

The service message notation used in the tables below is described in the Conventions, Section 5.4.

8.3.1 *Service List*

Table 11 Event Reporting Services

Service	*Type*	*Description*
Event Report Request	R	Requests the specified event report, and the data reports linked to it, to be sampled and returned to the service user.
Event Report Send	N	Notifies the service user of the occurrence of an event with a time stamp indicating when the event occurred. Any data reports linked to the event report are sampled and attached to the event report sent to the service user.
Data Report Create	R	Creates a data report which defines a set of attributes of an object to be reported.
Data Report Delete	R	Deletes one or more data report definitions.
Data Report Request	R	Requests the specified data report to be sampled and returned to the service user.
Collection Event Link	R	Request a Collection Event Object to link a Data Report. All Data Reports linked to an enabled Collection Event Object are sampled and sent with the event report when the collection event occurs.
Collection Event Unlink	R	Unlinks one or more Data Reports from the Collection Event Object.
Trace Create	R	Creates a trace object which defines the sample period, group size, start and stop events, auto delete, report change only, and the initial state (enabled or disabled).
Trace Delete	R	Deletes one or more trace object definitions.
Trace Report Request	R	Requests that the data reports assigned to the trace report be sampled and returned to the service user.
Trace Report Send	N	Notifies the service user of trace report samples taken. Group size samples are taken before the service user is notified. After notifying the service user, the group counter is reset to zero. A trace report sample is taken either when the start event occurs or the sample period elapses.

Service	*Type*	*Description*
Trace Report Reset	R	Resets one or more trace objects back to their initial state when created regardless of the current state of the trace object. All trace objects may be reset at once providing a simple way to re-establish the event reporting setup after modifications have been made.

8.3.2 *Use Object Services* — Attributes of the standardized objects listed in Table 12: SEMI E39 shall be accessed by using the GetAttr and SetAttr messages as defined in SEMI E39.

Table 12 SEMI E39 Access

Object	*Attribute*	*Comment*
Collection Event	Enabled	When set to TRUE, a Collection Event Report will be sampled and sent when the Collection Event occurs. When FALSE, Collection Event Reports are not sampled and sent.
Trace	Enabled	When set to TRUE, a Trace Report will be sampled and as described by the Trace Report definition. When FALSE, Trace Reports are not sampled and sent.
Data Source	AttrList	Use GetAttr to get the names of the attributes of a data source. These names can be used in the Data Report Create message.
Data Report	AttrList	Use GetAttr and SetAttr to change the variables included in a data report definition.

8.3.3 *Parameter Dictionary*

Table 13 Event Reporting Parameters

Parameter	*Definition*	*Form*
EvtSrcSpec	The object specifier of the Event Source Object.	Formatted text
CollEventID	References the Object Identifier of the collection event, within the scope of the event source object, that triggers an event report to be sent.	Text
Enabled	Specifies whether an Event or Trace Report is enabled.	Boolean: TRUE – Successful FASLE – Unsuccessful
DatSrcSpec	The object specifier of the data source object.	Formatted text
(list of) AttrName	An ordered list of attribute names from the data source object which specify the samples to be included in a data report. Maximum must be documented by supplier.	Text
ReportID	References the Object Identifier of a data report, either pre-defined by the service provider or specified by the service user.	Text
TraceID	References the Object Identifier of a trace object specified by the service user.	Text
(list of) ReportID	A list of data reports to be linked to a trace. Maximum must be documented by supplier.	Text
SamplePeriod	The time delay between samples. Minimum and maximum must be documented by supplier.	Float
TotalSamples	The number of samples to be taken before trace data reporting stops automatically.	Positive Integer
GroupSize	The number of trace report samples made before the report is sent to the service user. Maximum group size must be documented by supplier.	Positive Integer
StartEvtSrcSpec	The object which is the source of the event which starts the trace reporting.	Text
StartEventID	The Collection Event Object Identifier within the event source object which starts the trace reporting.	Text
StopEvtSrcSpec	The object which is the source of the event which stops the trace reporting.	Text
StopEventID	The Collection Event Object Identifier within the event source object which stops the trace reporting.	Text
AutoDelete	Specifies if trace report automatically deletes itself when its reporting cycle is complete.	Boolean: TRUE – Successful FASLE – Unsuccessful

Parameter	*Definition*	*Form*
ReportChangeOnly	If TRUE, then only data reports linked to the trace object which have changed since the last sample will be sent with the trace report. Otherwise, all data reports will be sent with the trace report.	Boolean: TRUE – Successful FASLE – Unsuccessful
(list of) DataReportSample	A list of data reports. Each data report sample contains the values of the variables specified in the data report at the time at which the sample was taken (see TimeStamp).	Structure
(list of) TraceReportSample	Trace report data. Each trace report contains a set of data report samples taken at a single point in time. This parameter is used when several trace report samples are made before notifying the service user (see Group Size).	Structure
ERAck	Indicates whether the activity was successful.	Boolean: TRUE – Successful FALSE - Unsuccessful
ErrorCode	Identifies errors that may arise.	Enumerated: Unknown event Unknown attribute Unknown data report Duplicate Report ID Data report not linked Unknown trace report Duplicate Trace ID Cannot delete predefined Too many data reports Sample period out of range Group size too large text
ErrorText	Text in support of Error code.	Text
ERStatus	Reports the acceptance or rejection of a requested operation.	Structure Composed of: ERAck (list of) Status
Status	Reports any errors found.	Structure Composed of: ErrorCode ErrorText

Table 14 Data Report Sample Parameter Detail

Parameter	*Definition*	*Form*
ReportID	Unique identifier of a data report.	Text
TimeStamp	Records time at which report sample values were recorded.	Formatted text: YYYMMDDhhmmsscc
(list of) AttributeValue	An ordered list of attribute values that match the (list of) AttrName in the data report.	(depends on attribute)

Table 15 Trace Report Sample Parameter Detail

Parameter	*Definition*	*Form*
(list of) DataReportSample	Samples of data reports assigned to the trace report.	Structure

8.3.4 *Service Detail*

8.3.4.1 *Event Report Request* — When the event object receives this message, it immediately generates and sends an event report to the service user.

Table 16 Event Report Request Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
EvtSrcSpec	C	C(=)	Source object for requested event report. If omitted, then the default object is assumed.
CollEventID	M	M(=)	Identifies requested Event Report. If not recognized, then the error code “unknown event” is returned.
TimeStamp	-	C	Time at which request was received. Parameter is omitted if Event ID is not recognized.
(list of)DataReportSample	-	C	Samples of data reports linked to the event. Parameter is omitted if Event ID is not recognized.
ERStatus	-	C	Possible errors: Unknown event

8.3.4.2 *Event Report Send* — This is a notification message to the service user. It contains the event report for the collection event which has triggered this action.

Table 17 Event Report Send Service Detail

Parameter	*Req/Ind*	*Comment*
EvtSrcSpec	C	Source object for event report to be sent. If omitted, then the default object is assumed.
CollEventID	M	Event Report being sent.
TimeStamp	M	Time at which Collection Event occurred.
(list of)DataReportSample	M	Samples of data reports linked to the event. An empty list can be sent by the service prvider when there are no data reports linked to the Collection Event.

8.3.4.3 *Data Report Create*

Table 18 Data Report Create Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
DatSrcSpec	C	-	Source object for data report to be created. If omitted, then the default object is assumed.
AttrList	M	-	Attributes of data source object to be included in data report. If an AttrName is not recognized, then the error code “unknown attribute” is returned.
ReportID	C	M(=)	If ReportID is not specified in the request, then the provider will return a unique ID to the user. If the user specifies ReportID, then the user must be responsible for making it unique, otherwise, the ReportID already identifies a Data Report, then that Report definition will be overwritten.
ErrorCode	-	C	Possible errors: Unknown Data Source Unknown Attribute

8.3.4.4 *Data Report Delete*

Table 19 Data Report Delete Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
(list of) Report ID	C	C	Identifies data report(s) to be deleted. If data report ispre-defined, then the error code “Cannot delete pre-defined” is returned. If a ReportID is not recognized, then the error code “unknown data report” is returned. If parameter is omitted in the request, then all data reports are deleted. In all cases, the (list of) ReportID that are deleted, if any, is returned in the response.
ERStatus	-	C	ERAck is set to FALSE if any requested ReportID cannot be deleted. ErrorCodes: Unknown data report Cannot delete predefined ErrorText shall be set to the ReportID of reports which could not be deleted.

8.3.4.5 *Data Report Request* — Message asking the service provider to sample the data report and send the resulting report.

Table 20 Data Report Request Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
ReportID	M	C(=)	Identifies requested Data Report to sample. If Report ID is not recognized then the error code “unknown data report” is returned. Parameter is omitted in the response when the ReportID is recognized by the service provider.
DataReportSample	-		Parameter is omitted if ReportID is not recognized.
ERAck	-	C	In the case where no Data Report can be returned then this value is returned set to FALSE. This informs the service requestor that the message was received but that for some reason, the report could not be sampled. The most likely reason is the ReportID was not recognized.

8.3.4.6 *Collection Event Link*

Table 21 Collection Event Link Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
EvtSrcSpec	C	C(=)	Source object for event report to be linked. If omitted, then the default object is assumed.
CollEventID	M	M(=)	Identifies Collection Event Object to be linked. If CollEventID is not recognized, then the error code “unknown event” is returned. This error takes precedence over other errors.
(list of) ReportID	M	-	Identifies Data Report(s) to be linked to an Event Report for this Collection Event Object. If ReportID is not recognized, then the error code “unknown data report” is returned.
ERStatus	-	C	ERAck is set to FALSE if any ReportID cannot be linked. ErrorCodes: Unknown event Unknown data report ErrorText shall be set to the ReportID of reports which could not be linked.

8.3.4.7 *Collection Event Unlink*

Table 22 Collection Event Unlink Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
EvtSrcSpec	C	C(=)	Source object for event report to be unlinked. If omitted, then the default object is assumed.
CollEventID	M	M(=)	Identifies Collection Event to be unlinked. If the ID is not recognized, then the error code “unknown event” is returned. This error takes precedence over other errors.
(list of) ReportID	M	-	Identifies data report to be unlinked. If link is pre-defined, then the error code “Cannot delete predefined” is returned. If ReportID is not recognized, then the error code “unknown data report” is returned. If data report is not linked to the specified event, then the error code “data report not linked” is returned.
ERStatus	-	C	ERAck is set to FALSE if any ReportID cannot be unlinked. ErrorCodes: Unknown event Unknown data report Data report not linked Cannot delete predefined ErrorText shall be set to the ReportID of reports which could not be unlinked.

8.3.4.8 *Trace Create*

Table 23 Trace Create Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
TraceID	C	M(=)	Identifier of Trace Object to be created. The service user can be responsible for the uniqueness of the Trace ID. If duplicate ID is used, then the current Trace Object definition overwritten. If omitted in the request, then the Service Provider will generate a unique identifer.
Enabled	M	-	If TRUE, then the initial state of trace report is enabled and will start reporting when the next Start Event occurs, if one is defined, or immediately otherwise.
(list of) ReportID	M	-	Data reports to be linked to trace report.
SamplePeriod	M	-	A Sample Period outside the range supported by the service provider will return the error code “Sample period out of range.”
TotalSamples	C	-	If this is equal to zero or is omitted, then the trace reporting will continue until it is stopped by some other means (i.e., by Stop Event, by Trace Report Disable or by Trace Report Delete).
GroupSize	C	-	If this is omitted, then every sample is immediately reported.
StartEvtSrcSpec	C	-	The object which is the source of the event which starts the trace reporting. If omitted, then the default object is assumed.
StartEventID	C	-	The event within the event source object which starts the trace reporting. If this is omitted, then trace reporting begins as soon as it is enabled.
StopEvtSrcSpec	C	-	The object which is the source of the event which stops the trace reporting. If omitted, then the default is assumed.
StopEventID	C	-	The event within the event source object which stops the trace reporting. If this is omitted, then trace reporting must be stopped by some other means (i.e., by Total Samples, or by Trace Report Delete).
AutoDelete	C	-	If this is omitted, then AutoDelete is assumed to be TRUE.
ReportChangeOnly	C	-	If this is omitted, then ReportChangeOnly is assumed to be FALSE.
ErrorCode	-	C	Possible errors: Duplicate trace ID Too many data reports Sample period out or range Group size too large Unknown event

8.3.4.9 *Trace Delete*

Table 24 Trace Delete Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
(list of) Trace ID	C	C	Identifies Trace Object(s) to be deleted. If a Trace Object ispre-defined, then the error code “Cannot delete predefined” is returned. If a TraceID is not recognized, then the error code “unknown trace report” is returned. If parameter is omitted in the request, then all trace reports are deleted. In all cases, the (list of) Trace ID that are deleted, is returned in the response.
ERStatus	-	C	ERAck is set to false if any requested TraceID cannot be deleted. ErrorCodes: Unknown trace report Cannot delete predefined ErrorText shall be set to the TraceID of reports which could not be deleted.

8.3.4.10 *Trace Report Request* — Request the data reports specified by the trace object ID be sampled and sent to the service user.

Table 25 Trace Report Request Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
TraceID	M	M(=)	Identifies requested trace report.
TimeStamp	-	C	Time at which request was serviced and data report samples taken. Parameter is omitted if Trace ID is not recognized.
(list of) DataReportSample	-	C	Samples of data reports assigned to the trace report. Parameter is omitted if Trace ID is not recognized.
ERAck	-	C	If the Data Report Samples cannot be sent, then this item shall be returned with its Boolean value set to false. The most likely cause of this would be that the TraceID was undefined.

8.3.4.11 *Trace Report Send* — The notification message sent by the service provider to the service user when the trace object has prepared a trace report according to its definition.

Table 26 Trace Report Send Service Detail

Parameter	*Req/Ind*	*Comment*
TraceID	M	Identifies trace report.
(list of) TraceReportSample	M	Number of samples in list is determined by Group Size parameter in TraceReportCreate.

8.3.4.12 *Trace Object Reset*

Table 27 Trace Reset Service Detail

Parameter	*Req/Ind*	*Rsp/Cnf*	*Comment*
(list of) TraceID	C	M	Identifies trace report(s) to be reset. If a Trace ID is not recognized, then the error code “unknown trace report” is returned. If parameter is omitted in the request, then all trace reports are reset. In all cases, the (list of) TraceID that are reset is returned in the response.
ERStatus	-	C	ERAck is set to false if any requested TraceID cannot be reset. ErrorCodes: Unknown trace report ErrorText shall be set to the TraceID of reports which could not be reset.

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E53.1-0704
SECS-II SUPPORT FOR EVENT REPORTING STANDARD

This standard was technically approved by the Global Information & Control Committee and is the direct responsibility of the Japanese Information & Control Committee. Current edition approved by the Japanese Regional Standards Committee on April 30, 2004. Initially available at www.semi.org June 2004; to be published July 2004. Originally published December 1996.

1 Purpose

1.1 This document maps the services and data of its prime document, SEMI E53, to SECS-II streams and functions and data definitions.

2 Scope

2.1 This is the standard way to implement the Event Reporting, which provides event-based reporting using the SECS-II message format.

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 Referenced Standards

3.1 *SEMI Standards*

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

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

SEMI E39.1 — SECS-II Protocol for Object Services Standard (OSS)

SEMI E53 — Event Reporting

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

4 Terminology

None.

5 Mapping of Event Reporting Messages

Table 1 Event Messages Mapping

Service Message Name	*Stream, Function*	*SECS-II Name*
Event Report Send	S6F11,F12,F13,F14	Event Report Send/Acknowledge Annotated Event Report Send/ Acknowledge
Event Report Request	S6F15,F16	Event Report Request/Data
Data Report Request	S6F19,F20	Individual Report Request/Data
Collection Event Link Request	S17F9,F10	Collection Event Link Request/Acknowledge
Collection Event Unlink Request	S17F11,F12	Collection Event Unlink Request/Acknowledge
Data Report Create	S17F1,F2	Data Report Create Request/Acknowledge
Data Report Delete Request	S17F3,F4	Data Report Delete Request/Acknowledge
Trace Create Request	S17F5,F6	Trace Create Request/Acknowledge
Trace Delete Request	S17F7,F8	Trace Delete Request/Acknowledge
Trace Report Send	S6F27,F28	Trace Report Send/ Acknowledge
Trace Reset Request	S17F13,F14	Trace Report Reset Request/Acknowledge
Trace Report Request	S6F29,F30	Trace Report Request/Data
Create Object	S14F9,F10	Create Object Request/Acknowledge (SEMI E39.1)
Delete Object	S14F11,F12	Delete Object Request/Acknowledge (SEMI E39.1)

6 Event Parameters Mapping

Table 2 Event Service Parameter Mapping

Parameter	*SECS-II Data Item or Object Attribute*
ErrorCode	ERRORCODE
ErrorText	ERRTEXT
ReportID	RPTID
DataSrcSpec	DATASRC
CollEventID	CEID
EvtSrcSpec	EVNTSRC
Enabled	CEED
(list of) AttrName	(list of) VID
ReportChangeOnly	RPTOC
TraceID	TRID
SamplePeriod	TRSPER
TotalSamples	TOTSMP
GroupSize	REPGSZ
StartEventID	CEID
StopEventID	CEID
AutoDelete	TRAUTOD

7 Implementation Details

7.1 Many of the messages of the Event Reporting Standard are implemented with currently defined SECS-II messages. Certain restrictions must be applied to their usage. These restrictions are specified in this section.

7.2 To be OSS-compliant, the following data items must use the format 20 specification:

- RPTID
- CEID
- TRID
- VID

7.2.1 The format of the ASCII strings shall conform to the Object identifier specifications in the Object Services Standards, document SEMI E39 and SEMI E39.1.

7.3 Each Data Report sample, whether stand alone or as part of an Event or Trace report, will report TIMESTAMP as the first item in its list of report variables and/or object attributes. This means for reports defined by the service user, the user should define TIMESTAMP as the first VID in the report definition. Further, all of the service provider's predefined reports shall deliver the value of TIMESTAMP in the first V.

7.4 Serveral capabilities of the Event Reporting Standard are accessed through the Object Services Standard. When an instance of a report (Report, Event, or Trace) object has been created, then its attributes can be read and written using the Object Service's GetAttr and SetAttr messages.

7.4.1 Object Services shall be used to set RPTOC and to get CHGND.

7.4.2 Object Services shall be used to set or clear CEED.

7.4.3 EVNTSRC shall be provided in supplier documentation or as a list of Event Sources obtained by using Object Services to interrogate the Collection Event Object, COLLEVENT, with the GetAttrName request message.

7.5 Event Reporting Services is protocol independent and therefore, makes no mention of SECS-II multi-block access and grant messages. When the Event Reporting Services use the SECS-II protocol, then S6F11,F13 shall be preceded by S6F5 if a report will be multi-block. Trace Report Send, S6F27 shall also use the multi-block access/grant transaction if the report will be multi-block.

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E54-0705
SENSOR/ACTUATOR NETWORK STANDARD

This standard was technically reapproved by the Global Information and Control Committee and is the direct responsibility of the North American Information and Control Committee. Current edition approved by the North American Regional Standards Committee on March 14, 2004. Initially available at www.semi.org May 2004; to be published July 2004. Originally published in 1997; previously published March 2003.

NOTICE: The designation of SEMI E54 was updated during the 0705 publishing cycle to reflect the creation of SEMI E54.16 and SEMI E54.17.

NOTICE: The document that was previously designated as SEMI E54 (Standard for Sensor/Actuator Network Common Device Model) has been redesignated as SEMI E54.1. Because the Sensor/Actuator Network Standard is the parent document, it has been designated as SEMI E54.

1 Purpose

1.1 This specification provides the structure of SEMI's Sensor/Actuator Network (SAN) standard. It provides the definition for interoperability with respect to SEMI SAN standard-compliant Sensor/Actuator devices.

2 Scope

2.1 This standard specifies how devices interoperate on a network as part of the control system for equipment.

2.2 This specification, which is the root of the SAN standard, defines the relationships of each of the other specifications that are components of the SEMI SAN standard.

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 Referenced Standards

3.1 *SEMI Standards*

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

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

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

3.2 *Other Documents*

ISO 7498[1] — Basic Reference Model for Open Systems Interconnection (OSI)

James Rumbaugh, Michael Blaha, William Premerlani, Frederic Eddy, William Lorensen, *Object-Oriented Modeling and Design*, Englewood Cliffs, New Jersey: Prentice-Hall, 1991.

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

4 Terminology

4.1 *Abbreviations & Acronyms*

4.1.1 *API* — Applications Programming Interface

4.1.2 *CDM — Common Device Model.* The root object application model for devices on the sensor bus.

4.1.3 *DM* — DeviceManager. An object specified in SEMI E54.1 (Standard for Sensor/Actuator Network Common Device Model).

---

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

4.1.4 *IDL* — Interface Definition Language. A programming language-independent notation for specifying interfaces.

4.1.5 *ISO* — International Organization for Standardization.

4.1.6 *NCS* — Network Communication Standard. Provides the specification for mapping the CDM and SDM to specific network technologies. These standards will be incorporated as parts of the SEMI SAN standard.

4.1.7 *OEM* — Original Equipment Manufacturer. They are usually the control system developer, but in any case they are almost always responsible for the control system.

4.1.8 *OMT* — Object Modeling Technique. A methodology developed by James Rumbaugh, et al. for using objects to model systems.

4.1.9 *OSI* — Open System Interconnection. A seven-layer model for communications developed by ISO.

4.1.10 *OSS* — Object Services Standard. SEMI E39.

4.1.11 *SAN* — Sensor/Actuator Network. It is frequently used to reference this standard.

4.1.12 *SDM* — Specific Device Model. This is an application model for a specific type of sensor, actuator, or entity.

4.2 *Definitions*

4.2.1 *application* — for software, this is a working series of computer instructions that provide end user services.

4.2.2 *applications model* — a formal description of the software elements and interactions that perform an end user task.

4.2.3 *entity* — in software engineering, this is something that is recognizable as distinct and particular from the other things that make up a software system or program.

4.2.4 *interface* — in information modeling, it is the boundary between two entities from which information will flow.

4.2.5 *user layer* — in communications, this is a set software function connected to the communication protocol's top layer, usually called the applications layer, from the user's application environment. It typically allows the user application to be protocol-independent.

5 Overview

5.1 Referring to Figure 1, the SAN standards specify a complete communications solution covering all the layers of the OSI reference model. The CDM and SDM together specify a protocol-independent set of application services, attributes, and behavior for a device that must be supported by the user layer as shown in Figure 1. An NCS completes the specification of the user layer by specifying a mapping to a network technology's interface. The NCS must specify its capabilities referenced against the ISO OSI Reference Model. An NCS may utilize an open or commercial solution. In any case, the NCS will provide references to those solution's specifications.

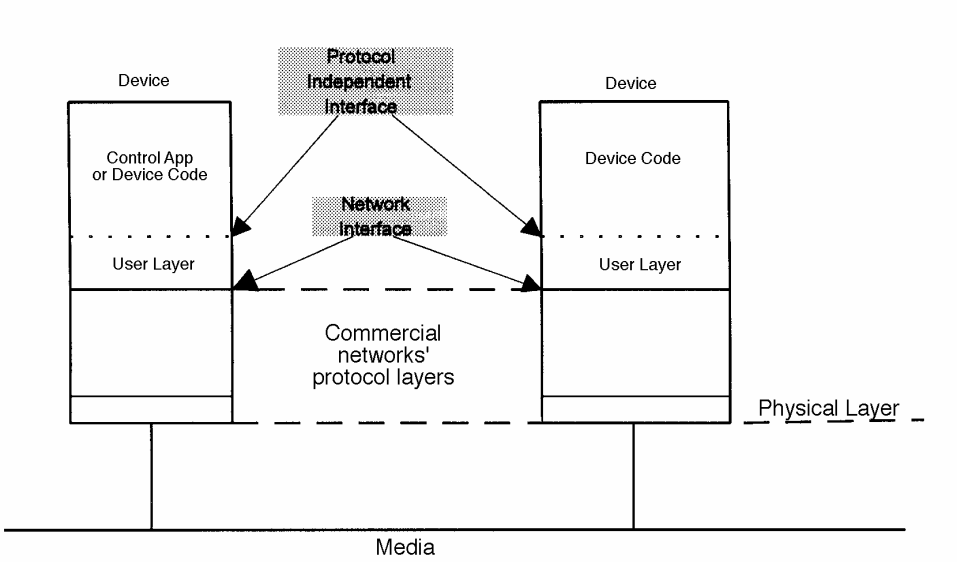

Figure 1
SEMI SAN Protocol Layers

5.2 Generally, the functionality of a software layer is embodied by the definition of the services it provides. Its services are the set of messages it generates and to which it responds, and its subsequent behavior upon receipt of any particular message. The CDM and an SDM provide a complete user layer specification in a platform and language-neutral form (dotted line in Figure 1). The goal being to provide device interoperability at the applications level. The NCS will guarantee interoperability at the network level (i.e., across a single type of network). The SAN standard does not require implementation of the top interface to the user layer, as specified by the CDM and an SDM. The CDM and an SDM provide the information needed to develop an NCS, which defines the interface requirement between the user layer and a specific network technologies communication layer (the solid line labeled as Network Interface). This means that the SAN standard allows user applications, whether from the device or the controller side implementation viewpoint, to interact directly with a network technologies interface.

5.3 *Network Technology Supplier View* of *the SAN Standard*

5.3.1 The CDM and an SDM together specify the messages, behavior, and attributes of a device type that will be visible over the network. In Figure 1, this is the area above the line that indicates the protocol-independent interface. Network suppliers will map this information to the top layer of their protocol. They do this by helping to develop and extend their NCS.

5.3.2 Most network suppliers have an application layer already defined for their protocol. In that case, the user layer as specified in a SEMI SAN NCS is quite thin. In fact, it is just a simple mapping of SDM interface to messages and data (attributes) provided by the network. Thus, an NCS (one for each supported network) specifies a user layer that sits on top of the network supplier's protocol.

5.4 *Device Supplier View of the SAN Standard*

5.4.1 Device suppliers are not required to implement designs illustrated in the specific device model (SDM) specifications. The designs are presented using an object-modeling notation as a means to clearly specify the devices interface. An SDM provides a model of a device, it is not a design for the device. Device suppliers are not expected to provide an object-oriented implementation. However, in order to ensure that a device interface responds as expected, it is necessary for the device supplier to not only provide the support for the services and attributes specified, but to ensure that the code that is implemented on the device behaves as specified by the applications model for a specific device type.

5.5 *Equipment Supplier View of the SAN Standard*

5.5.1 The OEM's have the largest task in adopting the SAN standard. They have to implement applications that use the interfaces defined for the many devices that make up the typical equipment control system. They have to be

familiar with the interfaces of a large number of the SDMs. On the other hand, they will become insulated from the inner workings of many devices, and from the code that they used to write to support those inner workings. In many cases, they may have to implement more than one Network Communication Standard.

5.5.2 During early adoption, OEM's may look to the SAN to act as a replacement for memory mapped discrete wired IO. With this type of adoption, the OEM will not lessen his burden of control code.

5.5.3 As Sensor Bus becomes more broadly adopted, OEM's may take advantage of the distributed processing capability offered by intelligent devices to reduce the complexity of their control software.

5.6 *The SAN Standard Structure*

5.6.1 A SAN employs devices from numerous sup-pliers. The SEMI SAN standard provides a framework to ensure interoperability of these devices within a network. The standard enables device suppliers and equipment control system developers to achieve that end. The SAN standard comprises five specifications.

5.6.2 The first specification is this document, which is the root document and specifies the connection between the other documents of the standard. It also specifies the construction for each of the other documents.

5.6.3 The next specification is the Common Device Model Specification. Each device on the SAN must comply with this specification.

5.6.4 The third specification defines the templates for creating ballots for specific device models and additions to the NCS.

5.6.5 The fourth specification is Specific Device Models standard. These standards contain models for various types of sensors, actuators, and entities. For instance, a model of a mass flow controller, a manometer, and a thermocouple will be specified.

5.6.6 The fifth type of specification is the Network Communication Standards. Each network technology has its own standard as an ancillary specification to the SAN standard.

5.6.7 Applications notes may accompany a number of the standards documents and are intended to guide implementors in the application of the standard.

6 Concepts

6.1 *Minimal device* — The SAN specifications allow simple and complex devices to be built compliant to the standard. This is done by specifying a minimum of required attributes, services, and behaviors in an SDM. The SDMs may also provide optional service definitions that must be followed if extended capabilities are to be built into a device. Devices that do not implement the extended capabilities must respond to extended service or access requests with an appropriate response code rather than failing to respond or issuing an exception or alarm message.

6.2 Many of the SAN NCS specifications rely on commercial technology. A significant number of network technologies are available off-the-shelf. The standards do not prescribe a choice. Once a technology described in an NCS is chosen, the NCS specifies how it is to be applied. The SAN standard is easily extensible for future technologies.

6.3 A device supplier must be able to use the standard to describe a device with little or no ambiguity for the device users. For the user, this means that he can select a device with confidence and that it can be easily integrated with his control system. The user must understand the electrical interface, the capabilities, and messages which a device supports based on the supplier's description of how this standard is supported by the device.

7 Requirements

7.1 *Organization of the SAN Specifications* — Figures 2 and 3 are intended to assist in understanding the document relationships that are specified in the following subsections. In these figures, all specifications that are standards are drawn as rounded rectangles. In Figure 3, the shadowed rectangles indicate the standards that contain requirements for implementing devices that are compliant to this standard.

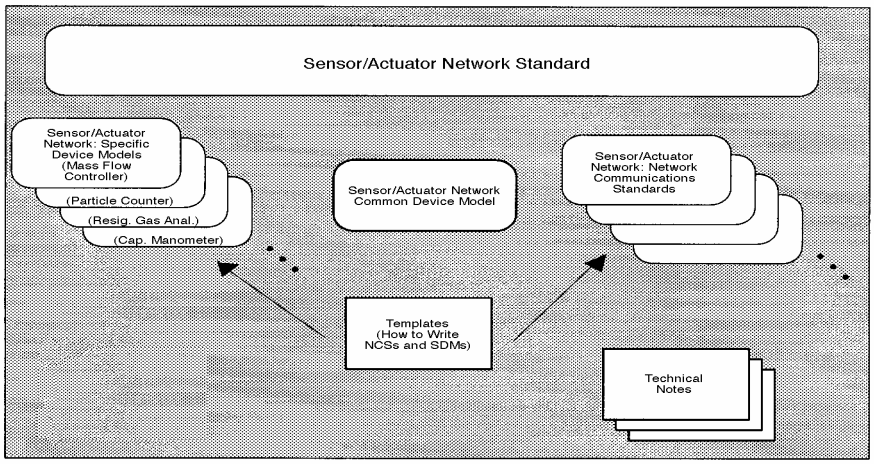

Figure 2
Organization of SAN Specification

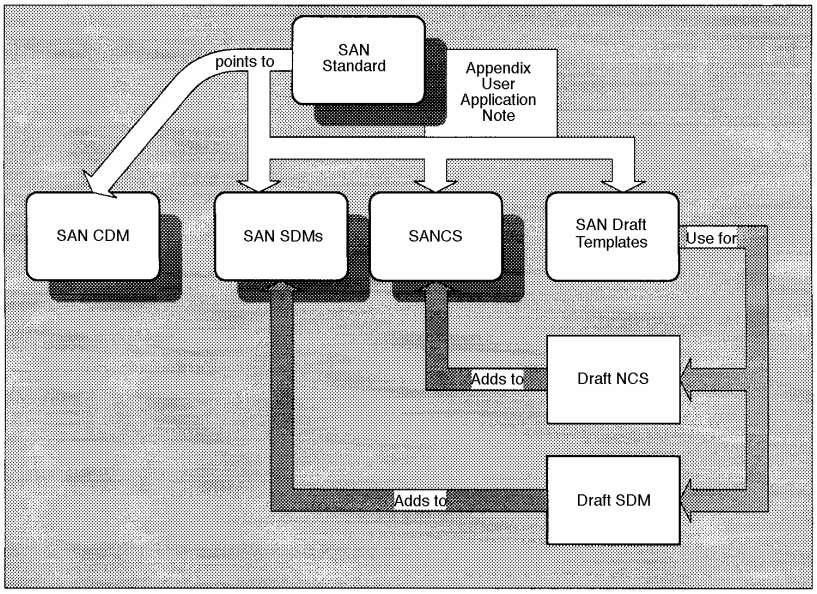

Figure 3
SAN Standard Relationships

7.1.1 The CDM specification is the first ancillary document of the SAN suite.

7.1.2 A second ancillary specification to the SAN standard provides templates that are guides to adding specific device models and network communication mappings to the SAN standard.

7.1.3 Specifications for specific device models for all types of devices are included as additional ancillary specifications of the SEMI SAN standard. Specific device models are added in the order they are developed and passed SEMI's ballot procedure.

7.1.4 Each network technology's user layer specification is included as an ancillary standard (see Figure 3). These *Network Communication Standards (NCS)* provide the mapping for each of the approved specific device models.

Therefore, each NCS is also a specification that grows over time. As new device models are developed and approved, a mapping of the model to a network technology is added to a NCS.

7.1.5 *Auxiliary Information* — Auxiliary information may be included with the standard in the form of appendices or technical notes. Generally, this information is not balloted, but is included as provided by the SEMI Standards Program regulations. Auxiliary information shall always be noted as such when it accompanies a specification or ballot.

7.2 *Interoperability*

7.2.1 *At the Applications Level* — The SDMs are used to derive the definitions for attributes, services, and behavior that are unique to each device type. A model may discuss relationships internal to a device to provide additional meaning to the services and attributes that can be manipulated over the network. Applications written to these models can operate a device independent of the network on which they communicate. The models shall assure that devices of different types can operate cooperatively at the applications level.

7.2.2 *Below the User Layer* — The various SANCS specifications describe the method by which a device communicates over a specific network such that it can meet the requirements of its type. All devices described within an NCS shall operate cooperatively on that specific network technology below the user layer.

7.2.3 The SEMI SAN standard does not address methods for internet-working below the applications layer, as described herein. An application entity may connect to more than one network. This could be typical of the application entities on the equipment controller. While the SAN standard allows that application entities could exist at any network node, it requires no mechanism for a message that begins on one network to be routed to another network.

7.3 *SEMI SAN Compliance* — To fully describe the characteristics of a network capable device, its compliance shall be specified by reference to the SAN ancillary standards. Thus, a device indicates both its type and network technology for its SDM and a NCS.

7.3.1 For a device to be compliant to the SAN standard, it shall implement all of the required attributes, behavior, and services as described in the SAN CDM.

7.3.2 *Type or Applications Model Specification* — The SDM of a device (which is its type, e.g., MFC or thermocouple) shall be denoted by SEMI E54.y, where “y” indicates the SDM ancillary standard.

7.3.3 Network Communication Specification Communication (protocol) compliance specification of a device is denoted by SEMI E54.x, where “x” is the network communications ancillary specification.

RELATED INFORMATION 1
APPLICATION NOTES

NOTICE: The material contained in these Application Notes is not an official part of SEMI E54 and is not intended to modify or supersede the official standard. Rather, these notes are auxiliary information included as reference material for implementers of the standard. The standard should be referred to in all cases. SEMI makes no warranties or representations as to the suitability of the material set forth herein for any particular application. The determination of the suitability of this material for any particular purpose is solely the responsibility of the user.

R1-1 Implementing a SAN Standard-Compliant Device

R1-1.1 This Related Information section describes how the SAN standard is used by a device supplier who wishes to offer SEMI SAN-compliant devices to the market.

R1-1.2 While the SAN standard may appear overwhelming on first viewing, it is actually built from independent components. It is not necessary to understand or 'know' the entire standard. It can be dealt with in small pieces. For device suppliers, only the pieces that deal with their particular device type need to be considered. The specifications to implement a specific device are typically less than thirty pages of text. The standard is structured such that it is extensible while providing the ability to specify a device by reference to the standard.

R1-1.3 *Synopsis of Device Operation* — All devices on a SAN have a similar operational profile. That profile is specified in SEMI E54.1 (Standard for Sensor/Actuator Network Common Device Model) and in the Common Device Model support section of each NCS.

R1-1.3.1 Table R1-1 explains where to find specifications for various device operations.

Table R1-1

Operation	*Spec.*
Power applied	CDM
Enable SAC	CDM
Enable DM	CDM
Enable Communications	NCS
Establish Communication	NCS
Send Events	CDM
Receive messages/answer	CDM
Send messages/accept	CDM

R1-1.4 *Device Type* — Each type of device is specified in the SDM specifications. One SDM is devoted to each device type. An SDM is always a specialization of the CDM. The SDM provides services, behavior, and attributes that are in addition to those specified in the CDM. For example, the SDM of a manometer might specify attributes for current pressure and alarm bands. It might have services needed for performing calibration.

R1-1.5 *Network Technology* — How a network technology is used by the SAN standard is specified in an NCS. The NCS for a technology provides a mapping of the CDM and SDM of a device type to a specific network protocol.

R1-1.5.1 It is only necessary for a device to support an NCS and the section for its type within that NCS. The NCS provides all that is needed to ensure that a device can communicate and interoperate across a given network technology. The point is that from the other side of the wire, the network-independent interface is not visible. That interface is implied by the behavior of the network interface.

R1-1.6 *Network Independence* — Device suppliers have the option of supplying an interface at the protocol-independent applications layer, as indicated in Figure 1. While this requires more implementation work for the device supplier, it speeds the integration of sensors and actuators for the OEM.

R1-1.6.1 In this case, it is still necessary for the device to support the NCS to ensure that their device will interoperate with other devices on the network.

R1-1.6.2 To supply this option, a device supplier would provide a device driver-like interface (software) to the OEM.

R1-1.6.3 Devices that supply a protocol-independent API that is compliant to the SEMI SAN standard provide advantages to both device and control system suppliers. The device supplier can assure that his device receives syntactically correct messages at his network interface layer. Control applications written to the protocol-independent interfaces have the advantage that they would not have to be rewritten in order to change to another network. To achieve this, device suppliers would have to ship software, in the form of 'device drivers', that would be installed on the control supplier's platform.

NOTE 1: Standards for these 'device drivers' may be developed at a later time as they do not currently exist. However, most operating systems used by control systems suppliers support installable device drivers.

R1-1.7 *Recommendations on Optional and Extended Device Capabilities* — It is expected that device suppliers would not compete based on their network interface. Devices traditionally compete based on cost, service, stability, reliability, accuracy and repeatability, and relevant dynamic attributes. From a network perspective, all devices have these common characteristics: periodic reporting, polled reporting, asynchronous event reporting, calibration, health, and diagnostic reporting. Access and control over the common characteristics should be implemented based on the SAN standard. Access and control of the competitive characteristics of devices may be implemented with supplier-specific interface extensions. However, the device supplier is strongly cautioned in the use of these extensions. Heavy use of extensions could be a serious competitive disadvantage unless the supplier is the clear winner in all the competitive characteristics. A locked door could lock one in, it could also lock one out. Device suppliers will usually find that their extensions can be implemented using the optional features of the CDM and SDM.

NOTE 2: Network interfaces could be a competitive factor during the early technology adoption phase.

R1-1.8 *The Controller Side Usage of the SAN Standard* — The control system supplier (in many cases this is the OEM) evolves a different view of SEMI's SAN standard. They are not as concerned with the network technology as with utilizing the applications capabilities as specified by the CDM and SDMs. It is strongly recommended that control applications not be network aware. This may produce some operating overhead, but for most applications, this is a good trade-off to gain flexibility provided by a protocol-independent interface. Many network devices may be supplied without any user layer (device driver) application side software. For these devices, the control system supplier would have to do some system's programming to provide a device driver for the application interface. Over time, it is likely that most devices will be delivered with a device driver. (Similar to the way printers are sold with Windows (TM) device drivers. This is what allows Windows applications to use a printer of the end users choosing.)

R1-1.8.1 The standards as of 1996 do not provide all the specifications needed to uniquely define how device drivers would be installed in the controller side of the API. This may become a future area of standardization, if OEMs decide to invest in the standard's development time.

R1-1.9 *Device Intelligence* — This varies considerably between device types and between devices of the same type. SEMI SAN specifications attempt to allow a range of intelligence to devices without forcing them to be 'highly' intelligent. To this end, the applications models specify a required minimum set of capabilities that must be supported for a device. Optional capabilities generally require more intelligent device operation. Devices that are not capable of supporting these extended capabilities provide a "not implemented" indication if access to those capabilities is requested.

R1-1.9.1 The following subsections describe device capabilities in order of increasing intelligence. A device's I/O direction is defined relative to the control application entity. Therefore, a sensor is an input, even though from the sensor's point of view it supplies output values.

R1-1.9.2 Polled input is the simplest sensor device requirement. All sensing devices should meet this requirement. In this mode of operation, a device only reports its value(s) when requested. This mode potentially provides the biggest burden on network bandwidth. However, it is the capability level that is most compatible with current control system architectures.

R1-1.9.3 Strobed output is the simplest actuator device requirement. All actuating devices should meet this requirement. This device only changes to a new state at the time of receipt of a specific message to do so.

R1-1.9.4 Periodic input is the capability to program the input device so that it sends its value(s) on a scheduled or periodic basis. Network bandwidth burden is reduced because it is not required to send a request each time an update for the input value is needed.

R1-1.9.4.1 This is also very compatible with current control architectures.

R1-1.9.5 Asynchronous input provides a capability for a sensing device to supply its value(s) when an event is detected. These events are usually programmed into the device. Examples of programmable events include threshold crossing of a value, value rate of change, measurement complete.

R1-1.9.5.1 Devices with this capability can significantly reduce network bandwidth burden, because they only need the network when some event has occurred. Equipment control architectures developed before 1990 may not be able to utilize this type of capability.

R1-1.9.6 Programmed output provides the capability for actuating devices to follow a programmable sequence of operations. For instance, an analog output could be given a ramp profile to follow as it moves between set points.

R1-1.9.6.1 This is another capability which can greatly reduce network bandwidth requirements. But, it is also a capability that 1990 vintage and earlier control systems may not be able to utilize easily.

R1-1.9.7 Devices with distributed processing capabilities are able to support features such as monitoring their health, providing diagnostics, and programmable preprocessing of data.

R1-1.9.7.1 These capabilities can have an impact on network burden, but more importantly may improve equipment reliability and maintainability. Again, this is a capability that is not readily utilized by control architectures developed prior to 1990.

R1-1.9.8 Devices with distributed control capabilities are capable of being programmed to create peer relationships with other devices on the network. Within these relationships, it is possible for one or more of the peers to control or regulate a subsystem within the equipment. For instance, a butterfly valve and manometer with these capabilities could be programmed to hold a particular pressure.

RELATED INFORMATION 2
INDEX FOR SENSOR/ACTUATOR STANDARDS

NOTICE: This related information is not an official part of SEMI E54 and is not intended to modify or supercede the official standard. After approval of this document, publication of this Related Information was authorized by the committee chairs solely to aid in identifying the relationships between the current Sensor Bus standards and documents in process. Determination of the suitability of the material is solely the responsibility of the user.

R2-1 Scope

R2-1.1 The table below is based on Figure 1 in SEMI E54.1, and is intended to show the relationships between standards that are currently published, as well as describe documents in process in SEMI Standards that, if approved, will be added to SEMI E54.

R2-1.2 There are five categories:

Main Document

Templates

Specific Device Model

Common Device Model

Network Communication Standards

R2-1.3 Standard titles with a designation number are published documents, and descriptors without designation numbers are documents in process.

Table R2-1 Sensor/Actuator Network (SAN) Standards

<table>
<tr><td colspan="3">SEMI E54, Sensor/Actuator Network (SAN) Standard</td></tr>
<tr><td colspan="3">Templates (How to Write Network Communications Standards (NCS) and Specific Device Models (SDM))</td></tr>
<tr><td>Specific Device Models</td><td>Common Device Models</td><td>Network Communications Standards</td></tr>
<tr><td rowspan="3">Mass Flow Controller SDM</td><td rowspan="3">SEMI E54.1, Standard for Sensor/Actuator Network Common Device Model</td><td>SEMI E54.4, Standard for SAN Communications for DeviceNet</td></tr>
<tr><td>SEMI E54.5, Standard for SAN Communications for the Smart Distributed System (SDS)</td></tr>
<tr><td>SEMI E54.6, Standard for SAN Communications for LONWorks</td></tr>
</table>

NOTICE: SEMI makes no warranties or representations as to the suitability of the standard set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

SEMI E54.1-1000
STANDARD FOR SENSOR/ACTUATOR NETWORK COMMON DEVICE MODEL

This standard 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 July 14, 2000. Initially available at www.semi.org August 2000; to be published October 2000. Originally published in 1996; previously published February 1999.

NOTE: This document was previously designated SEMI E54. Because this document is part of a suite of documents, its designation has been reassigned for ease of reference.

1 Purpose

This standard defines a model comprised of device objects which are common to all devices on a semiconductor equipment sensor/actuator communications network.

2 Scope

2.1 This document describes common device structure and behavior (i.e., the minimum data structure and behavior all devices must support to operate on the network). These devices may range from simple sensors and actuators through hosts, masters, or controllers.

2.2 The model specified in this document is used in conjunction with a sensor/actuator network specific device model which describes the data structure and behavior characteristic of the specific device. Together, these two models are sufficient to completely describe a device as it appears from the network interface.

2.3 This standard, together with a sensor/actuator network interoperability guideline, a sensor/actuator network communication specification, and one or more specific device model specifications, form a complete interoperability specification. The sensor/actuator network document architecture is shown in Figure 1.

2.4 To comply with this standard, a device must implement and support instances of the objects, object attributes, object services, and object behaviors identified in this document, unless explicitly stated otherwise.

3 Limitations

3.1 This standard is a companion to a suite of sensor/actuator network communication network specifications. Therefore, using portions of this standard that relate to network communications necessarily requires an understanding of the associated network specification.

3.2 As this document is a standard for a *common* device model, it does not contain any information (e.g., object, attribute, services, or behavioral descriptions) that relates to a specific device or device type.

3.3 While the standards depicted in Figure 1 are sufficient to completely describe a device as it appears from the network, they do not fully describe behavior of the device which is not visible from the network. Some of the behavior detail within standard objects is intentionally left to the manufacturer to define. This allows flexibility for product differentiation and creates room for technology evolution. Manufacturer-specific objects may be defined by the manufacturer as appropriate, but are by definition outside the scope of this standard.

3.4 This standard is compatible, but not compliant, with SEMI E39. This means that although this standard does not require compliance with SEMI E39, it is extensible such that implementations may be developed that are fully compliant with both standards. Note that the concepts and terminology of this standard are compatible with those of SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and thus are not applied to the Common Device Model.

4 Referenced Documents

4.1 *SEMI Documents*

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

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

4.2 *Other Documents*

ISO 7498[1] — Basic Reference Model for Open Systems Interconnection

David Harel, "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming 8, 1987

James Rumbaugh, Michael Blaha, William Premerlani, Frederic Eddy, William Lorensen, Object-Oriented Modeling and Design, Englewood Cliffs, New Jersey: Prentice-Hall, 1991

1 International Organization for Standardization, 1 rue de Varembe, Case postale 56, CH-1211, Geneva 20, Switzerland. Available in the US from American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY10036.

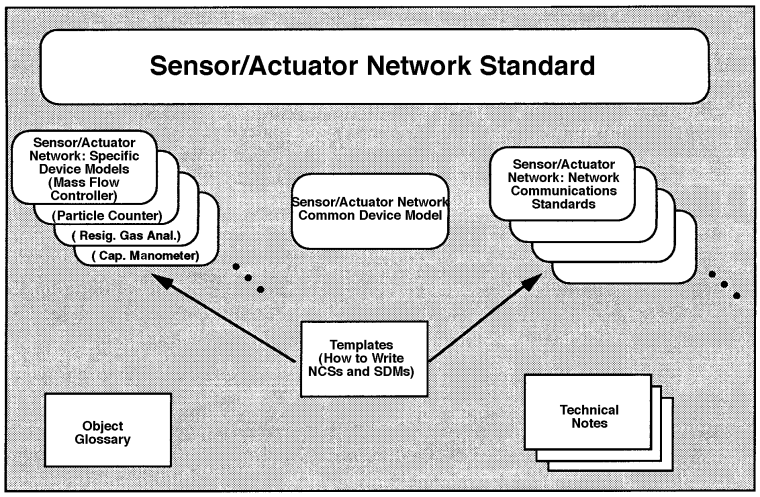

Figure 1
Sensor/Actuator Network-Related Documents

5 Terminology

Terminology may be reproduced here which is defined in other SEMI documents.

5.1 *Device Component Terminology*

5.1.1 *attribute* — Externally visible information concerning an object.

5.1.2 *behavior* — A specification of how an object acts. Actions result from different events the object detects, such as receiving service request, detecting internal faults, or elapsing timers.

5.1.3 *class* — A specific type or classification of objects.

5.1.4 *device* — A tangible thing consisting of: (1) at least one sensor and/or actuator and/or controller, (2) a communications controller which supports a single point of access to a network as specified in this document, and (3) interconnection and management hardware and software that provides for the consolidation of (1) and (2) into a system that has the capability to comply with the specification detailed in this document. Examples of devices are given in Figure 2.

5.1.5 *device model* — An abstraction of a device for the purpose of understanding it before building it or using it.

5.1.6 *embedded object* — An embedded object is similar in functionality or purpose to the object in which it is embedded, or supports the functionality of the object in which it is embedded. The embedding construct is utilized solely for purposes of documentation structure and understanding. As such, it does not imply any direct relationship, inheritance, similarity in structure, or connectivity in addressing scheme between the embedded object and the object in which it is embedded.

5.1.7 *instance* — A specific and real occurrence of an object.

5.1.8 *manufacturer* — In the context of this document, this refers to the manufacturer of the device.

5.1.9 *object* — An entity with a specific set of data and behaviors. Objects may be physical or conceptual. An object may be described in terms of its attributes, services it provides, and behavior it exhibits.

5.1.10 *service* — A function offered or supported by an object. A service consists of a sequence of service primitives, each described by a list of parameters. A service excludes definition of message structure and protocol.

5.1.11 *state diagram* — A means of representing state transitions, where the boxes represent states and the arrows represent transitions between states.

5.2 *Data Type Terminology* — Unless otherwise noted, data type terminology defined in SEMI E39 will be used in this document. The following terminology will also be used:

5.2.1 *Boolean (BOOL)* — A binary bit representing 0 and 1 corresponding to FALSE and TRUE or DISABLE and ENABLE respectively.

5.2.2 *byte* — A string of eight adjacent bits, interpreted as a unit and often representing a character.

5.2.3 *character* — A text symbol, letter, digit, or mark used to represent, control, or organize information that is one byte in length.

5.2.4 *character string* — A text string.

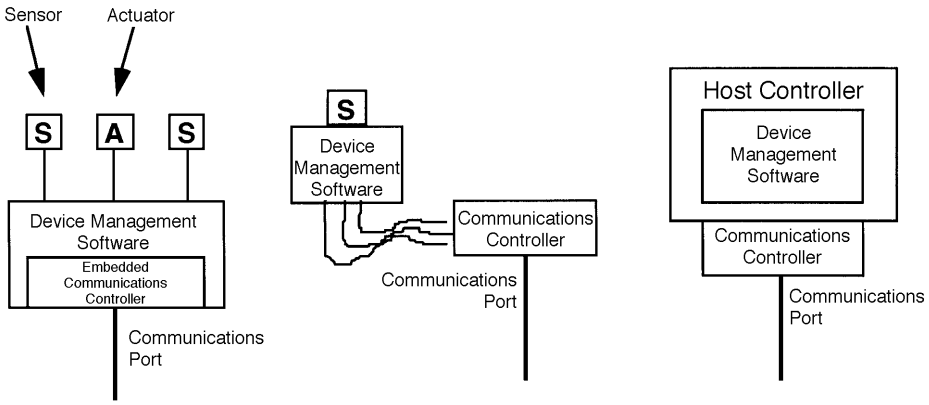

Figure 2
Examples of Devices

5.2.5 *data type* — An unsigned short integer formatted as an enumerated byte to specify attribute data format. The intended use of this attribute type is in cases where an attribute, or set of attributes, may be defined, allowing for more than one level of support (e.g., INT or REAL). The following values are defined:

0=INT

1=REAL

2=USINT

3=SINT

4=DINT

5=LINT

6=UINT

7=UDINT

8=ULINT

9=LREAL

10–99=reserved for CDM

100–199=reserved for SDMs

200–255=manufacturer-specified

5.2.6 *data units* — An unsigned integer enumerated to specify attribute data units. The intended use of this attribute type is in cases where an attribute, or set of attributes, may be defined, allowing for more than one unit's context. The values are defined in Appendix 1 of this document.

5.2.7 *date* — A data structure of four bytes used to represent a calendar date. Table 1 defines the format of the date data type.

Table 1 Date Format

Data #	*Description*	*Range*
0–1	Year	Unsigned Integer < 65,536
2	Month	Unsigned Integer (range of 1 to 12)
3	Day	Unsigned Integer (range of 1 to 31)

5.2.8 *Double Integer (DINT)* — An integer, four bytes long, in the range -2^{31} to $2^{31}-1$.

5.2.9 *enumerated byte* — A byte with assigned meaning to the values 0 through 255. May take on one of a limited set of possible values.

5.2.10 *full scale range* — The defined 100% value of an attribute in its assigned units. This value is not necessarily the maximum value for the attribute. As an example, the indicated flow attribute value may attain 120% of the full scale range.

5.2.11 *Last Valid Value (LVV)* — The most recent value successfully assigned to an attribute.

5.2.12 *Long Integer (LINT)* — An integer, eight bytes long, in the range -2^{63} to $2^{63}-1$.

5.2.13 *Long Real (LREAL)* — A double floating point number, 8 bytes long, as defined by IEEE 754.

5.2.14 *nibble* — A string of four adjacent binary bits.

5.2.15 *null character* — A byte with a value of zero.

5.2.16 *Real (REAL)* — A floating point number, 4 bytes long, as defined by IEEE 754.

5.2.17 *Short Integer (SINT)* — An integer, one byte long, in the range -128 to 127.

5.2.18 *Signed Integer (INT)* — An integer, 2 bytes long, in the range -32768 to 32767.

5.2.19 *text string* — A string of one byte characters.

5.2.20 *Unsigned Double Integer (UDINT)* — An unsigned integer, four bytes long, in the range 0 to 2^{32}-1.

5.2.21 *Unsigned Integer (UINT)* — An integer, 2 bytes long, in the range 0 to 65535.

5.2.22 *Unsigned Long Integer (ULINT)* — An unsigned integer, eight bytes long, in the range 0 to 2^{64}-1.

5.2.23 *Unsigned Short Integer (USINT)* — An integer, 1 byte long, in the range 0 to 255.

6 Conventions

6.1 *Harel State Model* — This document uses the Harel State Chart notation to describe the dynamic behavior of the objects defined. 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.

Transition tables (referred to in this document as "state transition matrices") are provided in conjunction with the state diagrams to explicitly describe the nature of each state transition. Each transition table contains columns for Transition #, Current State, Trigger, New State, Action(s). The "trigger" (column 3) for the transition occurs while in the "current" state. The "actions" (column 5) include a combination of: 1) actions taken upon exit of the current state, 2) actions taken upon entry of the new state, and 3) actions taken which are most closely associated with the transition. No differentiation is made.

6.2 *OMT Object Information Model* — The object models are presented using the Object Modeling Technique developed by Rumbaugh, James, et al, in "Object-Oriented Modeling and Design," Prentice Hall, Englewood Cliffs, NJ, ©1991. Overviews of this notation are provided in an appendix of SEMI E39.

6.3 *Object Attribute Representation* — The object information models for standardized objects will be supported by an attribute definition table with the following column headings:

Attribute	*Definition*	*Access*	*Rqmt*	*Form*
The formal text name of the attribute.	Description of the information contained.	RO or RW	Y or N	(see below)

The access column uses RO (Read Only) or RW (Read and Write) to indicate the access that users of Object Services have to the attribute. Note that, in this document, the access column is used only to indicate user access via the network; no indication is made as to local access as this level of specification is considered to be implementation-specific. Thus, in the context of this document, a RW attribute is a network-settable attribute (i.e., an attribute whose value can be altered from the network), while a RO attribute is a non-network-settable attribute.

A 'Y' or 'N' in the requirement (Rqmt) column indicates if this attribute must be supported in order to meet fundamental compliance for the service.

The Form column is used to indicate the format of the attribute.

6.4 *Service Message Representation*

6.4.1 *Service Resource Definition* — The service resource definition table defines the specific set of messages for a given service group, as shown in the following table:

Service	*Type*	*Description*
Message name	N or R	The intent of the service.

Type can be either Notification (N) or Request (R). Notification messages are initiated by the service provider. No response is expected. Request messages are initiated by the service user. Request messages ask for data or for an operation to be performed. Request messages expect a specific response (no presumption on the message content).

6.4.2 *Service Parameter Dictionary* — Each parameter should relate to either attributes from the object model or events of the dynamic model (Harel State Chart). All parameters to the services are listed in a single table or dictionary. The column headings for this parameter dictionary are as follows:

Parameter	*Form*	*Description*
Parameter X	Data type	A parameter

A row is provided in the table for each parameter of the service. The first column contains the name of the parameter. This is followed by columns describing the form and the contents of the parameter.

The form column is used to indicate the type of data contained in the parameter and the possible values it may take on.

The description column describes the meaning of the parameter and interrelationships with other parameters.

6.4.2.1 *Service Message Definition* — There is a table showing the parameter detail for each service in which parameters are explicitly specified. The column headings for the service detail are as follows:

Parameter	*Req/Ind*	*Rsp/Cnf*	*Description*
Parameter X	(see below)	(see below)	A description of the service.

The columns labeled Req/Ind and Rsp/Cnf link the parameters to the direction of the message. The message sent by the initiator is called the "Request" (Req). The receiver terms the message the "Indication" (Ind). The receiver may then send a "Response" (Rsp), which the original sender terms the "Confirmation" (Cnf).

The request (Req/Ind) and response (Rsp/Cnf) 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 another parameter.

"U" **User-defined 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 Device High Level Structure

The high level object view of a device aggregation is shown in Figure 3. A device is depicted as consisting of a sensor/actuator/controller (SAC) object, a device manager (DM) object, and at least one active element object (i.e., sensor or actuator or controller object). Each of these objects by definition has attributes, services, and behavior. Note that the "Device" object is depicted in Figure 3 only for purposes of illustrating a high level view of the device and its component objects; in the context of this document, the "Device" object is not addressable, does not have addressable attributes or accessible services, and has no behavior defined.

This document defines in detail only the DM object. The SAC, Sensor, Actuator, and Controller objects are defined here only in terms of characteristics common to all devices. A complete definition of a SAC object includes an appropriate sensor/actuator network specific device model. This companion model could also complete the definition of sensor, actuator, and controller objects as appropriate to specify a device model.

7.1 *General Requirements* — Objects are defined in terms of their object name and instance identifier. Identifiers for all objects described in this document are summarized in Table 2. Note that these identifiers may be used for remote interrogation of an object instance via the sensor/actuator network (see appropriate sensor/actuator communications specification). Also note that, in Table 2, many of the objects specified in this document have exactly one instantiation per device.

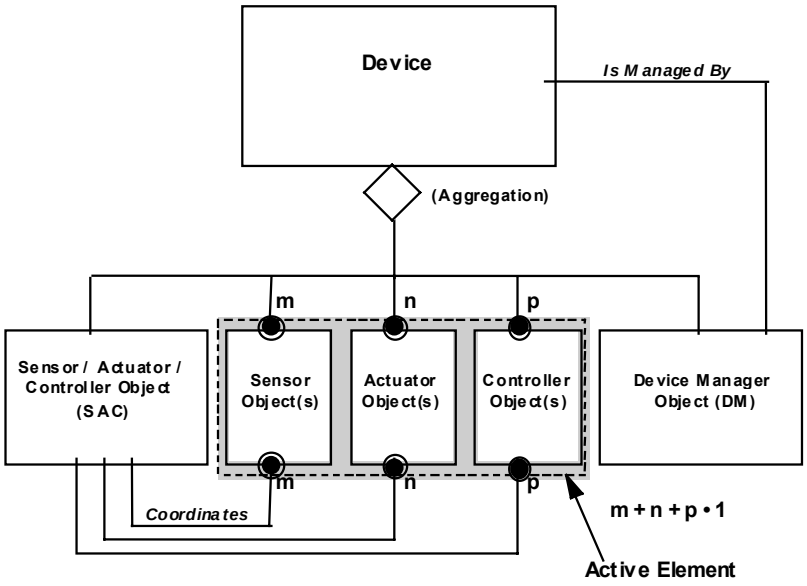

Figure 3
High Level Object View of a Device

Table 2 Device Objects and Identifiers

Object Name	*Object Identifier (tag)*	*Support Required in a Device*	*Comment*
Sensor/Actuator/Controller	SacI0*	Yes	Only one instance per device allowed.
Sensor	SenIn**	No	Zero or more instances per device allowed.***
Actuator	ActIn**	No	Zero or more instances per device allowed.***
Controller	CntIn**	No	Zero or more instances per device allowed.***
Device Manager	DmI0*	Yes	Only one instance per device allowed.

* Only one object instance per device; identifier uses "I0" to specify "instance zero."

** "In" is used to indicate the instance number of the object; "n" is a non-negative integer.

*** The specification of the number of sensor, actuator, and/or controller object instances allowed per device may be further constrained by the appropriate sensor/actuator network specification.

The objects described in this document collectively define a device's capabilities, including how it has been configured for network interoperability. The information in the attributes of these object instances must be accessible over the network and stored at the device.

Character strings described in this document have a prescribed maximum length. If the contents of the string are shorter than the prescribed length, the string must be terminated with the null character: a byte whose value is 00h. Note that this specification of null termination does not indicate that a sensor/actuator network protocol implementation communicating a character string over a network must send the null termination; the presentation of character string data over a network is sensor/actuator network communication protocol-specific.

7.2 *Sensor/Actuator/Controller (SAC) Object* — The SAC object is the device component responsible for coordinating the interaction of the device with the sensory/actuation/control environment. A view of the SAC object is shown in Figure 4. The SAC object coordinates operation of one or more sensor, actuator, and/or control object instances that collectively form the sensory/actuation/control portion of the device, so as to enable desired device behavior. For example, it could coordinate the operation of the device sensor, actuator, and/or control elements to enable device level data reporting or actuation, alarming detection and servicing, status reporting, device self-testing, device shutdown, etc. The number of sensor, actuator, and/or controller object instances allowed per device may be specified by the appropriate sensor/actuator network specific device model. An operating device shall contain exactly one instance of a SAC object.

The SAC object has embedded in it other objects that address specific tasks associated with coordinating the interaction of the device with the sensor/actuation/control environment. These objects are listed in Table 3. Note that although only one instance of the SAC object is allowed per device, a device can have multiple instances of embedded objects. These objects are described in Section 7.2.4.

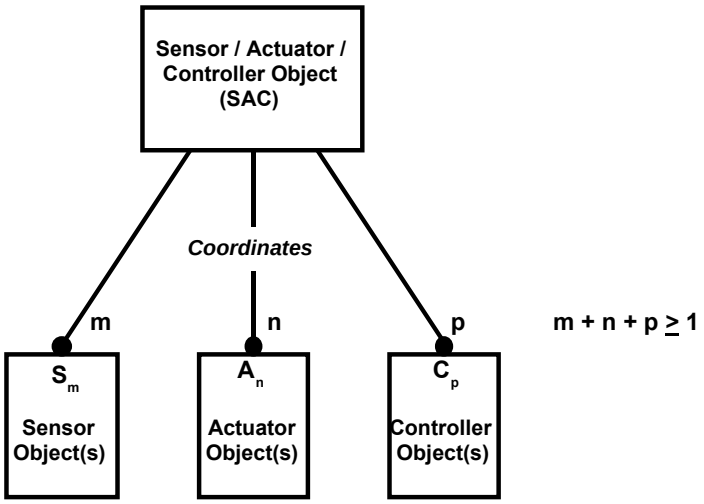

Figure 4
Detailed View of the Sensor/Actuator/Controller Object