W

SEMI E32.1-0997
SECS-II SUPPORT FOR MATERIAL MOVEMENT

1 Introduction

This document contains the necessary information to implement the Material Movement Management standard using the SEMI E5 SECS-II communications protocol.

2 Service to Message Mapping

The table below lists each message service defined in the Material Movement standard with the SECS-II message which should be used to perform that service.

Service	*SECS-II Msg*	*Comments*
TRJobCreate	S4,F19[1]	
TRJobCreate Response	S4,F20	
TRJobCommand	S4,F21	
TRJobCommand Response	S4,F22	
TRJobAlert	S4,F23	
TRJobEvent	S6,F11[2]	Corresponding messages to enable reporting, create and link reports, etc. are also assumed.
GetAttribute	S1,F19[3]	
GetAttribute Response	S1,F20[3]	
HOReady	S4,F27	
HOCommand	S4,F29	
HOCommand Response	S4,F31	
HOVerify	S4,F33	
HOVerify Response	S4,F33	
HOCancelReady	S4,F35	
HOCancelReady Response	S4,F37	
HOHalt	S4,F39	
HOHalt Response	S4,F41	

[1] If the S4,F19 message is multi-block, it will require the S4,F25/F26 multi-block inquire/grant transaction.

[2] TRJobEvent is an unconfirmed service. While the required S6,F12 reply must be transferred by the protocol layer, there is no action required on the part of the host. Therefore, it shall be discarded rather than delivered to the application which employs this service.

[3] The GetAttributes request and response will be removed from E32 and E32.1 in 1998. This message transaction is replaced by GetAttr, S14F1/F2. See E39 and E39.1 for details.

3 Parameter to Data Item/Parameter Mapping

In the table below, each parameter used in the SEMI E32 messaging services is listed along with its corresponding SECS-II data item. In some cases, a parameter will correspond to a SECS-II list. These cases are noted.

SEMI E32 defines a number of message parameters as conditional. This means that under certain conditions, these parameters do not need to be included in the message. In SEMI E5, this is done by means of a zero length data item, or if the parameter is a list, a zero length list. Please refer to SEMI E5, “Zero Length Items and Lists,” Sections 6.2.1 and 6.3.1.

Parameter	*SECS-II Data Item*
AtomicSpec	List Structure
AttrData	ATTRDATA
AttributeID	ATTRID
CmdParameter	List Structure
CmdParmName	CPNAME
CmdParmValue	CPVAL
EQName	EQNAME
ErrorCode	ERRCODE
ErrorText	ERRTEXT
HOAck	HOACK
HOCancelAck	HOCANCELACK
HOCmdName	HOCMDNAME
HOHaltAck	HOHALTACK
HOStatus	List Structure
ObjectID	OBJID
ObjType	OBJTYPE
Status	List Structure
TRAck	TRACK
TRAtomicID	TRATOMICID
TRAutoStart	TRAUTOSTART
TRCmdName	TRCMDNAME
TRDirection	TRDIR
TREventData	List Structure (See S6,F11 in SEMI E5.)
TREventID	CEID
TRJobID	TRJOBID
TRJobMS	TRJOBMS
TRJobName	TRJOBNAME
TRLink	TRLINK
TRLocation	TRLOCATION
TRObjName	TROBJNAME
TRObjType	TROBJTYPE
TRPartner	TRPTNR
TRPartnerPort	TRPTPORT
TRPort	TRPORT
TRRecipe	TRRCP
TRRole	TRROLE
TRStatus	List Structure
TRType	TRTYPE

4 Collection Event List

The following collection events must be available for reporting from the equipment via S6,F11:

Committed To Transfer
Waiting for StartHandoff
Transfer Job Paused
Transfer Job Resumed
Atomic Transfer Started
Atomic Transfer Complete

Application Note — SECS-I Equipment-to-Equipment Link

The direct SECS-I link between equipment poses a small problem. That is, it is not clear which partner should be the “SECS-I host” and which the “SECS-I equipment.” As a general rule, if one partner is always the primary transfer partner, it should be configured as the SECS-I host. However, this is not a requirement, since in some cases the choice of which is the primary partner may depend upon the situation. Therefore, where SECS-I is used to link the transfer partners, each partner should be user-configurable as either “host” or “equipment.”

SEMI E36-0704
SEMICONDUCTOR EQUIPMENT MANUFACTURING INFORMATION TAGGING SPECIFICATION

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on December 18, 1998. Initially available at www.semi.org February 1999; to be published June 1999. Originally published 1995; previous published revision February 1998.

This document replaces SEMI E36-0298 in its entirety.

NOTICE: The designation of this standard was updated during the 0704 publishing cycle to reflect the creation of SEMI E36.1.

NOTE: The Document Type Definitions, schemas, and documentation defined in other parts of Semiconductor Equipment Manufacturing Information Tagging may be used freely without permission or payment of royalties or other licensing charge, provided this source and/or its associated specifications are cited.

1 Purpose

The purpose of Semiconductor Equipment Manufacturing Information Tagging is to define a markup philosophy, a markup framework, and an information markup which is rich enough to facilitate:

- Electronic interchange and distribution of information;
- Quality hardcopy printing and screen display of information;
- Consistent high-precision online searching; and
- Reuse and repurposing of information for such applications as integrating source material into training material and online support applications.

The intent of this document is to create an information interchange specification, not an authoring specification or an electronic presentation specification. Semiconductor Equipment Manufacturing Information Tagging will define the markup necessary for exchanging documents electronically, for facilitating retrieval of information content, and for validating information interchanges.

2 Scope

2.1 *Information Scope* - Semiconductor Equipment Manufacturing Information Tagging is intended to cover the types of technical and commercial information typically needed to support the selection, installation, use, and maintenance of semiconductor manufacturing equipment.

The following list gives examples of information types within scope; this list is not exhaustive.

- *Operations Manuals/Guides* — typically used by equipment operators at customer sites to perform the tasks for which the equipment was designed.
- *Installation Manuals* — typically used by supplier and customer engineers to install the equipment at the customer site.
- *Maintenance Manuals* — typically used by trained maintenance technicians at customer sites to perform repair and upkeep of equipment.
- *Maintenance Schedules* — typically used by site coordinators and customer facility managers to schedule and document normal preventive maintenance.
- *Spares/Parts Lists* — typically used by on-site field service engineers and customers to maintain a sufficient stocking level to ensure prompt delivery of parts.
- *Repair/Troubleshooting Manuals* — typically used by equipment operators and qualified maintenance technicians to identify and correct equipment problems.
- *Release Notes* — typically used by operators, technicians and system administrators to identify changes in software configuration.
- *Training Manuals* — typically used by supplier training personnel and customers to train operators and maintenance technicians.

Additional information on potential use and re-use of these information types in electronic form can be found in the Auxiliary Information for this specification.

2.2 *Content of SEMI E36* - Semiconductor Equipment Manufacturing Information Tagging will be issued in three parts:

- this base document,

- associated information models such as SGML and XML Document Type Definitions (DTDs) and schema, and
- the documentation for these information models, including at least a comprehensive Tag Library.

Semiconductor Equipment Manufacturing Information Tagging does **not** include implementation advice, suggestions, or notes on how to implement the models described here. This material will be presented in the form of Auxiliary Information associated with Semiconductor Equipment Manufacturing Information Tagging.

Semiconductor Equipment Manufacturing Information Tagging by design does not include tutorials on SGML, XML, how to use the defined elements or modules, or how to create document content.

2.3 *SGML and XML* – Any documents marked up according to Semiconductor Equipment Manufacturing Information Tagging will be "valid" XML documents: that is, "well-formed" XML documents that have an associated document type declaration and that comply with the constraints expressed therein. Since valid XML documents are by definition type-valid SGML documents, such documents will be valid SGML documents as well.

Initially, information models will be maintained in both XML and SGML, with all changes made in parallel. SEMI will issue a dual set of models whenever a change is made, and both will always be (within the limits of their syntax) the same. Although parallel maintenance is unfortunate, it is intended to make it easier for members to create documents using current SGML or XML tools. As Web-SGML implementations and alternative XML schema syntax become more common, modules may be maintained in other syntactic forms.

2.4 *Examples of SEMI Content Specifications* - There are other SEMI standards and specifications that regulate the content of a document. Semiconductor Equipment Manufacturing Information Tagging is expressly not in conflict with any of them, since it specifies only how to identify the content present in the documents, and in some contexts, how to electronically structure the content. Semiconductor Equipment Manufacturing Information Tagging does not address what information is to be included. Examples of SEMI content-related or content guideline-related standards and specifications can be seen in section 7.

3 Referenced Documents

Semiconductor Equipment Manufacturing Information Tagging references the following information standards, which should be referred to in their entirety for full details:

3.1 *ISO Standards*[1]

International Standard for SGML; ISO-8879-1986 as modified by Amendment 1, of 1988

HyTime Standard ISO 10744 revision 2, which is referenced for hypertext linking

3.2 *Other references*[2]

The Extensible Markup Language (XML)V 1.0 Recommendation of the World Wide Web Consortium (W3C)

The draft W3C specifications for Extensible Linking Language (XLL).

4 Terminology

4.1 *Attribute* - An attribute is a markup construct that resides within the start tag of an element to provide additional information about the element. (Note: the start tag is the tag that is placed directly before the element in the datastream and determines where the element begins.)

4.2 *Conforming Authoring Application* - A conforming authoring application must be able to produce conforming documents. There is no requirement that all internal work in the application be performed in the markup described by Semiconductor Equipment Manufacturing Information Tagging, only that the end product, exported for interchange, must be a conforming document.

4.3 *Conforming Document* - A conforming document is one that meets the markup rules and specific information models defined in Semiconductor Equipment Manufacturing Information Tagging for the class of document. This specification comprises three parts: this base document, a set of information models, and the documentation for those models. A conforming document must conform to all three parts.

A conforming document created by one application must be interchangeable with a conforming application without significant information loss.

NOTE: Explicit rendering instructions such as font family and type size are not considered to be significant information.

---

[1] *ISO Central Secretariat, 1, rue de Varembé, Case postale 56, CH-1211 Genevè 20, Switzerland. http://www.iso.ch*

[2] *World Wide Web Consortium, Massachusetts Institute of Technology Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. http://www.w3.org*

4.4 *Conforming Element* - A conforming element is one that uses the names and markup rules defined in Semiconductor Equipment Manufacturing Information Tagging for the class of element. A conforming element created by a conforming application must be interchangeable with another conforming application without information loss.

4.5 *Conforming Rendering Application* - Rendering (presentation) is the process and means by which the elements in a document shall be made visually or otherwise understandable to the end user, for example, printing a document on paper, browsing a series of elements on screen, or stating a series of elements through voice synthesis. A conforming rendering application must be able to print or display conforming documents without significant loss of information. The presentation should be consistent with the intent of the originating system.

A conforming rendering application may not impose structural requirements that are not required by the DTD or other document models as stated in the second part of Semiconductor Equipment Manufacturing Information Tagging. For example, a rendering engine may not require an element which is optional in the models or require an attribute that is implied. A rendering application may require that the rules in this Base Specification and those in the Tag Library are followed.

For example, if the DTDs say that an element is optional, a Rendering Application may not require that it be present. If the Tag Library says that the contents of an element must be some particular information, the Rendering Application may require that it be so. More specifically, if the <description> element is optional, a Rendering Application may not fail to process a document because it does not contain <description>s. If the Tag Library says that the contents of a "datechanged" attribute is a date in YYYYMMDD format, a Rendering Application may fail if the attribute does not contain a valid date.

4.6 *Document* - A logical assembly of elements, that, if correctly ordered, is a means of transmitting an internally consistent piece of information. A document is made up of one or more elements which must all be contained in one "root" element.

4.7 *Document Class* - A document class consists of all of the documents or information fragments that follow a single set of markup rules, including the same elements and element relationships. In general, a class of documents can be considered to be documents that are nearly the same in structure and have similar types of content. A document class is usually (but not necessarily) defined by a structural model (such as a DTD or Schema) that governs its existence.

4.8 *Document Exchange* - For SEMI E36, "document exchange" is synonymous with "document interchange" (see definition).

4.9 *Document Interchange* - Document interchange entails passing one or more elements, usually as a document, from one conforming system to one or more other systems (also known as "exchange").

4.10 *Document Model* - "Document model" is a term for a set of structural rules that describe the legal markup for a particular class of documents. DTDs and Schema are types of document models.

4.11 *Document Type Definition (DTD)* - A DTD is a document model or list of markup rules, which may be used in both SGML and XML. As specified in the SGML standard, ISO 8879, a DTD is:

> "Rules, determined by an application, that apply SGML to the markup of documents of a particular type. A document type definition includes a formal specification, expressed in a document type declaration, of the element types, element relationships and attributes and references that can be represented by markup. It thereby defines the vocabulary of the markup for which SGML defines the syntax." (ISO 8879 4.105)

4.12 *Element* - An element is one named, contiguous piece of information in the information stream. Usually the data within an element has a semantic or structural relationship as well, for example a person's surname, an error code, a pointer to external data, a paragraph, or a procedure that contains steps. In the previous version of this specification (i.e., in SEMI E36-95) an element is also referred to as an "information component."

An element is:

- A uniquely addressable unit of information within the document; and/or
- An identifier to an addressable unit of information outside the document.

4.13 *Markup* - For Semiconductor Equipment Manufacturing Information Tagging, markup is defined as additional data characters that are added to data to provide information about the data and make the data more useable. The markup described by Semiconductor Equipment Manufacturing Information Tagging is internal markup, that is, markup which resides in the same data stream as the data — in specific SGML and XML markup.

4.14 *Nonconforming Document* - A non-conforming document is one that does not use the markup properly

or follow the markup rules defined in Semiconductor Equipment Manufacturing Information Tagging.

4.15 *SGML* - SGML is "Standard Generalized Markup Language." The Standard Generalized Markup Language is an International Organization for Standardization (ISO) standard. The ISO Reference Number is ISO 8879:1986(E). SGML is a meta-language for constructing markup. See also XML (Extensible Markup Language).

4.16 *Tag* - A tag is a string of characters delimited by rules set out in the SGML standard (and used in XML and HTML). Tags are placed in the datastream to indicate where an element begins and ends. In Semiconductor Equipment Manufacturing Information Tagging, tags match as closely as possible the names of the elements in English.

4.17 *XLL (Extensible Linking Language)* - When released, XLL is intended to be the syntax and semantics for both simple hypertext linking (as exemplified by HTML) and more complex linking, providing such functionality as multi-way links, multi-ended links, and addressing from an external document based on structure. A W3C Working Group is in the process of producing a draft specification, which uses many of the concepts from HyTime (Hypermedia/Time-based Document Representation Language ISO/IEC 10744) and the TEI (Text Encoding Initiative) extended pointers.

4.18 *XML (Extensible Markup Language)* - XML is a W3C Recommendation that describes a subset of SGML syntax and functionality intended for use on the World Wide Web. XML is, by design, simpler, less complex, and easier to implement than SGML. XML is also, by design, far less flexible than SGML because it includes only the most commonly implemented SGML features.

XML is a family of Recommendations that includes a linking specification (XLL); it may in the future include other specifications such as a stylesheet specification (XSL). The XLL linking specification will provide the syntax for hypertext links defined in Semiconductor Equipment Manufacturing Information Tagging. The XSL stylesheet language will be neither mandated nor prohibited in SEMI E36-98.

5 Design and Principles

5.1 *Information Markup* - For purposes of interchange, information will be marked-up according to the conventions established in Semiconductor Equipment Manufacturing Information Tagging. The defined markup will:

- Identify the start and end of elements, which may contain named information content or structural components;
- Provide information concerning the identified elements; and
- Specify connections among elements and between elements and external objects.

Applications will be able to utilize this markup to:

- Select, limit, and organize specific elements for reuse or interchange;
- Start and stop behavior, such as print formatting, complex screen display, and the actuation of programs;
- Limit retrieval (for increased precision);
- Generate navigation aids such as indexes and tables of contents; and
- Validate document structure or content.

5.2 *SGML Standard and XML Family of Recommendations* - The intent of Semiconductor Equipment Manufacturing Information Tagging is to define the rules and markup that can be used to create XML data (which is, by definition, SGML data) for maximum interchangeability and movement across both internets and intranets. Therefore all markup and rules will be designed to be fully conforming with the W3C XML Recommendation. It is also the intent to describe hypertext linking in conformance with the XLL (Extensible Linking Language), as much as it is known, and to migrate in the direction of that recommendation as the recommendation is finished.

Since the intent of Semiconductor Equipment Manufacturing Information Tagging is to create data that is both valid XML and type-valid SGML:

- The DTDs, which will be written in both SGML and XML syntax, will conform to the XML subset of SGML by not using features and syntax that XML prohibits such as exceptions and #CONREF attributes;
- Tagged data produced according to Semiconductor Equipment Manufacturing Information Tagging will be valid XML. Early implementers using SGML authoring or conversion tools should check with their tool suppliers to make sure those tools will work with their document without modification.

Names of elements, attributes, and entities will be used consistently within the SEMI E36 module set. While some elements may have different content models from

module to module, and in fact that is the intent of the parameter entity framework, all name definitions and use will be consistent throughout the SEMI E36 DTD suite. This means that namespace issues, such as those addressed by the XML Namespace Recommendation (currently under development), are not relevant within the scope of SEMI E36. It is likely that mechanisms such as XML Namespaces will be used to incorporate parts of the SEMI E36 markup set in other applications; but this use is outside the scope of SEMI E36 itself.

5.3 *Specification for Interchange* - Semiconductor Equipment Manufacturing Information Tagging defines markup optimized for interchange, not for information development (authoring) nor for information presentation (rendering). Thus, some of the capabilities and functionality that might be desired for authoring or rendering will not be included in the markup rules, by design. For example, few elements may be required and the sequence of elements will not be tightly constrained.

It is anticipated that more specific authoring and rendering models may be developed by other parties as part of conforming applications. All conforming interchange, however, will explicitly follow the markup and structural rules defined in Semiconductor Equipment Manufacturing Information Tagging.

5.4 *General Modeling Principles*

- Document (information) models will be constructed that name the elements, describe the relationships among these elements, and impose certain markup or content constraints. The information markup will be controlled by these models.
- All models will be made available by SEMI and distributed and balloted as additional parts of Semiconductor Equipment Manufacturing Information Tagging. Models will be described in the second part of Semiconductor Equipment Manufacturing Information Tagging rather than included in the base document directly because the models are assumed to be dynamic documents that will grow and change as requirements alter. The data models and the Tag Library may be balloted separately without requiring a re-balloting of this part of the specification.
- The models will initially be in the form of SGML and XML Document Type Definitions (DTDs) and may later include other XML-defining schema.
- There will be no all-inclusive general information models; each DTD or other collection of rules will model specific elements or specific document types.
- Markup will be primarily designed according to the nature of the information being described; such considerations as data volume and ease of implementation will only be considered secondarily.
- DTDs will be constructed in a modular fashion, with content-related or structurally-related modules. The purpose of such modularization is to reuse existing information content models as much as possible. New element structures should be created only when new content is analyzed.

5.5 *Information Modeling*

5.5.1 *Types of Elements* - SGML and XML elements can identify many types of information. There are situations in which each of the following element types are appropriate, and many situations in which several could be used. In designing structures and tagging documents, elements should be designed using the following hierarchy of types, from most- to least-preferred. Only if the information cannot be satisfactorily identified using one type of element should those below it in the following list be considered.

- Content Elements — those that identify the information content, such as Warning, Procedural Step, Part Number, Installation Manual, or Bill of Materials.
- Structural Elements — those that identify the structural role of a part of a document, such as Title, Paragraph, or List Item.
- Pointer Elements — those that point to other locations, either in this document or another, such as Cross-reference, Footnote Reference, or URL.
- Format Elements — those that identify portions of a document that are to be formatted differently from surrounding text and for which there is no Content, Structural, or Pointing reason that they should be so formatted, for example a Bold element. Format elements are to be used as a last resort; use of a content, structure, or pointer element with a format-related attribute is preferable to use of a format element.

5.5.2 *Naming Elements* - Markup in the DTDs will be named consistently across contexts. That is, the same element name will be used for an element that can occur in many places with the same function, regardless of the location. For example, there is only one element for Title, not many separate elements for Section Title, Table Title, Figure Title, etc.

In XML, names are case-sensitive. In the interest of keeping compliant with XML, all names will be case-

sensitive. (In SGML, names may not be case-sensitive since such names may be case-folded to upper case at any time.) All element tag names should be in lower case. Entity names and attribute names may be mixed case, lower case, or upper case to match common usage of the name involved. If there is no such consideration, lower case should be used. Attribute values should be in lower case unless there is a good reason to make them another case, such as a list of state or country codes which would naturally appear in upper case.

5.5.3 *Recursive Structures* - In cases where an element may contain one or more of the same elements within itself, there are several styles of modeling the solution:

- Use recursive structures. (In recursion, an element would be allowed to contain itself, thus a <section> could contain one or more <section>s, which could themselves contain <section>s, etc.);
- Create an explicit hierarchy by naming the top level and lower levels. (Thus a <section> could contain one or more <subsections>, which could themselves contain one or more <sub-subsections>, etc.); or
- Create an explicit hierarchy by numbering the top level as "1", and subsequent levels as "2", "3", etc. (Thus a <section1> could contain one or more <section2>s, which could themselves contain one or more <section3>s, etc.).

In such cases, recursion should be used to model an element because it is optimal for authoring, editing, and data reuse; in such an environment an editor can move a section from one level to another without needing to modify the tagging inside the section.

5.5.4 *Content Modeling* - To create SGML elements that are maximally re-useable and compatible with XML, there are several SGML modeling capabilities that are prohibited in SEMI E36:

- RCDATA and CDATA declared content (character data in which elements and entities cannot be recognized) are prohibited.
- The "AND" connector (which specifies that all the elements so connected are required but that order is irrelevant) is prohibited.
- Inclusions and Exclusions to content models are prohibited.
- Pernicious mixed element content is prohibited. This means that if an element model contains character data it must allow character data at any place in the model. In practice this means that any mixed content model groups should be repeatable "| " (or) groups.

In addition, the following rules also will apply:

- Each element must be declared separately;
- No inline comments or empty comments will be permitted; only full comment declarations may be used;
- The replacement text for external parameter entities must be well-formed, as per the XML specification declarations;
- The entity types CDATA, PI, and bracketed text are prohibited;
- No Marked Sections are allowed in the DTDs; and
- External entities must include both a SYSTEM and a PUBLIC identifier.

5.5.5 *Attributes* - The attribute list for each element must be declared separately. Attribute values must not contain references to external entities. If a Declared Value is not a specified list, only the following Declared Value keywords may be used:

- CDATA (character data);
- ID (a unique identifier for the element within its current document);
- IDREF and IDREFS (one or more pointer to the IDs of other elements within the same document);
- ENTITY or ENTITIES (the name of one or more defined entities); and
- NMTOKEN or NMTOKENS (one or more words that consist entirely of name characters).

On all attribute declarations, attribute values specified as defaults must be literals. Only the following Default Values keywords may be used:

- #REQUIRED (must be present);
- #IMPLIED (optional, may be implied by the application); and
- #FIXED (set, not able to be changed).

5.6 *Formatting from Element Context versus from Attributes* - The use of formatting attributes to indicate preferred rendition in display or print should be limited to situations in which no other mechanism is available. That is, if the desired formatting can be accomplished by use of the element name alone, or based on the context in which the element appears, no formatting attributes should be used. Elements should only carry formatting attributes such as "type-size" if there is no other mechanism to pass this information or if an author or editor, rather than the output system, must control the presentation directly. Formatting attributes, when used,

should be optional and considered advisory by the rendering agent.

5.7 *Data Markup Rules* - To create SGML documents that are maximally re-useable and compatible with XML, the following restrictions are placed upon the tagged instances produced in accordance with Semiconductor Equipment Manufacturing Information Tagging:

- The characters "<" and "&" must always be escaped as "<" and "&" respectively.
- No Marked Sections will be allowed in data content.
- The replacement text of general text entities must be well-formed, for example, an element that starts within an entity must also end within it.
- All attributes should be present (even default attributes) and attribute values should be well-formed, that is with the name, equal sign, and quotation marks present.
- Processing Instructions are prohibited with the exception of the XML declaration, for example: <? XML version=˝1.0˝ RMD=' INTERNAL' >.

6 How Semiconductor Equipment Manufacturing Information Tagging Should Be Used

6.1 *Conformance* - A conforming document is one that meets the markup rules defined for the document class in Semiconductor Equipment Manufacturing Information Tagging. A non-conforming document is one that does not use the markup properly or follow the markup rules defined in Semiconductor Equipment Manufacturing Information Tagging. A conforming document created by an application must be interchangeable with another conforming application without significant information loss. A conforming application may also deliver the markup rules (information models) for the document or information class being interchanged to another system.

Conforming applications are those that create, render, or work with conforming documents. Such applications may include browsers and online rendering software, print and composition engines, editors and authoring systems, search and retrieval engines, etc. In addition to strictly conforming applications, it is expected that many applications will accept conforming documents as input, converting them to the internal format of the software.

6.1.1 *Conforming Authoring Application* - A conforming authoring application must be able to produce conforming information. An application may state that it is conformant with Semiconductor Equipment Manufacturing Information Tagging if it has the ability to:

- Create and manipulate conforming documents;
- Create and manipulate one or more classes of conforming documents;
- Import conforming documents into the application and output matching conforming documents; or
- Turn non-conforming documents into conforming documents.

6.1.2 *Conforming Rendering Application* - There is no rendering specification stated or required by Semiconductor Equipment Manufacturing Information Tagging at this time. This specification does not define style conventions or endorse any particular language for the interchange of stylistic information. That said, a conforming rendering application is one that can render (print, display, pronounce, etc.) conforming documents without significant loss of information. The presentation should be consistent with the intent of the originating system, but need not be an exact, facsimile-reproduction match.

To call itself conforming, a rendering application must be able to take any conforming document, which is by definition legal according to this base specification and the models defined and explained in other parts of the specification, and render that document in some visual or non-visual fashion. A conforming rendering application may not impose additional requirements or constraints such as content requirements. For example, a rendering engine may not require the presence of an element that is optional in the models or require a more restrictive range of values for an attribute than the models state.

6.1.3 *Validation* - Documents tagged according to Semiconductor Equipment Manufacturing Information Tagging must parse cleanly according to their SGML or XML models and must follow the rules stated in this base document and the Tag Library and other documentation. Tagging validation must be accomplished through the use of validating SGML or XML parsers, in standalone form or as incorporated into other software packages. (Note: Validation of some of the rules described in this base document and the Tag Library documentation cannot be accomplished using only an XML or SGML parser. This does not mean that such rules can be ignored, merely that other validation techniques may be required.) SEMI E36 rules include both rules that can be checked with software and content rules that can only be verified intellectually; a conforming SEMI E36 document must conform to both types of rules.

6.2 *Maintenance of Semiconductor Equipment Manufacturing Information Tagging* - This base portion of the SEMI E36 specification is intended to be debated and balloted once and change little over time. Additional parts of the specification will contain the actual information models that describe the markup to be used. New information models may be proposed at any time. Such models will be distributed, discussed, and balloted individually or in small groups, in the form of one or more SEMI E36 modules, as they are created. Modules that are approved will be appended to Semiconductor Equipment Manufacturing Information Tagging. The documentation will be changed and balloted in parallel, to match any changes in the models.

6.2.1 *Module Library* - A growing module library containing all approved modules will be maintained, adding modules as additional subject areas and document types are defined. Once a module has been approved, it will be provided as part of Semiconductor Equipment Manufacturing Information Tagging. Place-holding modules may be introduced into the library that are designated as "Temporary", for use only until a formal analysis and balloting process can provide a more appropriate module. Initially modules will be XML and SGML DTDs and DTD fragments. As XML alternative modeling syntaxes such as schema become supported by software, these may be used to supplement SEMI E36 information models.

6.2.2 *Modules* - Each module will be given a unique short name (to be used as a file name), a version number, and a unique public identifier written according to the rules of ISO 8879 and incorporating the version number.

File names will be all lower case, constructed by concatenating the following, in order:

- A 3- to 8-character descriptive name;
- A hyphen;
- The file type as "xml" or "sgml";
- A period; and
- A 3-character descriptive extension that defines the type of file (such as "dec", "sgm", etc.) These extensions will be defined in the Tag Library.

Each module will contain:

- An initial comment which names the module, states the purpose of the modules, types the modules as Temporary or Implemented, and states how the module should be integrated with other modules;
- A growing change history comment that describes (in sedimentary order; that is, the most recent at the top) all modifications made to the model. Each item in the change history should be numbered, dated, provide a description of the change, give a brief rationale, and name the changing agent.
- Whatever syntax is employed, internal comments should be used throughout the module to name and describe the elements and provide information that may not be stated directly in the syntax used.

6.2.3 *Consolidated Tag Library* - A single, consolidated Tag Library will be maintained which will describe the elements, attributes, and entities defined in all the modules. Each description will contain sufficient detail to indicate the meaning and correct usage of the data markup. The library (or changes to the library) will be reissued whenever a module is added or modified in a way that affects any of the elements, attributes, or entities.

Modules should use elements (tags) that are already in the Tag Library wherever the definition of the element matches the need. If the definition of an existing tag does not meet the need, a new tag should be created.

A diagram of the structure of each element will be included in the Tag Library, except for elements that are empty, contain only character data, or contain a simple "or" group in which all contents may appear at any point in the model.

For each element, the Tag Library will contain:

- Element Title, consisting of its Tag and Name;
- Definition in clear, plain English, including description of intended use and application when appropriate;
- Remarks (if needed);
- Related Elements (if needed);
- For each Source, or module in which the element is defined:
    - Attributes Associated with the Element (if any);
    - The model of the Element in SGML syntax and narrative form;
    - Presentation information (if needed);
    - Formatted examples of the element (if needed);
    - Tagged examples of the elements (if needed); and
    - The Name of the Source, or module in which the element is defined.

For each Attribute, the Tag Library will contain:

- Attribute Title, consisting of the short form of the attribute which appears in the documents, and the attribute Name;
- Attribute usage, describing how that attribute is used;
- For each possible value of an attribute with a specified set of values, the meaning or implications of each value; and
- The default value of the attribute.

6.3 *Creation of More Specific Authoring or Rendering Models* - The DTDs and models described in Semiconductor Equipment Manufacturing Information Tagging will not be designed for optimal use in document creation. To create and display documents, data creators may wish to create more enforcing authoring models based on this markup. Such authoring models may:

- Require the presence of specific elements;
- Specify the sequence of elements; and/or
- Include additional elements for internal use (which would need to be removed before interchange). For example, such information might include elements for tracking the internal authoring and validation processes or for company-confidential information.

Similarly, it is likely that viewing, print rendition, search and retrieval, or other display applications will create display-only models based on Semiconductor Equipment Manufacturing Information Tagging. To load display tools, it is quite likely that the specified models will be transformed into a form that is optimized for the particular search and retrieval or display system. Such display models might:

- Resequence the data;
- Add additional markup such as “container” tags around structures to be considered one element for formatting purposes;
- Transform the form of some elements; or
- Remove parts of the information that are not relevant to a particular application.

7 Related Documents - SEMI Content Standards and Specifications

As described in section 2.4, the following list contains examples of SEMI content-related or content guideline-related standards and specifications.

SEMI C16 — Guide for Precision Reporting/Data Traceability

SEMI D9 — Definitions for Flat Panel Display Substrates

SEMI D13 — Terms and Definitions for FPD Color Filter Assemblies

SEMI E6 — Facilities Interface Specifications Guideline and Format

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

SEMI E29 — Standard Terminology for the Calibration of Mass Flow Controllers and Mass Flow Meters

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

SEMI E35 — Cost of Ownership for Semiconductor Manufacturing Equipment Metrics

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

SEMI G57 — Guideline for Standardization of Leadframe Terminology

SEMI M10 — Standard Nomenclature for Identification of Structures and Features Seen on Gallium Arsenide Wafers

SEMI P22 — Guidelines for Photomask Defect Classification and Size Definition

SEMI P29 — Guideline for Description of Characteristics Specific to Halftone/Attenuated Phase Shift Masks and Mask Blanks

SEMI P30 — Practice for Catalog Publication of Critical Dimension Measurement Scanning Electron Microscopes (CD-SEM)

SEMI S1 — Safety Guideline for Visual Hazard Alerts

SEMI S7 — Safety Guidelines for Environmental, Safety, and Health (ESH) Evaluation of Semiconductor Manufacturing Equipment

SEMI S13 — Safety Guidelines for Operation and Maintenance Manuals Used with Semiconductor Manufacturing Equipment

SEMI T6 — Procedure and Format for Reporting of Test Results by Electronic Data Interchange (EDI)

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 E36.1-0704
SPECIFICATION FOR THE SEMICONDUCTOR EQUIPMENT MANUFACTURING INFORMATION TAGGING STANDARD E36

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

1 Purpose

1.1 The purpose of this specification provides the consolidated Tag Library required in implementing the SEMI E36-0699 standard.

2 Scope

2.1 The specification contains two levels of compliance necessary in implementing the electronic documentation standard. The following two components make up the compliance levels.

2.1.1 *Level One Identity* — Allows for content to be uniquely identified, most notably the Item ID and Edition ID attributes.

2.1.2 *Level Two Type and Metadata* — Association of semantic detail with items. Important information about the item that allows us to interpret the item correctly. Compliance is established for this level by ensuring that the item uses a standard tag name and is equipped with a small standard set of metadata properties as defined by the specification.

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 See http://www.w3.org/TR/2000/CR-RDF-schema-20000327/ - RDF specification.

3.2 See http://www.w3.org/TR/2000/CR-rdf-schema-20000327/#s2.3.1 for the related RDF schema.

3.3 See http://dublincore.org/schemas/xmls/simpledc20020312.xsd for the related XML schema.

3.4 See http://dublincore.org/ - Dublin Core Metadata Initiative (DCMI).

3.5 For a list of Dublin Core metadata elements, see http://dublincore.org/documents/dces/.

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

4 Terminology

4.1 None.

5 Consolidated Tag Library

5.1 The consolidated tag library is required to implement SEMI E36-0699. The first layer, identity, is supported by the element “E36Item”, which identifies the tagged information as complying with SEMI E36-0699. The second layer, metadata, provides the tags most important for managing and retrieving the documentation.

5.2 Level two compliance is achieved by the inclusion of one or more of the metadata elements described below. For elements inserted for Level 2 compliance, all attributes are considered optional, unless otherwise indicated.

5.3 Attributes associated with the elements appearing below are multi-valued; no restrictions on cardinality are expressed or implied. If an attribute has multiple values, the attributes statement must be repeated, once for each value.

5.4 *Element Name — E36Item*

5.4.1 *Element Description* — This element serves as the basic super type from which, by default, all of the elements in the SEMI E36 tag set are derived. An E36Item element can contain arbitrary content. The E36Item element has the following attributes:

- *ItemID* — Globally unique URI[1] used to distinguish each E36Item. This attribute partitions the space of E36Items into equivalence classes, each of which represents an individual item of information.
- *EditionID* — Globally unique URI[2] used to distinguish each member of an equivalence class, as defined by the ItemID. This attribute is used to control variation within an equivalence class.

5.4.2 *Metadata* — The following namespaces are defined for E36Item metadata:

5.4.2.1 *DC* — representing the Dublin Core[3] namespace.

5.4.2.2 *RDF* — representing the Resource Description Framework[4] namespace.

5.4.2.3 *e36base* — representing the E36 basic namespace.

5.4.3 In the DC namespace, standard interpretations of the following items of metadata are utilized:

5.4.3.1 Title

5.4.3.2 Publisher

5.4.3.3 Subject

5.4.3.4 Description

5.4.3.5 Format

5.4.3.6 Source

5.4.3.7 Language

5.4.3.8 Rights

5.4.4 In the e36base namespace, the following items of metadata are utilized:

5.4.4.1 *ReleaseDate* — This item is a Date describing the moment when the content contained in the tag becomes valid. (mandatory)

5.4.4.2 *ExpirationDate* — This item is a Date describing the moment when the content contained in the tag becomes invalid.

5.4.4.3 *VersionString* — This is a string to be used in assigning a human-readable version name to an individual edition of an E36Item.

5.4.5 In the RDF namespace, we make use of the following metadata item:

5.4.5.1 Type (see: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/#s2.3.1).

5.5 *Element Name — Manual*

---

1 We recommend using GUID-based URI for this function, although any scheme which generates URI which are guaranteed to be globally unique is acceptable.

2 See footnote number one.

3 See http://dublincore.org/ for information relating to the Dublin Core Metadata Initiative (DCMI). See http://dublincore.org/schemas/xmls/simpledc20020312.xsd for the related XML schema. For a list of Dublin Core metadata elements, see http://dublincore.org/documents/dces

4 See http://www.w3.org/TR/2000/CR-RDF-schema-20000327/ for the latest RDF specification.

5.5.1 *Element Description* — The Manual element is used to define the documents in a document manual set. The content of the Manual element is not interpreted.

5.5.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Manual element supports the following additional metadata, all in the e36base namespace:

5.5.2.1 *ManualType*

5.5.2.1.1 *OperationsManual* — Manual type typically used by equipment operators at customer sites to perform the tasks for which the equipment was designed.

5.5.2.1.2 *InstallationManual* — Manual type typically used by supplier and customer engineers to install the equipment at the customer site.

5.5.2.1.3 *MaintenanceManual* — Manual type typically used by trained maintenance technicians at customer sites to perform repair and upkeep of equipment.

5.5.2.1.4 *MaintenanceSchedule* — Manual type typically used by site coordinators and customer facility managers to schedule and document normal preventive maintenance.

5.5.2.1.5 *PartsList* — Manual type typically used by on-site field service engineers and customers to identify parts to order and to maintain a sufficient stocking level to ensure prompt delivery of parts.

5.5.2.1.6 *TroubleshootingManual* — Manual type typically used by equipment operators and qualified maintenance technicians to identify and correct equipment problems.

5.5.2.1.7 *ReleaseNotes* — Manual type typically used by operators, technicians and system administrators to identify changes in software configuration.

5.5.2.1.8 *TrainingManual* — Manual type typically used by supplier training personnel and customers to train operators and maintenance technicians.

5.6 *Element Name — Configuration*

5.6.1 *Element Description* — The Configuration element is used to describe a unique piece of semiconductor manufacturing equipment. The content of the Configuration element is not interpreted.

5.6.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Configuration element supports the following additional metadata, all in the e36base namespace:

5.6.2.1 *WaferSize* — Specific to the wafer size for the specific semiconductor manufacturing equipment set; may be 200mm, 300mm, or “Other”. (mandatory)

5.6.2.2 *Mainframe* — Designation of mainframe type for the specific semiconductor manufacturing equipment set.

5.6.2.3 *TechnologyArea* — The area in which the semiconductor manufacturing equipment set is located in the semiconductor manufacturing process.

5.6.2.3.1 Lithography

5.6.2.3.2 Etch

5.6.2.3.3 Planarization

5.6.2.3.4 Diffusion

5.6.2.3.5 Thin Film

5.6.2.3.6 Implant

5.6.2.3.7 Analytical

5.6.2.3.8 Sort

5.6.2.3.9 Assembly/Test

5.6.2.4 *Process* — Free text field allowable for process type used in the specific semiconductor manufacturing equipment.

5.6.2.5 *Equipment* — Supplier-defined identifier for a specific type of semiconductor manufacturing equipment.

5.7 *Element Name — Recipe*

5.7.1 *Element Description* — The Recipe element defines a semiconductor manufacturing equipment recipe or set of instructions pertaining to the process, test or qualification on the specified semiconductor manufacturing equipment. The content of the Recipe element is not interpreted.

5.7.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Recipe element supports the following additional metadata, all in the e36base namespace:

5.7.2.1 *RecipeType*

5.7.2.1.1 *Process* — Use this value if the recipe is for the processing of wafers on the specific semiconductor manufacturing equipment.

5.7.2.1.2 *Test* — Use this value if the recipe is for testing to validate an adjustment or functional operation of a component or semiconductor manufacturing equipment.

5.7.2.1.3 *Qualification* — Use this value if the recipe is for the qualification of equipment parameters within the specified range limits on the specified semiconductor manufacturing equipment.

5.8 *Element Name — Procedure*

5.8.1 *Element Description* — The Procedure element describes a sequence of equipment related operations, typically performed by operations or service personnel. The content of a Procedure element is interpreted as information describing the sequence of operations or instructions represented by the procedure.

5.8.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Procedure element supports the following additional metadata, all in the e36base namespace:

5.8.2.1 *ProcedureType*

5.8.2.1.1 *Maintenance* — To include procedures but not limited to calibrations, qualifications, retrofits, cleaning, repair, preventive, corrective, diagnostic, troubleshooting, and software around semiconductor manufacturing equipment.

5.8.2.1.2 *Installation* — To include procedures but not limited to, setup, startup, and install around semiconductor manufacturing equipment.

5.8.2.1.3 *Operation* — To include procedures around the operation of semiconductor manufacturing equipment.

5.8.2.1.4 *ToolRemoval* — To include procedures but not limited to, shipping, decontamination, and de-install around semiconductor manufacturing equipment.

5.8.2.1.5 *Safety* — To include procedures concerning safety during the handling, assembly, installation, use, and removal of semiconductor manufacturing equipment.

5.8.2.1.6 *Usage* — To include procedures of usage of a non-technical nature.

5.8.2.2 *ScheduleType* — For periodic procedures, describes the frequency pattern exhibited by the procedure. On specific semiconductor manufacturing equipment, one of:

5.8.2.2.1 Wafer Based.

5.8.2.2.2 CalendarTimeBased.

5.8.2.2.3 ProcessTimeBased.

5.8.2.3 *ScheduleUnits* — For periodic procedures, describes the nominal frequency for performing the procedure.

5.8.2.3.1 When scheduleType is Wafer Based, this is the number of wafers.

5.8.2.3.2 When scheduleType is CalendarTimeBased, this is the number of elapsed days.

5.8.2.3.3 When scheduleType is ProcessTimeBased, this is the number of elapsed process hours based on the specific semiconductor manufacturing equipment.

5.9 *Element Name — Component*

5.9.1 *Element Description* — The Component element represents a physical or logical part of semiconductor manufacturing equipment. The content of a Component element is not interpreted.

5.9.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Component element supports the following additional metadata, all in the e36base namespace:

5.9.2.1 *SupplierPartNumber* — Contains a supplier-defined part number for an identified system or component on specific semiconductor manufacturing equipment.

5.9.2.2 *OEMPartNumber* — Original Equipment Manufacturer Part number system or component on a semiconductor manufacturing equipment set.

5.9.2.3 *PartName* — Contains the Supplier designated part name for a system or component on specific semiconductor manufacturing equipment set.

5.9.2.4 *OEMName* — Original Equipment Manufacturer Company Name for an identified system or component on a specific semiconductor manufacturing equipment set.

5.10 *Element Name — Narrative*

5.10.1 *Element Description* — The Narrative element represents arbitrary content related to the specific semiconductor manufacturing equipment. The content of a Narrative element is interpreted as text and graphics.

5.10.2 *Metadata* — None.

5.11 *Element Name — ErrorCode*

5.11.1 *Element Description* — The ErrorCode element represents an equipment-generated error code on a specific equipment set. The content of an ErrorCode is interpreted as text describing the error code. This text should be applicable to the context of the end-user.

5.11.2 *Metadata* — In addition to the metadata available from the E36Item tag, the ErrorCode element supports the following additional metadata, all in the e36base namespace:

5.11.2.1 *ErrorNumber* — An alphanumeric string defined by the supplier to identify the error on a specific semiconductor manufacturing equipment set.

5.12 *Element Name — Media*

5.12.1 *Element Description* — The Media element represents a generic linked media resource. The content of a Media element is not interpreted.

5.12.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Media element supports the following additional metadata, all in the e36base namespace:

5.12.2.1 *MediaURL* — A well-formed URL describing the location of the file containing the media element data.

5.12.2.2 *Caption* — A short description of the information contained in the media element, suitable for presentation to the end user.

5.12.2.3 *MediaCategory* — Captures the basic purpose of the media element.

5.12.2.3.1 *ElectricalSchematic* — A schematic diagram representing electrical application, theory, and/or process for the specified semiconductor manufacturing equipment.

5.12.2.3.2 *MechanicalDrawing* — A drawing or diagram showing the mechanical structure of a part, number of associated parts, assembly of parts and/or associated parts for the specified semiconductor manufacturing equipment or represent of this equipment in part or in its entirety.

5.12.2.3.3 *PlumbingSchematic* — A schematic diagram representing Gas/Pneumatic/Vacuum/Hydraulic application, theory, and/or process for the specified semiconductor manufacturing equipment.

5.12.2.3.4 *FacilityLayout* — A diagram noting specific installation requirements (space, electrical, plumbing, environmental) for the specified semiconductor manufacturing equipment.

5.12.2.3.5 *EquipmentTemplate* — A pre-designed, pre-cut guide to aid in determining space requirements (footprint) for the specified semiconductor manufacturing equipment.

5.12.2.3.6 *PartLocator* — Actual assembly or component part location and number for the specified semiconductor manufacturing equipment.

5.12.2.3.7 *FlowChart* — Instructional steps included in operations and maintenance or troubleshooting on a specific semiconductor manufacturing equipment set.

5.12.2.3.8 *ProcessChart* — Characterization data for the process on a specific semiconductor manufacturing equipment set.

5.12.2.3.9 *Diagram* — To include block and engineering diagrams for the specified semiconductor manufacturing equipment.

5.12.2.3.10 *ScreenCapture* — Includes the duplication of a displayed computer screen saved to a file.

5.12.2.3.11 *Photograph* — Includes any image bearing the likeness of an object obtained by photographic capture of that object.

5.13 *Element Name — SafetyItem*

5.13.1 *Element Description* — The SafetyItem element represents an item of safety related information per SEMI S13-0298, SEMI S2, ANSI Z400.1, ANSI/RIA R15.06, CDRH-21, CE Mark Machinery Directive, EN/IEC 60204, or NFPA 79-1997. The content of the SafetyItem element is the expression of the information.

5.13.2 *Metadata* — None.

5.14 *Element Name — Company*

5.14.1 *Element Description* — The Company element defines a company, as a legal entity, within the context of the SEMI E36 specification. The content of a Company element will be interpreted as a description of the referenced company that manufactures the specific semiconductor manufacturing equipment.

5.14.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Company element supports the following additional metadata, all in the e36base namespace:

5.14.2.1 *CompanyName* — String name of referenced company. (mandatory)

5.14.2.2 *CompanyURL* — URL to homepage of referenced company. (mandatory)

5.15 *Element Name — Publisher*

5.15.1 *Description* — The Publisher element specifies the business unit within a company that represents ownership of the content for specific semiconductor manufacturing equipment set. The content of a Publisher element can be interpreted as a description of the referenced business unit.

5.15.2 *Metadata* — In addition to the metadata available from the E36Item tag, the Publisher element supports the following additional metadata, all in the e36base namespace:

5.15.2.1 *BusinessUnit* — Supplier-defined string name of referenced business unit plus referenced company.

5.15.2.2 *BusinessUnitURL* — URL to homepage of appropriate business unit within referenced company.

6 Requirements

6.1 The standard was segmented into two levels for the following reasons. First, organizing the work into these segments makes each segment more coherent. Next, layering the segments provides a flexible means for adopting the standard incrementally.

6.1.1 The metadata depends on the identity layer. The internal structure layer implies the identity layer.

6.2 *Identifiers* — The use of identifiers is critical in the design. Each of the levels requires them to function. The reason is that being able to unambiguously identify an item independently of the item itself enables elegant solutions to many problems:

- Item Versioning

- Item Security
- Item Lookup
- Distributed Metadata
- Efficient Distribution

6.3 *Externalizing Information* — Many XML initiatives focus on rigorously defining rich structure for Elements. The current approach has been the opposite, to externalize as much structure as possible.

6.3.1 This approach is to promote flexibility and simplicity. The objective was to provide a small set of powerful modeling primitives from which richer models may be built. Providing a mechanism for distributing related information among different locations allows systems to capitalize on this property for economic efficiency. The metadata model allows content to be stored in some sort of native XML format, while associated metadata is maintained in a relational database system. This creates more options for users.

6.4 *Utilizing RDF as the Basis for Metadata* — Level 2 type and metadata is based on the existing XML standard for metadata called the Resource Description Framework (RDF). RDF provides a very simple, but very powerful model of information that is easy to understand and process. RDF allows flexible extension of the metadata standard without having to modify existing schema definitions. This will allow users to incorporate future key learnings into the standard.

6.5 *Tag and Attribute Selection* — A large number of tags and attributes were considered for possible inclusion in the standard. In making decisions about which of the many possibilities to choose as the basis of this work, several factors were considered. First representative users of the information were interviewed to identify concepts and relationships that they considered to be important and valuable. Tags resulting from this work were given high priority relative to tags that derived from other sources. Next, semantic or conceptual tags were favored over tags derived from structural or physical properties of the service documents. Finally, tags determined to have broad utility were chosen in favor of tags considered useful in only a limited context.

6.5.1 Once a reasonable set of tags were selected for inclusion, a set of attributes was produced for those tags. The basic approach was to determine an attribute set that was necessary and sufficient for making the information captured by the tags economical to produce and beneficial to end users.

RELATED INFORMATION 1
IDENTITY AND METADATA EXAMPLE

NOTICE: This related information is not an official part of SEMI E36.1 and was derived from the work of the North American Information & Control committee. This related information was approved for publication by full letter ballot on March 14, 2004.

R1-1 Identity and Metadata Example

R1-1.1 This section illustrates the use of the identity and metadata defined in the standard. The material shown below was produced using a tool called the "RDF Validation Service" available on the web at http://www.w3.org/RDF/Validator/. Given a candidate XML/RDF document, the tool will first validate the document for compliance with the RDF specification. Next, the document is parsed and represented in three alternative forms: the original XML version, a tabular version, and a graphical version.

R1-1.2 The graph shows that the sample RDF document contains five elements, as shown by the five ellipses on the left side of the graphic. These items are the subjects of the RDF model. These are the items we are associating metadata. Each subject has a URI embedded within; this URI points to the content that the subject refers to. This binds metadata to a subject in a completely different context than that in which the subject's content is developed.

R1-1.3 Emanating from the subjects are arrows. Each of these arrows defines a relationship between the subject and another information item. The nature of the relationship is indicated as a label on the arc and corresponds to one of the relationships defined in the E36 RDF model.

R1-1.4 These arrows terminate on graphics representing the object of the relationship. In cases where the terminating graphic is a box, the object of the relationship is a literal value such as a number, date, or phrase. When the terminating graphic is an oval, this is an indication that the object of the relationship is another resource. A URI is embedded within the oval to identify which resource is being referred to.

R1-1.5 *Original RDF/XML document*

```
1: <?xml version='1.0' encoding='UTF-8'?>
2: <rdf:RDF
3: xmlns:rdf = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'
4: xmlns:rdfs = 'http://www.w3.org/2000/01/rdf-schema#'
5: xmlns:dc = 'http://purl.org/dc/elements/1.1/'
6: xmlns:e36base = 'http://www.semi.org/schemas/e36Base#'
7: >
8:
9: <e36base:Component rdf:about='D7C2E128-A1F2-4FAC-962F-00BA4E6150E5'>
10:       <e36base:releaseDate>4/29/2003</e36base:releaseDate>
11:       <dc:title>Cables</dc:title>
12:       <dc:publisher>Some Publisher</dc:publisher>
13:       <e36base:oemPartNumber>01-1010110</e36base:oemPartNumber>
14:       <e36base:partName>Pump control</e36base:partName>
15:       <e36base:oemName>Some OEM</e36base:oemName>
16: </e36base:Component>
17:
18: <e36base:Component rdf:about='C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE'>
19:       <e36base:releaseDate>4/29/2003</e36base:releaseDate>
20:       <dc:title>Cables</dc:title>
21:       <dc:publisher>Some Other Publisher</dc:publisher>
22:       <e36base:oemPartNumber>02-1101001-01</e36base:oemPartNumber>
23:       <e36base:partName>Power Cable</e36base:partName>
24:       <e36base:oemName>Some Other OEM</e36base:oemName>
25: </e36base:Component>
26:
27: <e36base:Procedure rdf:about='CA46B949-B999-40B3-8554-2513DADEEBF7'>
28:       <e36base:releaseDate>4/29/2003</e36base:releaseDate>
29:       <dc:title>How To Use This Manual</dc:title>
30:       <dc:publisher>A Publisher</dc:publisher>
31:       <e36base:procedureType>OperationProcedure</e36base:procedureType>
32: </e36base:Procedure>
33:
34: <e36base:Recipe rdf:about='F9A6EB40-2DE4-4AF5-82C9-6C26E97BD197'>
35:       <e36base:releaseDate>4/29/2003</e36base:releaseDate>
```

```
36:       <dc:title>Some Operation</dc:title>
37:       <e36base:recipeType>ProcessRecipe</e36base:recipeType>
38: </e36base:Recipe>
39:
40: <e36base:SafetyItem rdf:about='0566910F-2C23-443F-B498-03B984735675'>
41:       <e36base:releaseDate>4/29/2003</e36base:releaseDate>
42:       <dc:title>Hazardous Material Detection</dc:title>
43: </e36base:SafetyItem>
44:
45: </rdf:RDF>
46:
```

Table R1-1 Triples of the Data Model

Number	*Subject*	*Predicate*	*Object*
1	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semi.org/schemas/e36Base#Component
2	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://www.semi.org/schemas/e36Base#releaseDate	" 4/29/2003"
3	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://purl.org/dc/elements/1.1/title	" Cables"
4	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://purl.org/dc/elements/1.1/publisher	" Some Publisher"
5	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://www.semi.org/schemas/e36Base#oemPartNumber	" 01-1010110"
6	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://www.semi.org/schemas/e36Base#partName	" Pump control"
7	http://unknown.org/D7C2E128-A1F2-4FAC-962F-00BA4E6150E5	http://www.semi.org/schemas/e36Base#oemName	" Some OEM"
8	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semi.org/schemas/e36Base#Component
9	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://www.semi.org/schemas/e36Base#releaseDate	" 4/29/2003"
10	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://purl.org/dc/elements/1.1/title	" Cables"
11	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://purl.org/dc/elements/1.1/publisher	" Some Other Publisher"
12	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://www.semi.org/schemas/e36Base#oemPartNumber	" 02-1101001-01"
13	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://www.semi.org/schemas/e36Base#partName	" Power Cable"
14	http://unknown.org/C89722A8-2B03-4DDF-8ABD-68F01FCD2ADE	http://www.semi.org/schemas/e36Base#oemName	" Some Other OEM"
15	http://unknown.org/CA46B949-B999-40B3-8554-	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semi.org/schemas/e36Base#Procedure

Number	*Subject*	*Predicate*	*Object*
	2513DADEEBF7		
16	http://unknown.org/CA46B949-B999-40B3-8554-2513DADEEBF7	http://www.semi.org/schemas/e36Base#releaseDate	" 4/29/2003"
17	http://unknown.org/CA46B949-B999-40B3-8554-2513DADEEBF7	http://purl.org/dc/elements/1.1/title	" How To Use This Manual"
18	http://unknown.org/CA46B949-B999-40B3-8554-2513DADEEBF7	http://purl.org/dc/elements/1.1/publisher	" A Publisher"
19	http://unknown.org/CA46B949-B999-40B3-8554-2513DADEEBF7	http://www.semi.org/schemas/e36Base#procedureType	" OperationProcedure"
20	http://unknown.org/F9A6EB40-2DE4-4AF5-82C9-6C26E97BD197	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semi.org/schemas/e36Base#Recipe
21	http://unknown.org/F9A6EB40-2DE4-4AF5-82C9-6C26E97BD197	http://www.semi.org/schemas/e36Base#releaseDate	" 4/29/2003"
22	http://unknown.org/F9A6EB40-2DE4-4AF5-82C9-6C26E97BD197	http://purl.org/dc/elements/1.1/title	" Some Operation"
23	http://unknown.org/F9A6EB40-2DE4-4AF5-82C9-6C26E97BD197	http://www.semi.org/schemas/e36Base#recipeType	" ProcessRecipe"
24	http://unknown.org/0566910F-2C23-443F-B498-03B984735675	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.semi.org/schemas/e36Base#SafetyItem
25	http://unknown.org/0566910F-2C23-443F-B498-03B984735675	http://www.semi.org/schemas/e36Base#releaseDate	" 4/29/2003"
26	http://unknown.org/0566910F-2C23-443F-B498-03B984735675	http://purl.org/dc/elements/1.1/title	" Hazardous Material Detection"

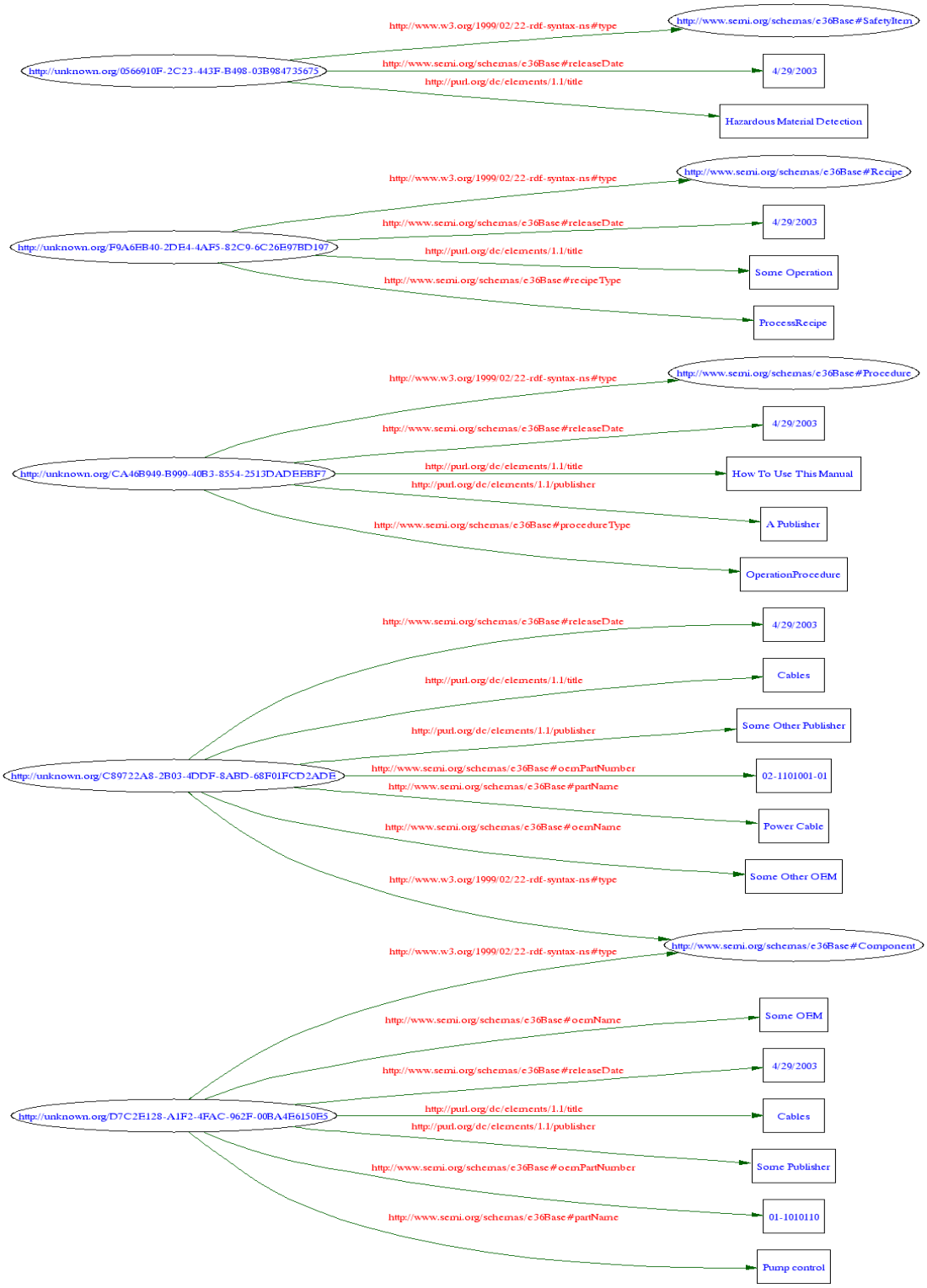


Figure R1-1
Validation Results

RELATED INFORMATION 2
E36 METADATA SCHEMA

NOTICE: This related information is not an official part of SEMI E36.1 and was derived from the work of the North American Information & Control Committee. This related information was approved for publication by full letter ballot on March 14, 2004.

R2-1 E36 Metadata Schema

R2-1.1 The following is the schematic description of the Level 1 and Level 2 requirements. The description is presented in the form of an RDF schema, compatible with the schema definitions available at the website listed in section 3.2b

R2-1.2 E36 Metadata Schema XML Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#E36Item"/>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Configuration">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Manual">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Recipe">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Procedure">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Component">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Narrative">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#ErrorCode">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Media">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#SafetyItem">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Company">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Publisher">
       <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.semi.org/e36/schema#Configuration">
```

```
        <rdfs:subClassOf rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
</rdfs:Class>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#releaseDate">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#expirationDate">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#versionString">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#E36Item"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#manualType">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Manual"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#waferSize">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Configuration"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#mainframe">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Configuration"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#technologyArea">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Configuration"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#process">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Configuration"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#equipment">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Configuration"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#recipeType">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Recipe"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#procedureType">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Procedure"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#scheduleType">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Procedure"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#scheduleUnits">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Procedure"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#supplierPartNumber">
        <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Component"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
```

```
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#oemPartNumber">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Component"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#partName">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Component"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#oemName">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Component"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#errorNumber">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#ErrorCode"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#mediaUrl">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Media"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#caption">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Media"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#mediaCategory">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Media"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#companyURL">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Company"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#companyName">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Company"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#businessUnit">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Publisher"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.semi.org/e36/schema#businessUnitURL">
       <rdfs:domain rdf:resource="http://www.semi.org/e36/schema#Publisher"/>
       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>


</rdf:RDF>
```

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 or equipment mentioned herein. These standards are subject to change without notice.

By publication of this standard, Semiconductor Equipment and Materials International (SEMI) takes no position respecting the validity of any patent rights or copyrights asserted in connection with any items 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 E37-0303
HIGH-SPEED SECS MESSAGE SERVICES (HSMS) GENERIC SERVICES

NOTICE: The designation of SEMI E37 was updated during the 0303 publishing cycle to reflect the reapproval of SEMI E37.2.

1 Purpose

HSMS provides a means for independent manufacturers to produce implementations which can be connected and interoperate without requiring specific knowledge of one another.

HSMS is intended as an alternative to SEMI E4 (SECS-I) for applications where higher speed communication is needed or when a simple point-to-point topology is insufficient. SEMI E4 (SECS-I) can still be used in applications where these and other attributes of HSMS are not required.

HSMS is also intended as an alternative to SEMI E13 (SECS Message Services) for applications where TCP/IP is preferred over OSI.

It is intended that HSMS be supplemented by subsidiary standards which further specify details of its use or impose restrictions on its use in particular application domains.

2 Scope

High-Speed SECS Message Services (HSMS) defines a communication interface suitable for the exchange of messages between computers in a semiconductor factory.

3 Referenced Documents

3.1 *SEMI Standards*

SEMI E4 — SEMI Equipment Communication Standard 1 — Message Transport (SECS-I)

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

3.2 *IETF Documents*[1]

IETF RFC 791 — Internet Protocol

IETF RFC 792 — Internet Control Message Protocol

IETF RFC 793 — Transmission Control Protocol

IETF RFC 1120 — Requirements for Internet Hosts - Communication Layers

IETF RFC 1340 — Assigned Numbers. Note: This RFC supersedes RFC 820.

3.3 *POSIX Document*[2]

IEEE POSIX P1003.12 — Protocol Independent Interfaces (PII)

4 Terminology

API — Application Program Interface. In the case of TCP/IP, a set of programming conventions used by an application program to prepare for or invoke TCP/IP capabilities.

communication failure — a failure in the communication link resulting from a transition to the NOT CONNECTED state from the SELECTED state. (See Section 9.)

confirmed service (HSMS) — an HSMS service requested by sending a message from the initiator to the responding entity which requires that completion of the service be indicated by a response message from the responding entity to the initiator.

connection — a logical linkage established on a TCP/IP LAN between two entities for the purposes of exchanging messages.

control message — an HSMS message used for the management of HSMS sessions between two entities.

data message — an HSMS message used for communication of application-specific data within an HSMS session. A Data Message can be a Primary Message or a Reply Message.

entity — an application program associated with an endpoint of a TCP/IP connection.

header — a 10-byte data element preceding every HSMS message.

initiator (HSMS) — the entity requesting an HSMS service. The initiator requests the service by sending an appropriate HSMS message.

1 The IETF documents can be obtained from The Network Information Center, Network Solutions, 14700 Park Meadow Drive, Suite 200, Chantilly, VA 22021 USA

2 POSIX documents can be obtained from Institute of Electrical and Electronic Engineers (IEEE), 345 East 47th Street, New York, NY 10017 USA

IP Address — Internet Protocol Address. A logical address which uniquely identifies a particular attachment to a TCP/IP network.

local entity — relative to a particular end point of a connection, the local entity is that entity associated with that endpoint.

local entity-specific — general qualifier to any procedure, option, issue, or other implementation matter which is not a subject of this standard and left to the discretion of the individual supplier.

message — a complete unit of communication in one direction. An HSMS Message consists of the Message Length, Message Header, and the Message Text. An HSMS Message can be a Data Message or a Control Message.

message length — a 4-byte unsigned integer field specifying the length of a message in bytes.

open transaction — a transaction in progress.

port — an endpoint of a TCP/IP connection whose complete network address is specified by an IP Address and TCP/IP Port number.

port number — (or TCP port number). The address of a port within an attachment to a TCP/IP network which can serve as an endpoint of a TCP/IP connection.

primary message — an HSMS Data Message with an odd numbered Function. Also, the first message of a data transaction.

published port — a TCP/IP IP Address and Port number associated with a particular entity (server) which that entity intends to use for receiving TCP/IP connection requests. An entity' s published port must be known by remote entities intending to initiate connections.

receiver — the HSMS Entity receiving a message.

remote entity — relative to a particular endpoint of a connection, the remote entity is the entity associated with the opposite endpoint of the connection.

reply — an HSMS Data Message with an even-numbered function. Also, the appropriate response to a Primary HSMS Data Message.

responding entity (HSMS) — the provider of an HSMS service. The responding entity receives a message from an initiator requesting the service. In the event of a confirmed service, the responding entity indicates completion of the requested service by sending an appropriate HSMS response message to the initiator of the request. In an unconfirmed service, the responding entity does not send a response message.

session — a relationship established between two entities for the purpose of exchanging HSMS messages.

session entity — an entity participating in an HSMS session.

session ID — a 16-bit unsigned integer which identifies a particular session between particular session entities.

stream (TCP/IP) — a sequence of bytes presented at one end of a TCP/IP connection for delivery to the other end. TCP/IP guarantees that the delivered sequence of bytes matches the presented stream. HSMS subdivides a stream into blocks of contiguous bytes - messages.

T3 — reply timeout in the HSMS protocol.

T5 — connect Separation Timeout in the HSMS protocol used to prevent excessive TCP/IP connect activity by providing a minimum time between the breaking, by an entity, of a TCP/IP connection or a failed attempt to establish one, and the attempt, by that same entity, to initiate a new TCP/IP connection.

T6 — control Timeout in the HSMS protocol which defines the maximum time an HSMS control transaction can remain open before a communications failure is considered to have occurred. A transaction is considered open from the time the initiator sends the required request message until the response message is received.

T7 — connection Idle Timeout in the HSMS protocol which defines the maximum amount of time which may transpire between the formation of a TCP/IP connection and the use of that connection for HSMS communications before a communications failure is considered to have occurred.

T8 — network Intercharacter Timeout in the HSMS protocol which defines the maximum amount of time which may transpire between the receipt of any two successive bytes of a complete HSMS message before a communications failure is considered to have occurred.

TCP/IP — Transmission Control Protocol/Internet Protocol. A method of communications which provides reliable, connection-oriented message exchange between computers within a network.

TLI — Transport Level Interface. One particular API provided by certain implementations of TCP/IP which provides a transport protocol and operating system independent definition of the use of any Transport Level protocol.

transaction — a Primary Message and its associated Reply message, if required. Also, an HSMS Control Message of the request (.req) type, and its response Control Message (.rsp), if required.

unconfirmed service (HSMS) — an HSMS service requested by sending a message from the initiator to the responding entity which requires no indication of completion from the responding entity.

5 HSMS Overview and State Diagram

High-Speed SECS Message Services (HSMS) defines a communication interface suitable for the exchange of messages between computers in a semiconductor factory using a TCP/IP environment. HSMS uses TCP/IP stream support, which provides reliable two way simultaneous transmision of streams of contiguous bytes. It can be used as a replacement for SECS-I communication as well as other more advanced communications environments.

The procedure for HSMS communications parallels the more familiar SECS-I communications it replaces. The following steps are followed for any communications (HSMS or otherwise):

1. Obtain a communications link between two entities. In SECS-I, this is the RS232 wire physically connecting host and equipment. In HSMS, the link is a TCP/IP connection obtained by the standard TCP/IP connect procedure. Note that the abstract term "entity" is used instead of "host" or "equipment." This is because, while HSMS is used for SECS-I replacement, it has more general applications as well. In a SECS-I replacement application, the "host" is an "entity" and the "equipment" is an "entity."

2. Establish the application protocol conventions to be used for exchanging data messages between two entities. For SECS-I, this step is implicit in the fact that semiconductor equipment is physically connected on the two ends of the wire: the protocol is SECS-II.

   In the case of HSMS, the communications link is a dynamically established TCP/IP connection on a physical link which may be shared with many other TCP/IP connections using protocols other than HSMS or connections using non TCP/IP protocols. HSMS adds a message exchange (called the Select procedure) which is used to confirm to both entities that the particular TCP/IP connection is to be used exlusively for HSMS communications.

3. Exchange Data. This is the normal intended purpose of the communications link. In both SECS-I and HSMS, the procedure is to exchange SECS-II encoded messages for the control of semiconductor equipment and/or processes. Data exchange normally continues until one or both of the entities are taken off-line for equipment-specific purposes, such as maintenance.

4. Formally end communications. In SECS-I, there is no formal requirement here; the equipment to be taken off-line stops communicating.

   In HSMS, a message exchange (either the "bilateral" Deselect procedure or the "unilateral" separate procedure) is used for both parties to confirm that the TCP/IP connection is no longer needed for HSMS communications.

5. Break the communications link. In SECS-I, this is done by physically unplugging the host or equipment from the communications cable, which only occurs during repair or physical reconfiguration of the factory network environment.

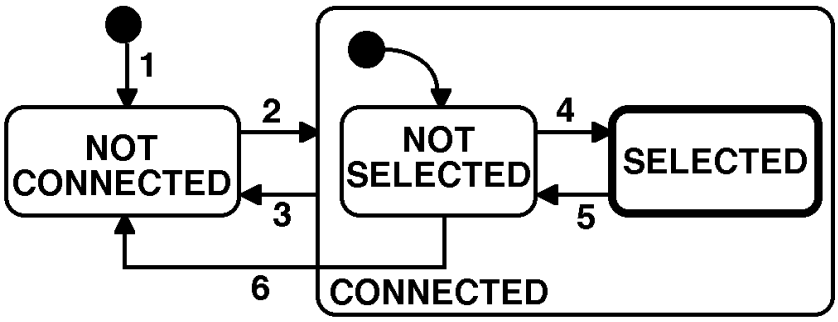
   In HSMS, since it uses the dynamic connection environment of TCP/IP, the TCP/IP connection is logically broken via a release or a disconnect procedure without any physical disconnect from the network medium.

Two additional procedures, of a diagnostic nature, are supported in HSMS, which are generally not required by a simple SECS-I link or a SECS-I direct replacement. These follow:

1. Linktest. This procedure provides a simple confirmation of connection integrity.

2. Reject. Because HSMS is intended to be extended to protocols other than just SECS-II (by means of subsidiary standards), it is possible that two entities can be connected (due to a configuration error) which use incompatible subsidiary standards. Also, during initial implementation, incorrect message types may be sent, or they may be sent out of order due to software bugs. The reject procedure is used to indicate such an occurrence.

5.1 *HSMS Connection State Diagram* — The HSMS state machine is illustrated in the diagram below. The behavior described in this diagram defines the basic requirements of HSMS: subsidiary standards may further extend these or other states.



5.2 *State Descriptions*

5.2.1 *NOT CONNECTED* — The entity is ready to listen for or initiate TCP/IP connections but either has not yet established any connections or all previously established TCP/IP connections have been terminated.

5.2.2 *CONNECTED* — A TCP/IP connection has been established. This state has two substates, NOT SELECTED and SELECTED.

5.2.2.1 *NOT SELECTED* — A substate of CONNECTED in which no HSMS session has been established or any previously established HSMS session has ended.

5.2.2.2 *SELECTED* — A substate of CONNECTED in which at least one HSMS session has been established. This is the normal "operating" state of HSMS: data messages may be exchanged in this state. It is highlighted by a heavy outline in the state diagram.

5.3 *State Transition Table*

Table 1

#	*Current State*	*Trigger*	*New State*	*Actions*	*Comment*
1	...	Local entity-specific preparation for TCP/IP communication.	NOT CONNECTED	Local entity-specific	Action depends on connection procedure to be used: active or passive.
2	NOTCONNECTED	A TCP/IP connection is established for HSMS communication.	CONNECTED - NOT SELECTED	Local entity-specific	none
3	CONNECTED	Breaking of TCP connection.	NOT CONNECTED	Local entity-specific	See Section 6.4.
4	NOT SELECTED	Successful completion of HSMS Select Procedure.	SELECTED	Local entity-specific	HSMS communication is now fully established: data message exchange is permitted.
5	SELECTED	Successful completion of HSMS Deselect or Separate.	NOT SELECTED	Local entity-specific	This transition normally indicates the end of HSMS communication and so an entity would immediately proceed to break the TCP/IP connection (transition 3 above).
6	NOT SELECTED	T7 Connection Timeout.	NOT CORRECTED	Local entity-specific	per Section 9.2.2

6 Use of TCP/IP

6.1 *TCP/IP API* — The specification of a required TCP Application Program Interface (API) for use in implementations is outside the scope of HSMS. A given HSMS implementation may use any TCP/IP API — sockets, TLI (Transport Layer Interface), etc. — appropriate to the intended hardware and software platform, as long as it provides interoperable TCP/IP streams protocol on the network.

The appendix contains examples of the TCP/IP procedures referenced in this standard and sample scenarios using both the TLI (POSIX standard 1003.12) and the popular BSD socket model for TCP/IP communication.

6.2 *TCP/IP Network Addressing Conventions*

6.2.1 *IP Addresses* — Each physical TCP/IP connection to a given Local Area Network (LAN) must have a unique IP Address. IP Addresses must be assignable at installation time, and an HSMS implementation cannot select a fixed IP Address. A typical IP Address is 192.9.200.1.

IP imposes restrictions on these numbers which are outside the scope of the HSMS protocol. Consult Section 2.3 of RFC 791, Internet Protocol (IP) in Section 3.

6.2.2 *TCP Port Numbers* — A TCP Port Number can be considered as an extension of the IP Address.

HSMS implementations should allow configuring TCP Port to the full range of the TCP/IP implementation used. A typical TCP Port Number is 5000.

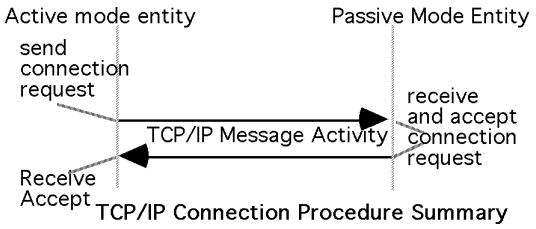
Conventions have been established for selecting TCP Port Numbers which are outside the scope of the HSMS protocol. Consult RFC 793, Transmission Control Protocol (TCP) in Section 3.

6.3 *Establishing a TCP/IP Connection*

6.3.1 *Connect Modes* — The procedures for establishing a TCP/IP connection are defined in RFC 793. However, not all the procedures defined by RFC 793 are supported by commonly available APIs. In particular, while RFC 793 permits both entities to initiate the connection simultaneously, this feature is rarely supported in available

APIs. Therefore, HSMS restricts an entity to one of the following modes:

- Passive Mode. The Passive mode is used when the local entity listens for and accepts a connect procedure initiated by the Remote Entity.
- Active Mode. The Active mode is used when the connect procedure is initiated by the Local Entity.



TCP/IP Connection Procedure Summary

The appendix provides an example of how an entity may operate alternately in the active and passive modes to achieve greater flexibility in establishing communications.

6.3.2 *Passive Mode Connect Procedure* — The procedure followed by the Passive Local Entity is defined in RFC 793. It is summarized as follows:

1. Obtain a connection endpoint and bind it to a published port.
2. Listen for an incoming connect request to the published port from a remote entity.
3. Upon receipt of a connect request, acknowledge it and indicate acceptance of the connection. At this point, the connect procedure has completed successfully, and the CONNECTED state is entered (Section 5).

These procedures are carried out through the API of the local entity' s implementation of TCP/IP. The appendices provide the API-specific procedures for the above steps using both TLI and BSD.

NOTE: A failure may occur during the above steps. The reason for failure may be local entity-specific or may be due to a lack of any connect request after a local entity-specific timeout. The action to be taken (for example: return to step 1 to retry) is a local entity-specific issue.

NOTE: See Section 9, Special Considerations, for issues relating to multiple connection requests to the same passive mode entity.

6.3.3 *Active Mode Connect Procedure* — The procedure followed by the Active Local Entity is defined in RFC 793. It is summarized as follows:

1. Obtain a connection endpoint.
2. Initiate a connection to the published port of a passive mode remote entity.
3. Wait for the receipt of the acknowledge and the acceptance of the connect request from the remote entity. Receipt of the acceptance from the remote

entity indicates successful completion of the connect procedure, and the CONNECTED state is entered (Section 5).

These procedures are carried out through the API of the local entity' s implementation of TCP/IP. The appendix provides the API-specific procedures for the above steps using both TLI and BSD.

NOTE: A failure may occur during the above steps. The reason for failure may be local entity-specific or may be due to a lack of any accept message after a local entity-specific timeout. The action to be taken is a local entity-specific issue. If, however, the local entity intends to retry the connection, it should do so subject to the T5 connect separation timeout (see "Special Considerations").

6.4 *Terminating a TCP/IP Connection* — Connection termination is the logical inverse of Connection establishment. From the Local Entity' s perspective, a TCP/IP connection may be broken at any time. However, HSMS only permits termination of the connection when the connection is in the NOT SELECTED substate of the CONNECTED state.

The procedures for termination of a connection are defined in RFC 793. Either entity may initiate termination of the connection. The NOT CONNECTED state is entered, indicating the end of HSMS communications. The appendix illustrates the procedures for both release and disconnect using the TLI and BSD APIs.

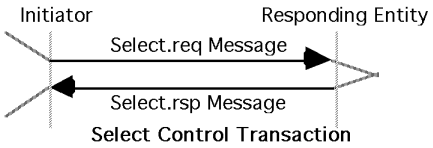
7 HSMS Message Exchange Procedures

HSMS defines the procedures for all message exchange between entities across the TCP/IP connection established according to the procedures in the previous section. As explained in the overview, once the connection is established, the two entities establish HSMS communications with the Select procedure. Then data messages may be exchanged in either direction at any time. When the entities wish to end HSMS communications, the Deselect or Separate procedure is used to end HSMS communications.

7.1 *Sending and Receiving HSMS Messages* — All HSMS procedures involve the exchange of HSMS messages. These messages are sent and received as TCP/IP streams using the previously established TCP/IP connection at standard priority. In particular, the use of "Urgent" data is not supported under HSMS (see RFC 793 for more information on send and receive procedures).

The appendix gives examples of sending and receiving HSMS messages using both TLI and BSD socket APIs.

7.2 *Select Procedure* — The Select procedure is used to establish HSMS communications on a TCP/IP connection using the Select.req and Select.rsp messages in a control transaction.



Although HSMS permits Select at any time in the CONNECTED state, subsidiary standards may further require the connection to be in the NOT SELECTED substate (see "Special Considerations").

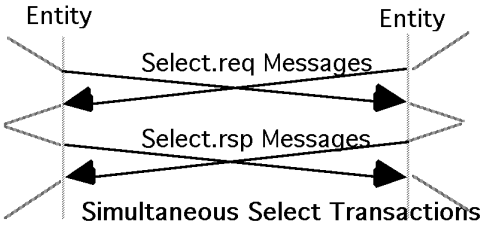
7.2.1 *Initiator Procedure* — The procedure followed by the initiator is as follows.

1. The initiator of the select procedure sends the Select.req message to the responding entity.
2. If the initiator receives a Select.rsp with a Select Status of 0, The HSMS Select procedure completes successfully and the SELECTED state is entered (see Section 5).
3. If the initiator receives a Select.rsp with a non-zero Select Status, the Select completes unsuccessfully (no state transitions).
4. If the T6 timeout expires in the initiator before receipt of a Select.rsp, it is considered a communications failure (see "Special Considerations").

7.2.2 *Responding Entity Procedure* — The procedure followed by the responding entity is as follows.

1. The responding entity receives the Select.req.
2. If the responding entity is able to accept the select, it transmits the Select.rsp with a Select Status of 0. The HSMS Select Procedure for the responding entity is successfully completed, and the SELECTED state is entered (see Section 5).
3. If the responding entity is unable to permit the select, it transmits the Select.rsp with a non-zero Select Status. The HSMS Select Procedure for the responding entity completes unsuccessfully (no state transitions).

7.2.3 *Simultaneous Select Procedures* — If the subsidiary standards do not restrict the use of the Select, it is possible that both entities simultaneously initiate Select Procedures with identical SessionID's. In such a case, each entity will accept the other entity' s select request by responding with a Select.rsp.

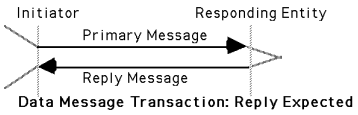


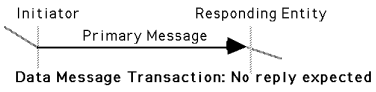
7.3 *Data Procedure* — HSMS data messages may be initiated by either entity as long as the connection is in the SELECTED state. Receipt of a data message when not in the SELECTED state will result in a reject procedure (see Section 7.7).

Data messages may be further defined as part of a data transaction as either a "Primary" or "Reply" data message. In a data transaction, the initiator of the transaction sends a primary message to the responding entity. If the Primary message indicates that a reply is expected, a Reply message is sent by the responding entity in response to the Primary.

The following types of Data Transactions are supported:

1. Primary Message with reply expected and the associated Reply Message.
2. Primary Message with no reply expected.

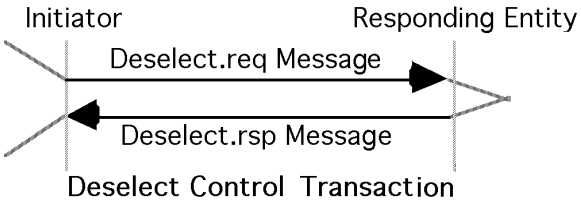




The specific procedures for these transactions are determined by the application layer and are subject to other standards (for example, E5 and E30 for GEM equipment using SECS-II encoded messages).

The applicable upper layer standard is identified by the message type. The type is determined from the specific format defined in Section 8. The normal type for HSMS messages is SECS-II text. Also refer to "Special Considerations" concerning the T3 Reply Timeout.

7.4 *Deselect Procedure* — The Deselect procedure is used to provide a graceful end to HSMS communication for an entity prior to breaking the TCP/IP connection. HSMS requires that the connection be in the SELECTED state. The procedure is as follows.



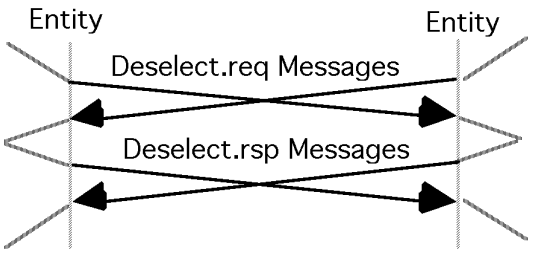
7.4.1 *Initiator Procedure*

1. The initiator of the Deselect procedure sends the Deselect.req message to the responding entity.
2. If the initiator receives a Deselect.rsp with a Deselect Status of 0, its Deselect procedure terminates successfully. The NOT SELECTED state is entered (see Section 5).
3. If the initiator receives a Deselect.rsp with a non-zero Deselect Status, its Deselect procedure terminates unsuccessfully. No state change occurs.
4. If the T6 timeout expires in the initiator before receipt of a Deselect.rsp, it is considered a communications failure (see "Special Considerations").

7.4.2 *Responding Entity Procedure*

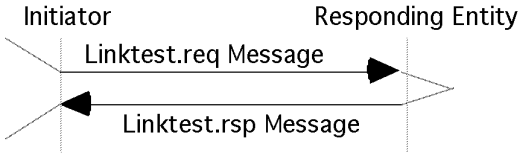
1. The responding entity receives the Deselect.req message.
2. If the responding entity is in the SELECTED state, and if it is able to permit the Deselect, it responds using the Deselect.rsp with a zero response code. The responding entity' s Deselect procedure completes successfully. The NOT SELECTED state is entered (see Section 5).
3. If the responding entity is unable to permit the Deselect, either because it is not in the SELECTED state or because local conditions do not permit the Deselect, it responds using the Deselect.rsp with a non-zero response code. The responding entity' s Deselect procedure terminates unsuccessfully. No state change occurs.

7.4.3 *Simultaneous Deselect Procedures* — If the subsidiary standards do not restrict the use of the Deselect, it is possible that both entities simultaneously initiate Deselect Procedures with identical SessionID's. In such a case, each entity will accept the other entity' s Deselect request by responding with the deselect.rsp.



Simultaneous Deselect Transactions

7.5 *Linktest Procedure* — The Linktest is used to determine the operational integrity of TCP/IP and HSMS communications. Its use is valid anytime in the CONNECTED state.



Linktest Control Transaction

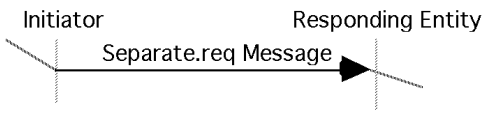
7.5.1 *Initiator Procedure*

1. The initiator of the Linktest procedure sends the Linktest.req message to the responding entity.
2. If the initiator receives a Linktest.rsp within the T6 timeout, the Linktest is successfully completed.
3. If the T6 timeout expires in the initiator before receipt of a Linktest.rsp, it is considered a communications failure (see "Special Considerations").

7.5.2 *Responding Entity Procedure*

1. The responding entity receives the Linktest.req from the initiator.
2. The responding entity sends a Linktest.rsp.

7.6 *Separate Procedure* — The Separate procedure is used to abruptly terminate HSMS communication for an entity prior to breaking the TCP/IP Connection. HSMS requires that the connection be in the SELECTED state when using Separate. The responding entity does not send a response and is required to terminate communications regardless of its local state. The procedure is as follows.



Separate Control Transaction

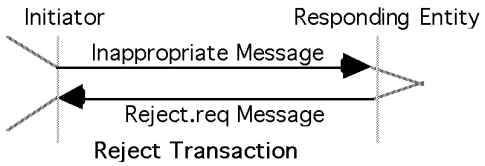
7.6.1 *Initiator Procedure*

1. The initiator of the select procedure sends the Separate.req message to the responding entity. The initiator' s Separate procedure completes successfully. The NOT SELECTED state is entered (see Section 5).

7.6.2 *Responding Entity Procedure*

1. The responding entity receives the Separate.req from the initiator.
2. If the responding entity is in the SELECTED state, its Separate procedure completes successfully.
3. If the responding entity is not in the SELECTED state, the Separate.req is ignored.

7.7 *Reject Procedure* — The Reject procedure is used in response to an otherwise valid HSMS message received in an inappropriate context. Supporting the reject procedure can provide useful diagnostic information during the development of a distributed application using HSMS. The procedure is as follows:



Reject Transaction

7.7.1 *Initiator (Sender of Inappropriate Message) Procedure*

1. The initiator of the inappropriate message, upon receiving the Reject.req, takes appropriate action (local entity-specific).

7.7.2 *Responding Entity Procedure*

1. The entity receiving the inappropriate message responds with a Reject.req message.

HSMS requires the reject procedure for the receipt of a data message in the NOT SELECTED state, or the receipt of a message whose SType or PType (see next section: Message Format) is not defined for the entity receiving the message. Subsidiary standards may define other conditions which require the Reject Procedure. In general, receipt of a reject message is an indication of an improperly configured system or a software programming error.

8 HSMS Message Format

This section defines the detailed format of the messages used by the procedures in the previous section.

8.1 *General Message Format*

8.1.1 *Byte Structure* — Within HSMS, a byte contains eight (8) bits. The bits in a byte are numbered from Bit 7 (most significant) to Bit 0 (least significant).

8.1.2 *Message Format* — An HSMS Message is transmitted as a single contiguous stream of bytes in the following order:

Table 2 HSMS Message Format

Number of Bytes	*Description*
4 Bytes	Message Length. MSB First. Specifies the number of bytes in the Message Header plus the Message Text.
10 Bytes	Message Header.
0–n Bytes	Message Text. Format is further specified by PType field of message header.

8.1.3 *Message Length* — Message Length is a four-byte unsigned integer value which specifies the length in bytes of the Message Header plus the Message Text. Message Length is transmitted most significant byte (MSB) first and least significant byte (LSB) last.

The minimum possible Message Length is 10 (Header only). The maximum possible Message Length is implementation-specific.

8.1.4 *Message Header* — The Message Header is a ten-byte field. The bytes in the header are numbered from byte 0 (first byte transmitted) to byte 9 (last byte transmitted). The format of the Message Header is as follows:

Table 3 HSMS Message Header

Bytes	*Description*
0–1	Session ID (Device ID)
2	Header Byte 2
3	Header Byte 3
4	PType
5	SType
6–9	System Bytes

The physical byte order is designed to correspond as closely as possible to the SECS-I header.

8.1.4.1 *Session ID* — Session ID is a 16-bit unsigned integer value, which occupies bytes 0 and 1 of the header (byte 0 is MSB, 1 is LSB). Its purpose is to provide an association by reference between control messages (particularly Select and Deselect) and subsequent data messages. It is the role of HSMS subsidiary standards to specify this association further.

8.1.4.2 *Header Byte 2* — This header byte is used in different ways for different HSMS messages. For Control Messages (see SType, below) it contains zero or a status code. For a Data Message whose PType (see below) = 0, it contains the W-Bit and SECS Stream. For a Data Message with PType not equal to 0, see "Special Considerations."

8.1.4.3 *Header Byte 3* — This header byte is used in different ways for different HSMS messages. For Control Messages, it contains zero or a status code. For a Data Message whose PType (see below) = 0, it contains the SECS Function. For a Data Message with PType not equal to 0, see "Special Considerations."

8.1.4.4 *PType* — PType (Presentation Type) is an 8-bit unsigned integer value which occupies byte 4 of the header. PType is intended as an enumerated type defining the presentation layer message type: how the Message Header and Message Text are encoded. Only PType = 0 is defined by HSMS to mean SECS-II message encoding. For non-zero PType values, see "Special Considerations."

Table 4 PType

Value	*Description*
0	SECS-II Encoding
1–127	Reserved for subsidiary standards
128–255	Reserved, not used

8.1.4.5 *SType* — SType (Session Type) is a one-byte unsigned integer value which occupies header byte 5.

SType is an enumerated type identifying whether this message is an HSMS Data Message (value = 0) or one of the HSMS Control Messages (other). Those values not explicitly defined in the table are addressed in "Special Considerations."

Table 5 SType

Value	*Description*
0	Data Message
1	Select.req
2	Select.rsp
3	Deselect.req
4	Deselect.rsp
5	Linktest.req
6	Linktest.rsp
7	Reject.req
8	(not used)
9	Separate.req
10	(not used)
11–127	Reserved for subsidiary standards
128–255	Reserved, not used

8.1.4.6 *System Bytes* — System Bytes is a four-byte field occupying header bytes 6-9. System Bytes is used to identify a transaction uniquely among the set of open transactions.

Uniqueness — The System Bytes of a Primary Data Message, Select.req, Deselect.req, or Linktest.req message must be unique from those of all other currently open transactions initiated from the same end of the connection. They must also be unique from those of the most recently completed transaction.

Reply Message — The System Bytes of a Reply Data Message must be the same as those of the corresponding Primary Message. The System Bytes of a Select.rsp, Deselect.rsp, or Linktest.rsp must be the same as those of the respective ".req" message.

8.2 *HSMS Message Formats by Type* — The specific interpretation of the header bytes in an HSMS message is dependent on the specific HSMS message type as defined by the value of the SType field. The complete set of messages defined is summarized in the table below, shown for PType = 0 (SECS-II message format).

Table 6 HSMS Message Format Summary

	Message Header						
Message Type	*Bytes 0-1 SessionID*	*Byte 2*	*Byte 3*	*Byte 4 PType*	*Byte 5 SType*	*Bytes 6–9 System Bytes*	*Message Text*
Data Message	*	W-bit and SECS Stream	SECS Function	0	0	Primary: UniqueReply: Same as primary	Text
Select.req	*	0	0	0	1	Unique	none
Select.rsp	Same as .req	0	Select Status	0	2	Same as .req	none
Deselect.req	*	0	0	0	3	Unique	none
Deselect.rsp	Same as .req	0	Deselect Status	0	4	Same as .req	none
Linktest.req	0xFFFF	0	0	0	5	Unique	none
Linktest.rsp	0xFFFF	0	0	0	6	Same as .req	none
Reject.req	same as message being rejected	PType or SType of message being rejected	Reason Code	0	7	Same as message being rejected	none
Separate.req	*	0	0	0	9	Unique	none

* Indicates further specification by subsidiary standards.

8.2.1 *SType=0: Data Message* — An HSMS message with SType = 0 is used by the HSMS Data procedure to send a Data message, either Primary or Reply. The message format is as follows:

HSMS Message Length is always 10 (the length of the header alone) or greater.

The HSMS Message Header is as follows:

Session ID — As described above. Specific value subject to subsidiary standards.

Header Byte 2 — For messages with PType value = 0 (SECS-II), header byte 2 is formatted as shown below.

7	6	0
W-Bit	Stream	

The most significant bit (bit 7) of Header Byte 2 is the W-Bit. In a Primary Message, the W-Bit indicates whether the Primary Message expects a Reply message. A Primary Message which expects a Reply should set the W-Bit to 1. A Primary Message which does not expect a Reply should set the W-Bit to 0. A Reply Message should always set the W-Bit to 0. The low-order 7 bits (bits 6-0) of Header Byte 2 contain the SECS Stream for the message. The Stream is a 7-bit unsigned integer value, which identifies a major topic of the message, and its use is defined within SEMI E5 (SECS-II).

Header Byte 3 — For messages whose PType value=0, header Byte 3 contains the SECS Function for the message. The Function is an 8-bit unsigned integer value which identifies a minor topic of the message (within the Stream), and its use is defined within SEMI E5 (SECS-II). The least significant bit (bit 0) of the Function defines whether the Data Message is Primary or Reply; the value 1 indicates Primary and the value 0 indicates Reply.

PType — Set PType = 0 for SECS-II messages.

SType = 0

System Bytes — For PType=0 (SECS-II), the following definition applies. For a Primary Message, System Bytes contain a value uniquely identifying this transaction from all other open transactions initiated from the same end of the Connection. For a Reply Message, System Bytes contain the same value as the corresponding Primary Message.

The HSMS Message Text contains the text of the Data Message (if any), formatted as specified by the PType field. For PType = 0, the text will be formatted as SECS-II messages.

NOTE 4: Some Data Messages consist of header only, with no text.

8.2.2 *SType=1: Select.req* — An HSMS message with SType 1 is a "Select Request" Control Message, which is used by the initiator of the procedure for establishing HSMS communications. The message format is as follows:

Message Length is always 10 (Header only).

The HSMS Message Header is as follows:

SessionID — As described above. Specific value subject to subsidiary standards.

Header Byte 2 = 0

Header Byte 3 = 0

PType = 0.

SType = 1

System Bytes — A unique value among open transactions.

8.2.3 *SType=2: Select.rsp* — An HSMS message with SType 2 is a "Select Response" Control Message, used as the response to a Select.req Control message in the procedure for establishing HSMS communications. The message format is as follows:

Message Length is always 10 (Header only).

The HSMS Message Header is as follows:

SessionID — must be equal to the value of the session ID in the corresponding Select.req.

Header Byte 2 =0

Header Byte 3 — SelectStatus. A code of zero indicates success of the Select operation. A non-zero code indicates failure.

Table 7 SelectStatus

Value	*Description*
0	Communication Established. Select was successfully completed.
1	Communication Already Active. A previous select has already established communications to the entity being selected in this select.
2	Connection Not Ready. The Connection is not yet ready to accept select requests.
3	Connect Exhaust. The connection was accepted, but the entity is already servicing a separate TCP/IP connection and is unable to service more than one at any given time.

Value	*Description*
4–127	Reserved for subsidiary standard-specific reasons for select failure.
128–255	Reserved for local entity-specific reasons for select failure.

PType = 0

SType = 2

System Bytes — Equal to value of System Bytes in the corresponding Select.req.

8.2.4 *SType=3: Deselect.req* — An HSMS message with SType 3 is a "Deselect Request" Control Message, used by the initiator of the Select procedure for ending HSMS communication. The message format is as follows:

Message Length is always 10 (Header only).

The HSMS Message Header is as follows:

SessionID — The SessionID must match the value of the SessionID of a previously sent Select.req to indicate the particular HSMS session that is ending. Subject to further specification by subsidiary standards.

Header Byte 2 = 0

Header Byte 3 = 0

PType = 0

SType = 3

System Bytes — A unique value among open transactions.

8.2.5 *SType=4: Deselect.rsp* — An HSMS message with SType 4 is a "Deselect Response" Control Message, used as the response to a Deselect.req Control message in the Deselect procedure for ending HSMS communications. The message format is as follows:

Message Length is always 10 (Header only).

The HSMS Message Header is as follows:

SessionID — must equal the session ID in the corresponding Deselect.req

Header Byte 2 = 0

Header Byte 3 — DeselectStatus. A code of zero indicates success of the Deselect operation. A non-zero code indicates failure.

Table 8

Value	*Description*
0	Communication Ended. The Deselect completed successfully.
1	Communication Not Established. HSMS communications has not yet been established with a select, or has already been ended with a previous Deselect.
2	Communication Busy. The session is still in use by the responding entity and so it cannot yet relinquish it gracefully. In this case, if the original requester must terminate communications, the separate procedure should be used as a last resort.
3–127	Reserved for subsidiary standard-specific reasons for Deselect failure.
128–255	Reserved for local entity-specific reasons for Deselect failure.

PType = 0

SType = 4

System Bytes — Equal to System Bytes in corresponding Deselect.req.

8.2.6 *SType=5: Linktest.req* — An HSMS message with SType 5 is a "Linktest Request" Control Message. It is used to verify the integrity of the HSMS Connection, or as a periodic heartbeat. The message format is as follows:

Message Length is always 10 (Header only).

The HSMS Message Header is as follows:

SessionID = 0xFFFF (in binary, all ones)

Header Byte 2 = 0

Header Byte 3 = 0

PType = 0

SType = 5

System Bytes — A unique value among open transactions.

8.2.7 *SType=6: Linktest.rsp* — An HSMS message with SType 6 is a "Linktest Response" Control Message, used as the response to a Linktest.req Control message in the Linktest Procedure. The message format is as follows:

Message Length is always 10 (Header only).

The HSMS Message Header is as follows:

SessionID = 0xFFFF (binary, all ones)

Header Byte 2 = 0

Header Byte 3 = 0

PType = 0

SType = 6

System Bytes — Equal to System Bytes in corresponding Linktest.req.

8.2.8 *SType=7: Reject.req* — An HSMS message with SType 7 is used in response to any valid HSMS message received which is not supported by the receiver of the message or which is not valid at the time. It is intended for dealing with attempts to use subsidiary standards or user-defined extensions which are not supported by the receiver (for example, SType equal to any value not defined in this standard). It must be used when an entity receives a control message which is a response (even numbered SType) for which there was no corresponding open transaction.

The HSMS Message Header is as follows:

SessionID — equal to the value of the Session ID in the message being rejected.

Header Byte 2 — For ReasonCode = PType Not Supported, equal to the PType in the message being rejected. Otherwise equal to the value of the SType in the message being rejected.

Header Byte 3 — reason code (always non-zero)

Table 9 ReasonCode

Value	*Description*
1	SType Not Supported. A message was received whose SType value not defined in the HSMS standard or the particular subsidiary standard(s) supported by the entity.
2	PType Not Supported. As above, but for PType.
3	Transaction Not Open. A Response control message was received when there was no outstanding request message which corresponded to it.
4	Entity Not Selected. A data message was received when not in the SELECTED state.
4–127	Reserved for subsidiary standard-specific reasons for reject.
128–255	Reserved for local entity-specific reasons for reject.

PType = 0

SType = 7

System Bytes — Equal to System Bytes in corresponding message being rejected.

8.2.9 *SType=9: Separate.req* — An HSMS message with SType = 9 is used to terminate HSMS communications immediately. With the exception of the SType value, it is identical to the Deselect.req message. Its purpose is to end HSMS communications immediately and without exception. No response is defined.

9 Special Considerations

9.1 *General Considerations*

9.1.1 *Communications Failures* — If a communications failure is detected, the entity should terminate the TCP/IP connection. Upon termination of the connection, the entity may, at this point, attempt to reestablish communications.

9.2 *TCP/IP Considerations*

9.2.1 *Connect Separation Time (T5)* — The connect procedures initiate some network activity. Frequent use of the active mode connect procedure to the IP Address and Port Number of an entity not yet ready to accept connections can be hostile to TCP/IP operations. The passive mode does not generate network activity and is not considered hostile to the network, although it may affect local application performance. An Entity initiating a connection in the active mode should limit its use of the connect procedure in a manner that is equivalent to the procedure described here.

After an active connect procedure terminates by any means (successfully or unsuccessfully), the Entity should not initiate another active connect procedure (for the same Remote Entity) until the T5 Connect Separation Time has elapsed. The separation of connect operations will be the sum of the T5 Connect Separation Time interval, plus the duration of the connect operation itself.

9.2.2 *NOT SELECTED Timeout (T7)* — Entry into the NOT SELECTED state is achieved either by state transition #2 (establishment of a TCP/IP connection). There is a time limit on how long an entity is required to remain in the NOT SELECTED state before either entering the SELECTED state or by returning to the NOT CONNECTED state.

Some entities, particularly those unable to accept more than a single TCP/IP connection, may be impaired in their operation by remaining in their NOT SELECTED state as they will be unavailable for communications with other entities. Such entities shall disconnect the TCP/IP connection (State Transition Event #3) if communication remains in the NOT SELECTED state for longer than the T7 timeout period.

9.2.3 *Network Intercharacter Timeout (T8)* — Because TCP/IP is a stream rather than a message protocol, it is possible that bytes which are all part of a single HSMS message may be transmitted in separate TCP/IP messages without any violation of the TCP/IP protocol. Since it is possible that these separate messages may be

separated by a substantial period of time, the Network Intercharacter Timeout (T8) is defined.

T8 is similar in purpose to the SECS-I T1 timer except that the communications issues which necessitate T8 are not entirely in the control of the sender of the message. Therefore, it is defined only in terms of the receiver of the message. In particular, if after receipt of a partial message, the T8 timeout period expires prior to receipt of the complete message, the receiving entity shall consider such case as a communications failure, as defined above.

9.2.4 *Multiple Connection Requests Directed to a Single Published Port* — Once a passive entity has accepted a connection on its published port, TCP/IP permits (though does not require) the entity to listen for and accept additional connections directed to the same published port.

HSMS permits (though does not require) entities to operate in this manner. However, for the purposes of HSMS compliance, each connection so formed must exhibit the behavior defined in the HSMS state diagram as if it were completely independent of any other connection to the same published port.

9.2.4.1 *Rejection of Additional Connection Requests by a Passive Mode Entity* — A passive mode entity unable to service more than a single TCP/IP connection for HSMS communications will follow one of these three procedures with respect to additional connection requests.

a. Accept the connection, but always respond to any subsequent HSMS select procedures with the Communication Already Active response code. For the purpose of the HSMS State Diagram, the connect procedure terminates successfully (enters CONNECTED state), but HSMS communications are never established (remain in NOT SELECTED substate). This is the preferred option in that it can provide the most information to the remote entity as to why the connection is refused (see HSMS Select Procedure), but places an addition implementation requirement on the local entity.

b. Actively reject the connection request. This can be done in a TLI implementation using the t_snddis procedure. This will cause the connect procedure in the remote entity to terminate unsuccessfully. This option may not be available to all implementations because some API's, notably some implementations of BSD Sockets, do not provide for initiating an active reject. Note, however, that all TCP/IP implementations, including BSD Sockets, properly respond to an active reject from the remote entity.

c. Refuse to listen for or accept the connect request. No action is taken in the local entity: the remote entity's connect procedure will eventually time out. This option is permitted, but not recommended, as it can cause considerable delay on the part of the remote entity. However, it may be the only alternative available to implementations with network resource limitations.

The documentation of the passive local entity shall indicate which means it uses to refuse connections.

9.3 *HSMS-Specific Considerations*

9.3.1 *Control Transactions T6 Control Timeout* — A number of the control messages are part of procedures which require a message exchange or transaction: <xx>.req from the initiator of the control service, followed by an <xx>.rsp from the receiver of the <xx>.req in response to it. A control transaction is considered open from the time the <xx>.req request is sent until the time the <xx>.rsp is received.

The time a control transaction may remain open is subject to the T6 control transaction timeout. Upon initiation of a control transaction, the local entity should set a timer whose duration is equal to the T6 timeout value. If the transaction is properly closed prior to the expiration of the timer, the timer should be canceled. If the timer expires prior to the proper closing of the transaction, the transaction shall be considered closed by the initiator and considered an HSMS communications failure.

9.3.2 *Procedures and "Stateless" Transactions* — Most of the HSMS control procedures involve a transaction: the initiator sends a request message to the responding entity and waits for a response message. The responding entity receives the initiator' s request message and sends a reply.

Note that such transactions are "stateless" in the following sense: while the initiator of a transaction is waiting for a response, it may receive a message other than that response, and this message may be any message valid for the state the initiator was in at the time the original transaction was initiated. For example, the two entities may simultaneously initiate transactions. As a result, no states for "TRANSACTION OPEN" or "TRANSACTION NOT OPEN" are reflected in the HSMS state machine. The use of such state information in an implementation is strictly a local entity-specific issue.

9.3.3 *Alternative Message Types and Header Byte Values* — The HSMS standard does not completely define all possible enumerated values of either the PType or SType field. Further, Header bytes 2 and 3 have a format determined by the PType for messages

whose SType is equal to 0, but is otherwise specified for all other SType values. The message text formatting is defined by the PType as well, but only for data messages.

Subsidiary standards must be consistent with this convention. In particular, for SType = 0, subsidiary standards defining PType values not equal to 0 may specify both the message text encoding and the interpretation of header bytes 2 and 3. For STypes not equal to 0 but otherwise specified in this standard, PType must = 0, and no message text may be transmitted. For STypes defined in subsidiary standards, the meaning of header bytes 2 and 3 may be specified on a per SType value basis, and these STypes may optionally define message text as long as the PType field is used in a manner consistent with the preceding paragraph.

9.4 *SECS-II Considerations* — The SECS-II standard (SEMI E5) makes certain references to SECS-I (SEMI E4). This section addresses issues specific to SECS-II when HSMS is used to transport SECS-II messages.

9.4.1 *Reply Matching* — When a Sender sends a Primary Message with W-Bit 1 (Reply Expected), the Sender should expect a Reply message whose header meets the following requirements.

The SessionID of the Reply must match the SessionID of the Primary Message.

The Stream of the Reply must match the Stream of the Primary Message.

The Function of the Reply must be one greater than the Function of the Primary Message, or else the Function of the Reply must be 0 (Function Zero Reply).

The System Bytes of the Reply must match the System Bytes of the Primary Message.

9.4.1.1 *T3 Reply Timeout* — The T3 reply timeout is a limit on the length of time that the HSMS message protocol is willing to wait for a Reply message.

After sending a Primary Message with W-bit 1 (Reply Expected), the sender must begin a reply timer, initialized to the T3 value. If the sender does not receive the Reply Message before the reply timer expires, then a T3 Timeout Error has occurred. The sender should close the transaction and no longer expect the Reply Message.

Each open transaction for which a Reply is expected requires a separate reply timer.

9.4.2 *Stream 9 Messages* — The SECS-II standard defines error messages S9F1, S9F3, S9F5, S9F7, S9F9, and S9F11, with message text containing the SECS-II Data Items MHEAD or SHEAD, which are defined to contain a 10-byte SECS-I block header.

When using SECS-II with HSMS, MHEAD and SHEAD should contain the ten bytes of the HSMS Message Header.

10 HSMS Documentation

An HSMS implementation is required to document the following information:

1. Method for setting protocol parameters (see Section 10.1).
2. Range allowed and resolution for each parameter.
3. The option used for refusing incoming connection requests if the implementation uses the passive mode for TCP/IP connection establishment.
4. Maximum message size which can be received.
5. Maximum expected size of messages sent.
6. Maximum number of supported concurrent open transactions.

10.1 *Parameter Setting* — Implementations of HSMS must provide for installation time setting of the following parameters. The range and resolution of all parameters must be at least as shown in the table. All parameters must be stored in such a manner that the settings will be retained if the power fails or if the system software is reloaded.

Table 10

Parameter Name	*Value Range*	*Resolution*	*Typical Value*	*Description*
T3 Reply Timeout	1-120 seconds	1 second	45 seconds	Reply timeout. Specifies maximum amount of time an entity expecting a reply message will wait for that reply.
T5 Connect Separation Timeout	1-240 seconds	1 second	10 seconds	Connection Separation Timeout. Specifies the amount of time which must elapse between successive attempts to connect to a given remote entity.
T6 Control Transaction Timeout	1-240 seconds	1 second	5 seconds	Control Transaction Timeout. Specifies the time which a control transaction may remain open before it is considered a communications failure.
T7 NOT SELECTED Timeout	1-240 seconds	1 second	10 seconds	Time which a TCP/IP connection can remain in NOT SELECTED state (i.e., no HSMS activity) before it is considered a communications failure.
T8 Network Intercharacter Timeout	1-120 seconds	1 second	5 seconds	Maximum time between successive bytes of a single HSMS message which may expire before it is considered a communications failure.
Connect Mode	PASSIVE, ACTIVE	—	—	Connect Mode. Specifies the logic this local entity will use during HSMS connection establishment.
Local Entity IP Address and Port number	determined by TCP/IP conventions	—	—	Required for any entity operating in PASSIVE mode. Determines the address on which the local entity will listen for incoming connection requests.
Remote Entity IP Address and Port Number	determined by TCP/IP conventions	—	—	Required for any entity operating in ACTIVE mode. Determines the address of the remote entity to which the local entity will attempt to connect.

NOTE: Parameter defaults shown above are for small networks (10 nodes or less). Settings may need to be adjusted for larger network configurations.

APPENDIX 1

NOTE: This appendix was approved as a part of SEMI E37 by full letter ballot procedure.

A1-1 TCP/IP Procedures Using TLI and BSD Socket Interfaces

A1-1.1 *Passive Mode Connect Procedure*

Table 1

Intended Action	*TLI Construct*	*BSD Construct*	*Comment*
Obtain a connection endpoint and bind it to a published port.	tep = t_open(...) t_bind(tep,...)	skt = socket(...) bind(skt,...)	BSD refers to a connection endpoint as a "socket." TLI refers to it as a TEP (transport end point).
Permit socket to listen for connections.	...	listen(skt...)	In TLI, the equivalent of BSD listen is not necessary.
Connect procedure: receive incoming connect request and accept it.	t_listen(tep,...) t_accept(tep,...)	accept(skt,...)	
Connect procedure: receive incoming connect request, but reject it.	t_listen(tep,...) t_snddis(tep,...)	...	The BSD API does not support originating a reject of a connect request, as receiving request and accepting it are a single operation.

A1-1.2 *Active Mode Connect Procedure*

Table 2

Intended Action	*TLI Construct*	*BSD Construct*	*Comment*
Obtain a connection endpoint.	tep = t_open(...) t_bind(tep,...)	skt = socket(...)	TLI requires bind to null address for active entity.
Connect procedure: send connect request and receive accept or reject from passive entity.	t_connect(tep,...) t_rcvconnect(tep,...)	connect(skt,...)	The BSD connect will correctly handle an active reject from the TLI-based remote entity.

A1-1.3 *Terminating the Connection*

Table 3

Intended Action	*TLI Construct*	*BSD Construct*	*Comment*
Release the connection and free connection endpoint.	t_sndrel(tep,..) t_close(tep)	close(skt)	The "gracefulness" of the BSD close is a function of the local implementation.
Disconnect and free the connection endpoint.	t_snddis(tep,..) t_close(tep)	shutdown(skt,2) close(skt)	Shutdown immediately disables further sends and receives if the second arg = 2.

A1-1.4 *Sending and Receiving HSMS Messages*

Table 4

Intended Action	*TLI Construct*	*BSD Construct*	*Comment*

Send an HSMS message	hdr->Len = length;
	t_snd(tep,hdr,14,0);
	t_snd(tep,Text,hdr->Len,0);
		hdr->Len = length;
		write(skt,hdr,14);
		write(skt,Text,hdr->Len);
			The procedure illustrates a "typical" implementation style in which the length bytes and header are combined into a single 14-byte item sent first, followed by the text. This is not to imply that combining everything is not permitted.

Receive and HSMS message	t_rcv(tep,hdr,14,...); t_rcv(tep,Text,hdr->Len,...);	read(skt,hdr,14); read(skt,Text,hdr->Len);	As above, but for receiving.

A1-2 HSMS Scenarios

The following scenarios are provided to illustrate the HSMS procedures as used for a typical complete session. The terminology, procedure names, and message names are further explained in the remainder of this document. Also note that either entity may initiate the HSMS Select procedure, the Deselect or Separate procedures, and HSMS Data Messages and Transactions. For convenience, the scenarios show the left-hand entity as the initiator of all transactions.

A1-2.1 *Begin HSMS Communication* — This scenario illustrates the TCP/IP connection procedure, an HSMS select procedure, and exchange of data messages. Note that the data message activity for TCP is for illustrative purposes only. In fact, the actual network activity can vary. For example, even if the data messages are sent as separate calls to t_snd (or write) as shown, the TCP/IP implementation may buffer the header and transmit it in a single packet with the text, or the text may be split into multiple packets.

Table 5

BSD API Calls	*TLI API Calls*	*Network Activity*	*TLI API Calls*	*BSD API Calls*
Prepare to initiate a connection request.			Prepare to receive a connection request.	
skt = socket(...);	tep = t_open(...); t_bind(tep, ...);		tep = t_open(...); t_bind(tep,...);	skt = socket(...);bind(skt,...); listen(skt,...);
Initiate a connection request and wait for response.			Receive a connection request, accept it, and send response.	
connect(skt,...);	t_connect(tep,...); t_rcvconnect(tep,...);	TCP/IP Connect Req Msg(s) → ← TCP/IP Accept Msg(s)	t_listen(tep,...);t_accept(tep,...);	accept(skt,...);
Initiate an HSMS Select procedure: send request and receive response.			Respond to HSMS select procedure: receive request and send response.	
write(skt,hdr,14); read(skt,hdr,14);	t_snd(tep,hdr,14,0); t_rcv(tep,hdr,14,...);	Select.req Message → ← Select.rsp Message	t_rcv(tep,hdr,14,...); t_snd(tep,hdr,14,0);	read(skt,hdr,14); write(skt,hdr,14);
Send an HSMS data message as length bytes and header, followed by Text.			Receive an HSMS data message as length bytes and header, followed by Text.	

hdr->Len = Length;
write(skt,hdr,14);
write(skt,Text,...);
	hdr->Len = Length;
	t_snd(tep,hdr,14,0);
	t_snd(tep,Text,...);
		HSMS Data Message (hdr) →
		HSMS Data Message (text) →
			t_rcv(tep,hdr,14,...);
			t_rcv(tep,Text,...);
				read(skt,hdr,14);read(skt,Text,...);

A1-2.2 *Ending Communicaiton Using Deselect* — This scenario illustrates ending an HSMS Session using the Deselect procedure to end the HSMS session.

BSD API Calls	*TLI API Calls*	*Network Activity*	*TLI API Calls*	*BSD API Calls*
Send the Deselct.req and receive Deselect.rsp.			Receive the Deselect.req and send the Deselect.rsp.	
write(skt,hdr,14); read(skt,hdr,14);	t_snd(tep,hdr,14,0); t_rcv(tep,hdr,14,...);	Deselect.req Message → ← Deselect.rsp Message	t_rcv(tep,hdr,14,...); t_snd(tep,hdr,14,0);	read(skt,hdr,14); write(skt,hdr,14);
Disconnect the TCP/IP Connection.			Respond to Disconnect of connection.	
shutdown(skt,2); close(skt);	t_snddis(tep; t_close(tep);	TCP/IP Disconnect Msg(s) ↔	t_rcvdis(tep,...);	close(skt);

A1-3 HSMS Alternating Mode Connect Procedure

Some users have particular requirements which prevent them from determining which connect mode (active or passive) a given entity will use at any particular time. In such a case, a Local Entity alternately attempts the active mode and passive mode connect procedure until a connection is successfully established. Note that this requires that local entity provide a published port when in the passive phase. The general logic sequence at the Alternating Local Entity is as follows:

1. Attempt an active connect procedure as described in Section 6.3.3 using a timeout value for the t_rcvconnect greater than or equal to the connection separation timeout T5.
2. If the active connect procedure completes successfully, the alternating mode connect procedure completes successfully.
3. If the active connect procedure terminates unsuccessfully, attempt the passive connect procedure as described in 6.3.2 with the timeout for the t_listen greater than or equal to the connection separation timeout T5.
4. If the passive connect procedure completes successfully, the alternating mode connect procedure completes successfully as described in 6.3.2.
5. If the passive connect procedure terminates unsuccessfully, the local entity may either return to step 1 to continue the alternating mode procedure or terminate unsuccessfully. The number of times the above sequence of steps are repeated in attempting to form a connection is a local entity-specific issue.

A1-3.1 *Alternating Mode Cycle Time* — The Alternating Mode Cycle Time is the time between iterations of the Connect Procedure of an Alternating Mode entity. In the above procedure, this corresponds to the duration between the initiation of step 1 and completion of step 5 immediately prior to the reinitiation of step 1. This time is implementation-dependent.

In the case that two entities are both using the Alternating Mode Connect Procedure, it is desirable to ensure that they both have different alternating mode cycle times, to prevent the entities from attempting to connect in lock step: both in active mode, then both in passive mode. Adjusting the Alternating Mode Cycle Time can be readily achieved by adjustment of T5 so that the cycle time is different for the two entities:

A1-3.2 *HSMS Connect Combinations* — An entity configured as alternating between active and passive mode can connect with either an passive or active mode remote entity. The list below summarizes the combinations possible using the standard with this particular connections strategy.

1. An Entity “A” configured as ACTIVE can connect to an Entity “B” configured as PASSIVE or as ALTERNATING, and Entity A always establishes the Connection.
2. An Entity “A” configured as ALTERNATING can connect to an Entity “B” configured as PASSIVE, and Entity “A” always establishes the Connection.
3. An Entity “A” configured as ALTERNATING can connect to an Entity “B” configured as ALTERNATING, and either end can establish the Connection. In implementations which use multi-threaded connect logic, rather than the sequential logic described in this document, it may be possible that both ends of the HSMS connection attempt to connect at the same time. In this case, there can be two separate TCP/IP connections established, and it is necessary to establish a convention so that one connection is allowed to remain and the other is terminated.
4. It is not allowed to connect two Entities both configured as PASSIVE, or both configured as ACTIVE.

A1-4 Non-HSMS TCP/IP Protocols

For typical TCP/IP implementations, HSMS can co-exist with other TCP/IP based protocols on the same IP Address. This can be very useful. For example, a SECS-II message transaction could trigger an application to begin a TCP/IP FTP (File Transfer Protocol) sequence to transfer a large data file.

A1-5 Non-TCP/IP Protocols

The use of protocols other than TCP/IP on the same network as the HSMS entities is possible but beyond the scope of this standard. Typically, other protocols could be used, provided they have no impact on TCP/IP or HSMS entities on the network.

A1-6 Multiple LANs

The HSMS specification considers only a single TCP/IP LAN. Interconnecting multiple LANs is outside the scope of HSMS. However, since TCP/IP implementations typically support such configurations seamlessly through gateways, routers, and similar entities, it may be possible to establish an HSMS Connection across interconnected LAN’s.

A1-7 TCP/IP Physical Layer

HSMS does not specify the physical layer of IP. Any physical layer supported by TCP/IP can be used. Most commonly, TCP/IP implementations use Ethernet (IEEE 802.3) as the physical layer. However, some TCP/IP implementations use other protocols (e.g., Token Bus, IEEE 802.4 and .5). To ensure interoperability within a given installation, it may be desirable to establish additional local standards for the physical layer.

A1-8 Well-Known TCP/IP Port Numbers

Some TCP/IP-based protocols specify a particular "Well Known" TCP Port Number, which is published and is not available for other protocols. HSMS does not specify a particular "Well Known" TCP Port Number, but instead requires that it be configurable. The IETF defines "Assigned Well Known Port Numbers" in RFC 1340.

A1-9 Delay between Disconnect and Re-Connect

Some TCP/IP systems exhibit problems with applications which terminate a connection and then quickly re-connect, when using identical TCP ports on both ends of the connection. When using such systems, it may be advisable to delay after disconnect before re-connecting. The delay time varies among TCP/IP implementations, but typically can be calculated as twice the "Maximum Segment Lifetime" or MSL. For example, most TCP/IP systems based on BSD (e.g., Sun, AIX) use a MSL of 30 seconds, so a delay of 60 seconds would be appropriate. If rapid connects are required, your application should use a different Port, if allowed by the Connect Mode you are using. In some TCP/IP systems, this problem does not occur.

A1-10 User-Defined Message Types

It is recognized that equipment suppliers may find it desirable to develop additional features not found in the base level HSMS or any defined subsidiary standards. This will be the case during the testing and development of any proposed new subsidiary standard. User-defined extensions through new message types are permissible as long as they are confined to intra-vendor communication interfaces: any inter-vendor communications interface which requires the use of such extensions is considered to be noncompliant with the HSMS standard.

If a supplier does deem it necessary to extend or otherwise go outside the standard, the use of "reserved, not used," values of PType and SType may simplify their implementation by permitting the reuse of the HSMS implementation rather than the implementation and use of a completely separate parallel standard. By remaining within the "reserved, not used," ranges for SType and PType, the implementor can be assured that future subsidiary standards which define new values for SType and/or PType will not conflict with user-defined extensions.

A1-11 Comparison of SECS-I and HSMS

The following table compares major features of SECS-I and HSMS.

Feature	*SECS-I*	*HSMS*
Communications Protocol Base	RS-232	TCP/IP
Physical Layer	25-pin connector and 4-wire serial cable	Physical layer not defined. HSMS allows any TCP/IP supported physical medium. Typical example is Ethernet (IEEE 802.3) and thin coax (10-BASE-2).
Communications Speed	Typically about 1000 bytes/second (assuming 9600 baud).	Typically 10 MBits/second (assuming typical Ethernet).
Connections	One physical RS-232 cable per SECS-I connection.	One physical network cable can support many HSMS Connections.
Message Format	Message text is SECS-II Data Items. Transmits a SECS-II message as a series of transmittal blocks each approximately 256 bytes in size. Each block has a one-byte block length, a ten-byte Block Header, text, and a two-byte Checksum.	Message Text is SECS-II Data Items. Transmits a SECS-II message as a TCP/IP byte stream. The message has a four-byte Message Length, a ten-byte Message Header, and text. The TCP/IP layer may impose blocking limits which depend on the physical layer used, but this blocking is transparent to the TCP/IP API and is outside the scope of HSMS.

Feature	*SECS-I*	*HSMS*
Header	Ten-byte header on each block of a message. Header bytes 4-5 contains E-Bit and Block Number.	One ten-byte Header for the entire message. Header bytes 4–5 contain PType and SType. Header bytes 2–3 are W-Bit, Stream, and Function when SType = 0 (Data Message). For SType not equal to 0 (Control Message), bytes 2–3 have other uses. No R-Bit.
Maximum message size	Limited to approximately 7.9 million bytes (32767 blocks times 244 text bytes per block).	Message size limited by 4-byte message length (approximately 4 GBytes). Local implementation of TCP/IP and HSMS may further limit this in practice.
Protocol Parameters (Common)	T3 Reply Timeout Device ID	T3 Reply Timeout Session ID (analogous to Device ID).
Protocol Parameters (SECS-I only)	Baud Rate T1 Inter-Character Timeout T2 Block Protocol Timeout T4 Inter-Block Timeout RTY Retry Count Host/Equipment	Not used in HSMS. Corresponding issues addressed by TCP/IP layers.
Protocol Parameters (HSMS Only)	Not needed by SECS-I.	IP Address and Port of Passive Entity. T5 Connect Separation Timeout. T6 Control Transaction Timeout. T7 NOT SELECTED Timeout. T8 Network Intercharacter Timeout.

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 E37.1-0702
HIGH-SPEED SECS MESSAGE SERVICE SINGLE SELECTED-SESSION MODE (HSMS-SS or HSMS-SSS)

This standard was technically approved by the Global Information and Control Committee and is the direct responsibility of the Japan Information and Control Committee. Current edition approved by the Japan Regional Standards Committee on April 26, 2002. Initially available at www.semi.org June 2002; to be published July 2002. Originally published in 1995; previoulsy published in 1996.

1 Purpose

1.1 HSMS-SS provides a means for independent manufacturers to produce implementations which can be connected without requiring specific knowledge of one another.

1.2 HSMS-SS is intended as an alternative to SEMI E4 (SECS-I) for applications where higher speed communication is needed.

1.3 HSMS-SS is intended as an alternative to SEMI E13 (SECS Message Services) for applications where TCP/IP is preferred over OSI as a communications basis.

2 Scope

2.1 High-Speed SECS Message Services Single-Session Mode (HSMS-SS) is a subsidiary standard to High-Speed SECS Message Services (HSMS) Generic Services.

2.2 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.

3 Reference Standards

NOTE 1: Unless otherwise indicated, all documents cited shall be the latest published versions.

3.1 *SEMI Standards*

SEMI E4 — SEMI Equipment Communication Standard 1 Message Transfer (SECS-I)

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

SEMI E37 — High-Speed SECS Message Services (HSMS) Generic Services

4 Terminology

4.1 *Definitions*

4.1.1 *device ID* — a 15-bit field in the message header used to identify a subentity within the equipment.

4.2 In addition, all definitions for HSMS Generic Services apply.

4.3 Note that the terms HSMS and HSMS generic services both refer to the HSMS Generic Services standard definition (SEMI E37).

5 HSMS-SS Overview and State Machine

5.1 This definition defines the HSMS-SS-specific use of HSMS Generic Services suitable for applications requiring a simple SECS-I replacement. The purpose of this standard is to explicitly limit the capabilities of the HSMS Generic Services to the minimum necessary for this type of application. Specifically, HSMS imposes the following limitations:

1. HSMS-SS eliminates the use of a number of HSMS procedures. Deselect is not to be used to end HSMS-SS communications (use Separate instead), and the Reject procedure is optional.
2. HSMS-SS limits certain other procedures such as Select to simplify operation for the specific case of SECS-I replacement.

5.2 The remainder of this document describes these limitations in more detail.

5.3 *HSMS-SS State Machine* — The HSMS-SS behavior and state machine differ from that specified in the HSMS Generic Services in the following ways:

1. The SelectionCounter defined in HSMS Generic Services is not required.
2. Various transitions are defined differently as illustrated in the HSMS-SS state machine illustrated below.