SEMI E37-0298
HIGH-SPEED SECS MESSAGE SERVICES (HSMS) GENERIC SERVICES

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.

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

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

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 transmission 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.

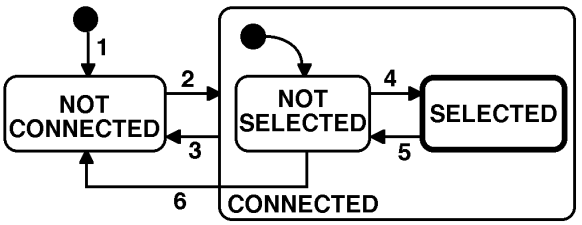
   In HSMS, since it uses the dynamic connection environment of TCP/IP, the TCP/IP connection is logically broken via a release or a disconnect procedure without any physical disconnect from the network medium.

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*

#	*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	NOT CONNECTED	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.

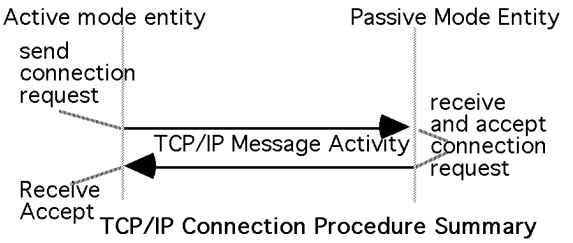
Conventions have been established for selecting TCP Port Numbers which are outside the scope of the HSMS protocol. Consult RFC 793, Transmission Control Protocol (TCP) in Section 3.

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

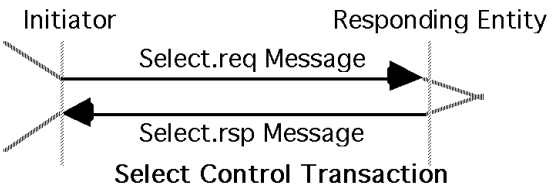
HSMS defines the procedures for all message exchange between entities across the TCP/IP connection established according to the procedures in the previous section. As explained in the overview, once the

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").

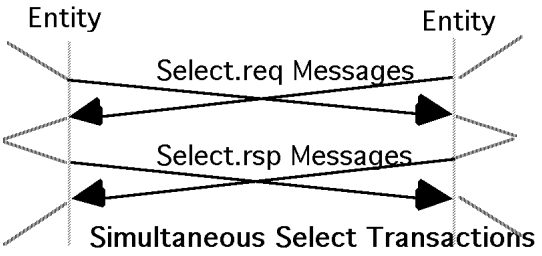
7.2.1 *Initiator Procedure* — The procedure followed by the initiator is as follows.

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.

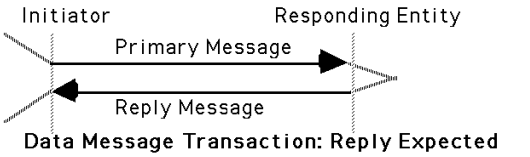


7.3 *Data Procedure* — HSMS data messages may be initiated by either entity as long as the connection is in the SELECTED state. Receipt of a data message when not in the SELECTED state will result in a reject procedure (see Section 7.7).

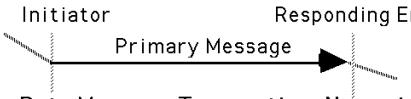
Data messages may be further defined as part of a data transaction as either a "Primary" or "Reply" data message. In a data transaction, the initiator of the transaction sends a primary message to the responding entity. If the Primary message indicates that a reply is expected, a Reply message is sent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



Data Message Transaction: Reply Expected

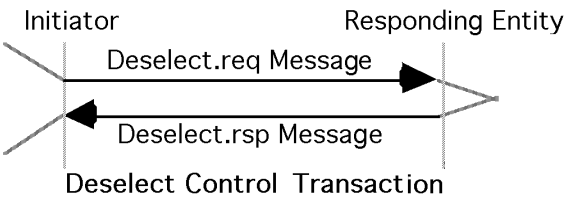


Data Message Transaction: No reply expected

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.



Deselect Control Transaction

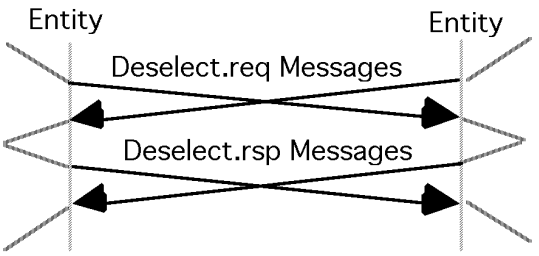
7.4.1 *Initiator Procedure*

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.

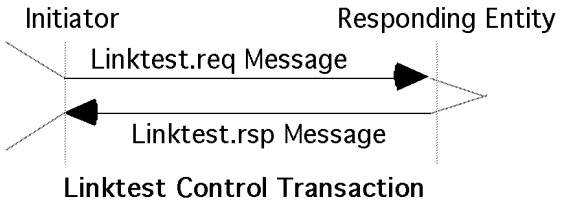
7.4.3 *Simultaneous Deselect Procedures* — If the subsidiary standards do not restrict the use of the Deselect, it is possible that both entities simultaneously initiate Deselect Procedures with identical SessionID's. In such a case, each entity will accept the other entity's Deselect request by responding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

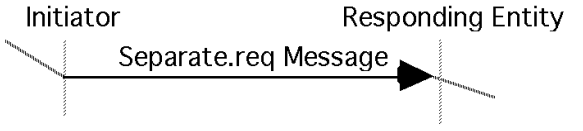
7.5.1 *Initiator Procedure*

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

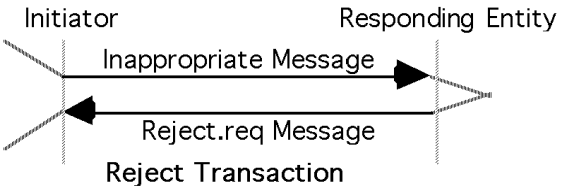
7.6.1 *Initiator Procedure*

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:

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:

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."

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."

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 Sel-ect.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).

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: Unique Reply: 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

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*
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: 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.

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.

SelectStatus

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.

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)

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 CON-

NECTED 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 theT6 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.

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*

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*

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*

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 1.

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.

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_rcv-connect 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 coexist 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.
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.

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.