8.2.13 *GetActiveSessions Operation*

8.2.13.1 *XML Schema Types*

8.2.13.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the GetActiveSessions operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.2.13.1.2 *GetActiveSessions Request*

8.2.13.1.2.1 The service request is mapped to the XML Schema element GetActiveSessionsRequest of type GetActiveSessionsRequestType. There are no input parameters for this operation.

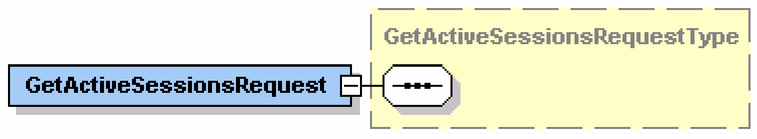

Figure 25
GetActiveSessionsRequest XML Schema Element

8.2.13.1.3 *GetActiveSessions Response*

8.2.13.1.3.1 The service response is mapped to the XML Schema element GetActiveSessionsResponse of type GetActiveSessionsResponseType. Table 40 shows how the output parameters are mapped to XML. The XML Schema diagram for GetActiveSessionsResponse is shown in Figure 26. The ActiveSession class is described in ¶8.2.13.1.4. Errors for this operation are returned using the XML Schema element Error of type GetActiveSessionsErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 40 Translation Table for GetActiveSessions Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
activeSessions	Unordered list	Element	ActiveSession: ActiveSessionType (one or more)
error	Structured data	Element	Error: GetActiveSessionsErrorType

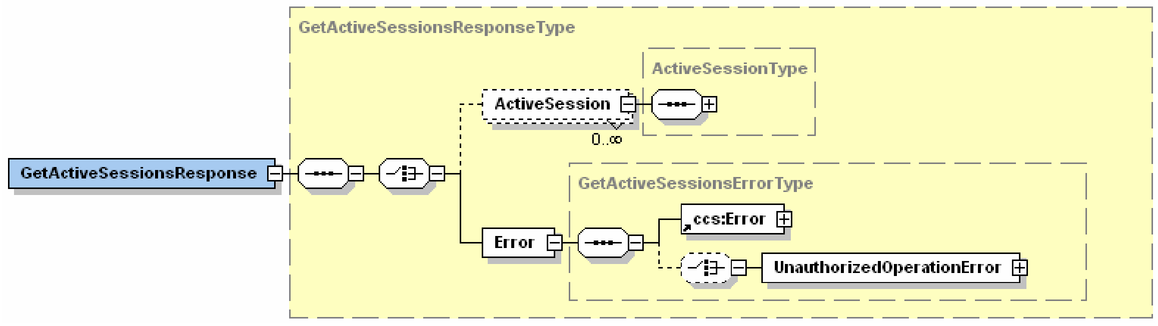
GetActiveSessionsResponseType
ActiveSessionType
ActiveSession
0..∞
GetActiveSessionsResponse
GetActiveSessionsErrorType
ccs:Error
Error
UnauthorizedOperationError

Figure 26
GetActiveSessionsResponse XML Schema Element

8.2.13.1.4 *ActiveSession Class*

8.2.13.1.4.1 The ActiveSession class is mapped to the XML ActiveSessionType. Table 41 shows how the attributes and associations of ActiveSession class are mapped to XML. The XML Schema diagram is shown in Figure 27. The association with the EndPoint class is modeled as an XML element of type EndPointSelectorType, and currently has but one choice defined for allowed derived class, the HTTPEndPointType which contains just an URL.

Table 41 Translation Table for Subject Class

Attribute or Role Name	*UML Name/Type*	*XML Element or Attribute*	*XML Name/Type*
sessionId	Text	Element	SessionID: xsd:string
isPersistent	Boolean	Element	IsPersistent: xsd:boolean
clientId	Text	Element	ClientID: xsd:string
sessionEndPoint	Association	Element	SessionEndPoint: EndPointSelectorType

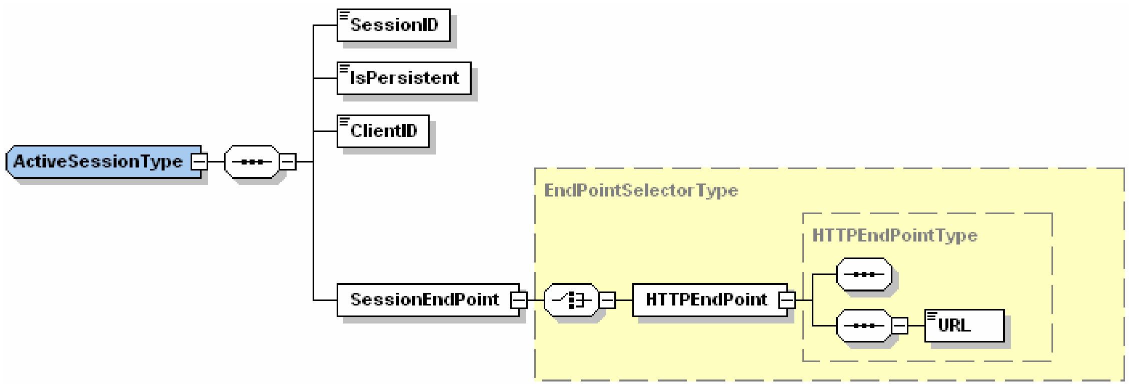

Figure 27
ActiveSession XML Schema Type

8.2.13.1.5 *EndPoint Class*

8.2.13.1.5.1 The EndPoint class is mapped to the XML EndPointType. EndPoint class is an abstract base class and there are no attributes or roles defined for this class. Associations with the EndPoint class are modeled as XML elements of type EndPointSelectorType, and currently has but one choice defined for allowed derived class, the HTTPEndPointType which contains just an URL.

8.2.13.1.6 *HTTPEndPoint Class*

8.2.13.1.6.1 The HTTPEndPoint class is mapped to the XML HTTPEndPointType. Table 42 shows how the attributes and associations of HTTPEndPoint class are mapped to XML. HTTPEndPointType extends EndPointType.

Table 42 Translation Table for HTTPEndPoint Class

Attribute or Role Name	*UML Name/Type*	*XML Element or Attribute*	*XML Name/Type*
url	Text	Element	URL: xsd:anyURI

8.2.13.2 *WSDL Message(s)*

8.2.13.2.1 Three messages are defined for this operation corresponding to the E132Header, Request and Response messages. See the message definitions for "E132HeaderMessage", "GetActiveSessionsRequestMessage", and "GetActiveSessionsResponseMessage" in E132-1-V0305-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 43.

Table 43 GetActiveSessions Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header GetActiveSessionsRequestMessage → auth:GetActiveSessionsRequest GetActiveSessionsResponseMessage → auth:GetActiveSessionsResponse

8.2.13.3 *WSDL Operation*

8.2.13.3.1 See the portType operation definition for “GetActiveSessions” in E132-1-V0305-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 44.

Table 44 GetActiveSessions PortType Operation

Input Message Name	GetActiveSessionsRequestMessage
Output Message Name	GetActiveSessionsResponseMessage

8.2.13.4 *WSDL Operation Binding*

8.2.13.4.1 See the operation binding for “GetActiveSessions” in E132-1-V0305-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 45.

Table 45 GetActiveSessions Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:GetActiveSessions
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.2.14 *SetMaxSessions Operation*

8.2.14.1 *XML Schema Types*

8.2.14.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the SetMaxSessions operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.2.14.1.2 *SetMaxSessions Request*

8.2.14.1.2.1 The service request is mapped to the XML Schema element SetMaxSessionsRequest of type SetMaxSessionsRequestType. Table 46 shows how the input parameters are mapped to XML.

Table 46 Translation Table for SetMaxSessions Input Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
maxSessions	Integer	Element	MaxSessions: xsd:int

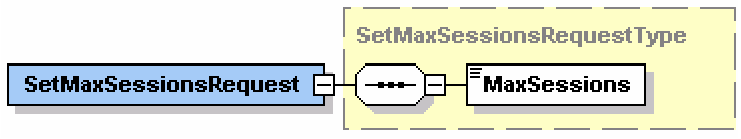

Figure 28
SetMaxSessionsRequest XML Schema Element

8.2.14.1.3 *SetMaxSessions Response*

8.2.14.1.3.1 The service response is mapped to the XML Schema element SetMaxSessionsResponse of type SetMaxSessionsResponseType. Table 47 shows how the output parameters are mapped to XML. The XML Schema diagram for SetMaxSessionsResponse is shown in Figure 29. Errors for this operation are returned using the XML Schema element Error of type SetMaxSessionsErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 47 Translation Table for SetMaxSessions Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
sessionCount	Integer	Element	SessionCount: xsd:int
error	Structured data	Element	Error: SetMaxSessionsErrorType

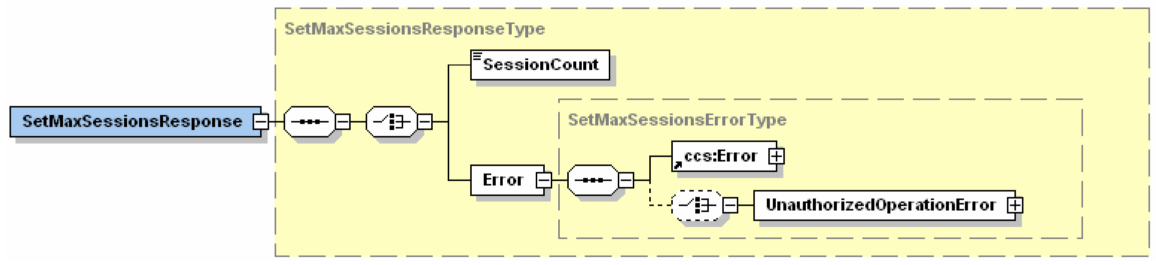

Figure 29
SetMaxSessionsResponse XML Schema Element

8.2.14.2 *WSDL Message(s)*

8.2.14.2.1 Three messages are defined for this operation corresponding to the E132 Header, Request and Response messages. See the message definitions for "E132HeaderMessage", "SetMaxSessionsRequestMessage", and "SetMaxSessionsResponseMessage" in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 48.

Table 48 SetMaxSessions Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header SetMaxSessionsRequestMessage → auth: SetMaxSessionsRequest SetMaxSessionsResponseMessage → auth: SetMaxSessionsResponse

8.2.14.3 *WSDL Operation*

8.2.14.3.1 See the portType operation definition for "SetMaxSessions" in E132-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 49.

Table 49 SetMaxSessions PortType Operation

Input Message Name	SetMaxSessionsRequestMessage
Output Message Name	SetMaxSessionsResponseMessage

8.2.14.4 *WSDL Operation Binding*

8.2.14.4.1 See the operation binding for "SetMaxSessions" in E132-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 50.

Table 50 SetMaxSessions Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:SetMaxSessions
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.2.15 *GetMaxSessions Operation*

8.2.15.1 *XML Schema Types*

8.2.15.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the GetMaxSessions operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.2.15.1.2 *GetMaxSessions Request*

8.2.15.1.2.1 The service request is mapped to the XML Schema element GetMaxSessionsRequest of type GetMaxSessionsRequestType. There are no input parameters for this operation.

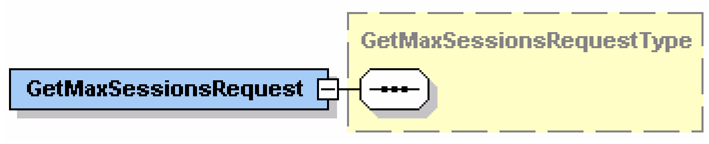

Figure 30
GetMaxSessionsRequest XML Schema Element

8.2.15.1.3 *GetMaxSessions Response*

8.2.15.1.3.1 The service response is mapped to the XML Schema element GetMaxSessionsResponse of type GetMaxSessionsResponseType. Table 51 shows how the output parameters are mapped to XML. The XML Schema diagram for GetMaxSessionsResponse is shown in Figure 31. The MaxSessions and SessionCount output arguments are contained in the GetMaxSessionsResponseBodyType, this is to allow for better compatibility across WSDL compilers. Errors for this operation are returned using the XML Schema element Error of type GetMaxSessionsErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 51 Translation Table for GetMaxSessions Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
maxSessions	Integer	Element	MaxSessions: xsd:int
sessionCount	Integer	Element	SessionCount: xsd:int
error	Structured data	Element	Error: GetMaxSessionsErrorType

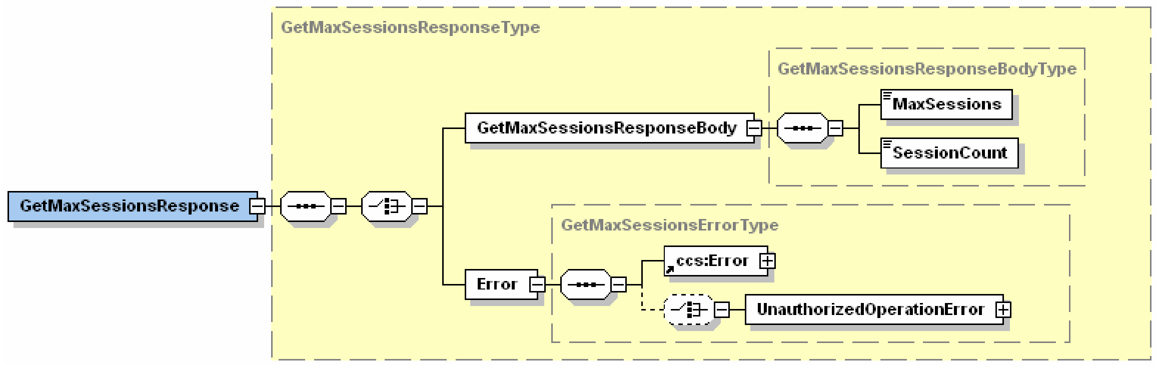
GetMaxSessionsResponseType
GetMaxSessionsResponseBodyType
MaxSessions
SessionCount
GetMaxSessionsResponseBody
GetMaxSessionsResponse
GetMaxSessionsErrorType
ccs:Error
Error
UnauthorizedOperationError

Figure 31
GetMaxSessionsResponse XML Schema Element

8.2.15.2 *WSDL Message(s)*

8.2.15.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “GetMaxSessionsRequestMessage”, and “GetMaxSessionsResponseMessage” in E132-1-V0305-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 52.

Table 52 GetMaxSessions Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header GetMaxSessionsRequestMessage → auth:GetMaxSessionsRequest GetMaxSessionsResponseMessage → auth:GetMaxSessionsResponse

8.2.15.3 *WSDL Operation*

8.2.15.3.1 See the portType operation definition for “GetMaxSessions” in E132-1-V0305-SecurityAdmin-PortType.wsdl. Key information is shown for convenience in Table 53.

Table 53 GetMaxSessions PortType Operation

Input Message Name	GetMaxSessionsRequestMessage
Output Message Name	GetMaxSessionsResponseMessage

8.2.15.4 *WSDL Operation Binding*

8.2.15.4.1 See the operation binding for "GetMaxSessions" in E132-1-V0305-SecurityAdmin-Binding.wsdl. Key information is shown for convenience in Table 54.

Table 54 GetMaxSessions Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.secAdmin-binding:GetMaxSessions
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.3 *SessionManager Interface*

8.3.1 *Session Identifier*

8.3.1.1 The SessionManager interface uses the same SOAP header as described in ¶8.2.7 for the SecurityAdmin interface. With the exception of the EstablishSession operation, all SOAP request and response messages defined in the SessionManager interface includes this SOAP header to contain the session identifier. The EstablishSession operation is where the session identifier is first defined by the equipment and communicated to client in the EstablishSession response.

8.3.2 *XML Schema and WSDL Files*

8.3.2.1 The XML Schema and WSDL defined by this specification for the SessionManager interface is contained in the following documents:

Table 55 XML Schema

File Name	E132-1-V0305-Schema.xsd
Target Namespace	urn:semi-org:xsd.E132-1.V0305.auth
Imported/Referenced Namespaces	http://www.w3.org/2001/XMLSchema urn:semi-org:xsd.CommonComponents.V0305.ccs
Description	This file defines all of the SEMI E132 data types and elements used by the SecurityAdmin, SessionManager and SessionClient interfaces.

Table 56 SessionManager PortType Definitions

File Name	E132-1-V0305-SessionManager-PortType.wsdl
Target Namespace	urn:semi-org:ws.E132-1.V0305.sessMgr-portType
Imported/Referenced Namespaces	urn:semi-org:xsd.E132-1.V0305.auth http://schemas.xmlsoap.org/wsdl/ http://www.w3.org/2001/XMLSchema
Description	This file defines all of the input/output messages and operations for the SessionManager interface, based on the data types defined in the SEMI E132 XML Schema.

Table 57 SessionManager Binding Definitions

File Name	E132-1-V0305-SessionManager-Binding.wsdl
Target Namespace	urn:semi-org:ws.E132-1.V0305.sessMgr-binding
Imported/Referenced Namespaces	urn:semi-org:ws.E132-1.V0305.sessMgr-portType http://schemas.xmlsoap.org/wsdl/soap/ http://schemas.xmlsoap.org/wsdl/
Description	This file binds the abstract portType definition to HTTP and SOAP, specifying required SOAP and HTTP headers for each operation and the XML encoding style for the SessionManager interface.

8.3.2.2 These documents are a part of this specification and should accompany this document. The contents of these documents constitute the core part of this specification.

8.3.3 *WSDL Port Type Overview*

8.3.3.1 The SessionManager WSDL portType definition organizes the SEMI E132.1 XML Schema data types into a collection of named operations with inputs and outputs that correspond to the UML operations that are defined for the SessionManager interface in SEMI E132. The WSDL portType itself is named after the SEMI E132 interface, and each WSDL portType operation is named after the corresponding UML operation defined for the interface. Each WSDL operation consists of two WSDL message definitions corresponding to the input and output (request and response) for the UML operation defined in SEMI E132. The relationship between the UML interface and the WSDL portType definition is similar in structure to that of the SecurityAdmin interface. Table 58 describes the mapping between UML operations and WSDL operations of the SessionManager interface. The WSDL message and operation definitions are described in ¶¶8.3.6 through 8.3.9.

Table 58 SessionManager Port Type

SEMI E132 Class Name	SessionManager
WSDL Port Type Name	SessionManager
SEMI E132 Operation → WSDL Operation	EstablishSession → EstablishSession CloseSession → CloseSession SessionPing → SessionPing PersistSession → PersistSession

8.3.4 *WSDL Binding Overview*

8.3.4.1 The WSDL SessionManager binding definition specifies a message and transport protocol to use for a given portType (SOAP and HTTP for SEMI E132.1), the interface style used (document style for SEMI E132.1). These settings are shown in Table 59. For each WSDL portType operation, the binding defines any SOAP headers used and whether or not they are required, the HTTP SOAPAction header value to use for that operation, and the XML encoding to use (literal for SEMI E132.1). The binding definitions for each portType operation are described in ¶¶8.3.6 through 8.3.9.

Table 59 SessionManager Binding

SEMI E132 Class Name	SessionManager
WSDL Binding Name	SessionManagerBinding
SOAP Binding Style	document
SOAP Transport	http://schemas.xmlsoap.org/soap/http

8.3.5 *WSDL Service Overview*

8.3.5.1 This specification does not provide a WSDL service definition. WSDL service definitions provide one way to define and locate the endpoint(s) at which a given interface can be accessed by a client. Such information cannot be determined until the equipment has been installed on a factory network, and therefore is out of scope for this specification. Users are responsible for defining the mechanisms by which clients locate E132 equipment services in the factory. A sample SessionManager service definition is provided in Figure 32 for reference only.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SessionManager"
       xmlns="urn:semi-org:ws.E132-1.V0305.sessMgr-session"
       targetNamespace="urn:semi-org:ws.E132-1.V0305.sessMgr-session"
       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
       xmlns:SMBind="urn:semi-org:ws.E132-1.V0305.sessMgr-binding">

    <wsdl:import namespace="urn:semi-org:ws.E132-1.V0305.sessMgr-binding"
location="E132-1-V0305-SessionManager-Binding.wsdl"/>

       <wsdl:service name="SessionManager">
              <wsdl:port name="SessionManagerPort"
binding="SMBind:SessionManagerBinding">
                     <soap:address location="https://www.someplace/SessionManager"/>
              </wsdl:port>
       </wsdl:service>

</wsdl:definitions>
```

Figure 32
Sample SessionManager WSDL Service Definition

8.3.6 *EstablishSession Operation*

8.3.6.1 *XML Schema Types*

8.3.6.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the EstablishSession operation. See GetDefinedPrivileges operation regarding schema conventions used. The session identifier SOAP header is not used for this operation.

8.3.6.1.2 *EstablishSession Request*

8.3.6.1.2.1 The service request is mapped to the XML Schema element EstablishSessionRequest of type EstablishSessionRequestType. Table 60 shows how the input parameters are mapped to XML. EndPointSelectorType is as described in ¶8.2.13.1.5, XML description for the EndPoint class.

Table 60 Translation Table for EstablishSession Input Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
endpoint	Structured type	Element	EndPoint: EndPointSelectorType

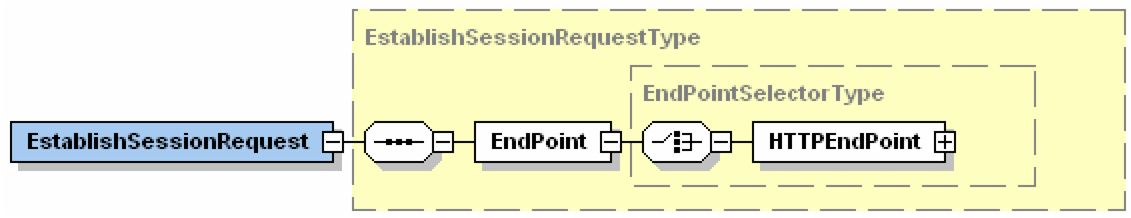

Figure 33
EstablishSessionRequest XML Schema Element

8.3.6.1.3 *EstablishSession Response*

8.3.6.1.3.1 The service response is mapped to the XML Schema element EstablishSessionResponse of type EstablishSessionResponseType. Table 61 shows how the output parameters are mapped to XML. The XML schema diagram for EstablishSessionResponse is shown in Figure 34. Errors for this operation are returned using the XML Schema element Error of type EstablishSessionErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 61 Translation Table for EstablishSession Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
sessionId	Text	Element	SessionID: xsd:string
error	Structured data	Element	Error: EstablishSessionErrorType

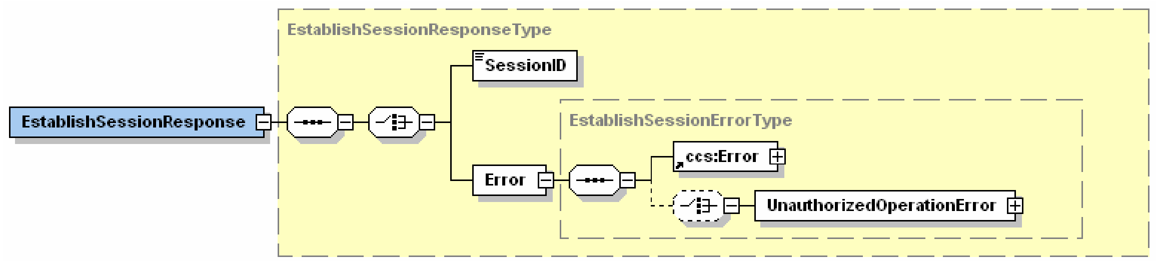
EstablishSessionResponseType
SessionID
EstablishSessionResponse
EstablishSessionErrorType
ccs:Error
Error
UnauthorizedOperationError

Figure 34
EstablishSessionResponse XML Schema Element

8.3.6.2 *WSDL Message(s)*

8.3.6.2.1 Three messages are defined for this operation corresponding to the Request and Response messages. The SOAP header message for session identifier is not used for this operation. See the message definitions for "EstablishSessionRequestMessage", and "EstablishSessionResponseMessage" in E132-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 62.

Table 62 EstablishSession Messages

Message Name → Schema Element Name	EstablishSessionRequestMessage → auth:EstablishSessionRequest EstablishSessionResponseMessage → auth:EstablishSessionResponse

8.3.6.3 *WSDL Operation*

8.3.6.3.1 See the portType operation definition for "EstablishSession" in E132-1-V0305-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 63.

Table 63 EstablishSession PortType Operation

Input Message Name	EstablishSessionRequestMessage
Output Message Name	EstablishSessionResponseMessage

8.3.6.4 *WSDL Operation Binding*

8.3.6.4.1 See the operation binding for "EstablishSession" in E132-1-V0305-SessionManager-Binding.wsdl. Key information is shown for convenience in Table 64. Note that the E132Header is still required, though the equipment shall ignore the session ID provided in the request as this will be generated by the equipment following session establishment. If SSL is enabled, equipment shall always use the ID from the client's certificate for authorization verifications, otherwise if SSL is disabled, the client can be identified using the FROM field of the E132Header. See ¶7.3 for more information on and impact of disabling SSL.

Table 64 EstablishSession Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.SessionManagerBinding:EstablishSession
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.3.7 *PersistSession Operation*

8.3.7.1 *XML Schema Types*

8.3.7.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the PersistSession operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.3.7.1.2 *PersistSession Request*

8.3.7.1.2.1 The service request is mapped to the XML Schema element PersistSessionRequest of type PersistSessionRequestType. Table 65 shows how the input parameters are mapped to XML.

Table 65 Translation Table for PersistSession Input Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
persist	Boolean	Element	Persist: xsd:boolean

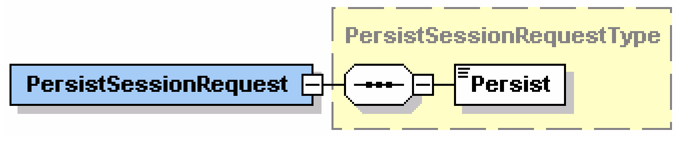

Figure 35
PersistSessionRequest XML Schema Element

8.3.7.1.3 *PersistSession Response*

8.3.7.1.3.1 The service response is mapped to the XML Schema element PersistSessionResponse of type PersistSessionResponseType. Table 66 shows how the output parameters are mapped to XML, the error type is just the common error described in ¶8.2.6.1. The XML Schema diagram for PersistSessionResponse is shown in Figure 36.

Table 66 Translation Table for PersistSession Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
error	Structured data, of type Error, defined in SEMI E138	Element	Error: ErrorType

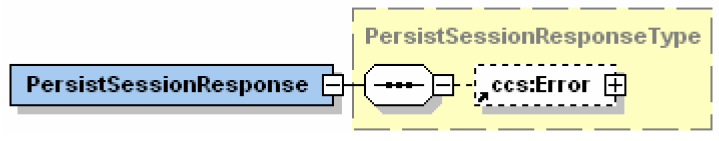

Figure 36
PersistSessionResponse XML Schema Element

8.3.7.2 *WSDL Message(s)*

8.3.7.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “PersistSessionRequestMessage”, and “PersistSessionResponseMessage” in E132-1-V0305-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 67.

Table 67 PersistSession Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header PersistSessionRequestMessage → auth:PersistSessionRequest PersistSessionResponseMessage → auth:PersistSessionResponse

8.3.7.3 *WSDL Operation*

8.3.7.3.1 See the portType operation definition for “PersistSession” in E132-1-V0305-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 68.

Table 68 PersistSession PortType Operation

Input Message Name	PersistSessionRequestMessage
Output Message Name	PersistSessionResponseMessage

8.3.7.4 *WSDL Operation Binding*

8.3.7.4.1 See the operation binding for “PersistSession” in E132-1-V0305-SessionManager-Binding.wsdl. Key information is shown for convenience in Table 69.

Table 69 PersistSession Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.SessionManagerBinding:PersistSession
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.3.8 *SessionPing Operation*

8.3.8.1 *XML Schema Types*

8.3.8.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the SessionPing operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.3.8.1.2 *SessionPing Request*

8.3.8.1.2.1 The service request is mapped to the XML Schema element SessionPingRequest of type SessionPingRequestType. There are no input parameters to this operation, so the schema element has no content.

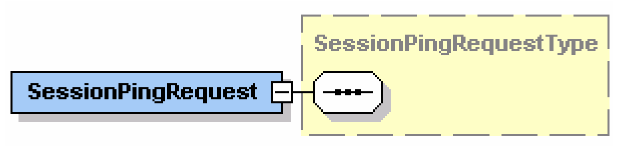

Figure 37
SessionPingRequest XML Schema Element

8.3.8.1.3 *SessionPing Response*

8.3.8.1.3.1 The service response is mapped to the XML Schema element SessionPingResponse of type SessionPingResponseType. Table 70 shows how the output parameters are mapped to XML, the error type is just the common error described in ¶8.2.6.1. The XML Schema diagram for SessionPingResponse is shown in Figure 38.

Table 70 Translation Table for SessionPing Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
equipmentId	Text	Element	ID: xsd:string
error	Structured data, of type Error, defined in SEMI E138	Element	Error: ErrorType

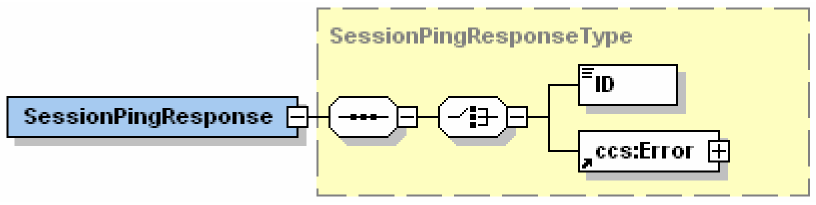

Figure 38
SessionPingResponse XML Schema Element

8.3.8.2 *WSDL Message(s)*

8.3.8.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “SessionPingRequestMessage”, and “SessionPingResponseMessage” in E132-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 71.

Table 71 SessionPing Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header SessionPingRequestMessage → auth:SessionPingRequest SessionPingResponseMessage → auth:SessionPingResponse

8.3.8.3 *WSDL Operation*

8.3.8.3.1 See the portType operation definition for “SessionPing” in E132-1-V0305-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 72.

Table 72 SessionPing PortType Operation

Input Message Name	SessionPingRequestMessage
Output Message Name	SessionPingResponseMessage

8.3.8.4 *WSDL Operation Binding*

8.3.8.4.1 See the operation binding for “SessionPing” in E132-1-V0305-SessionManager-Binding.wsdl. Key information is shown for convenience in Table 73.

Table 73 SessionPing Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.SessionManagerBinding:SessionPing
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.3.9 *CloseSession Operation*

8.3.9.1 *XML Schema Types*

8.3.9.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the CloseSession operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.3.9.1.2 *CloseSession Request*

8.3.9.1.2.1 The service request is mapped to the XML Schema element CloseSessionRequest of type CloseSessionRequestType. Table 74 shows how the input parameters are mapped to XML.

Table 74 Translation Table for CloseSession Input Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
sessionId	Text	Element	SessionID: xsd:string

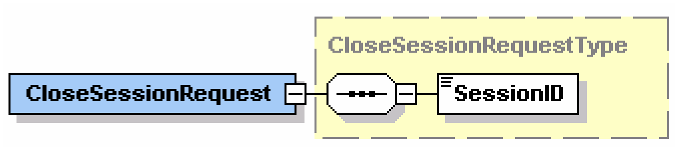

Figure 39
CloseSessionRequest XML Schema Type

8.3.9.1.3 *CloseSession Response*

8.3.9.1.3.1 The service response is mapped to the XML Schema element CloseSessionResponse of type CloseSessionResponseType. Table 75 shows how the output parameters are mapped to XML. The XML Schema diagram for CloseSessionResponse is shown in Figure 40. Errors for this operation are returned using the XML Schema element Error of type CloseSessionErrorType. The error type is the same as that defined for the GetDefinedPrivileges operation described in ¶8.2.9.1.5.

Table 75 Translation Table for CloseSession Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
error	Structured data	Element	Error: CloseSessionErrorType

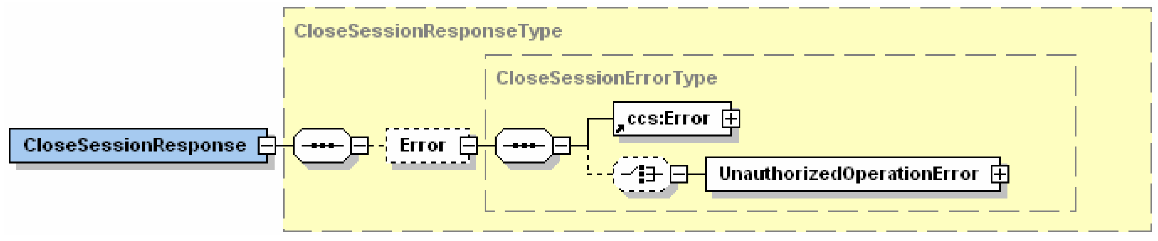

Figure 40
CloseSessionResponse XML Schema Element

8.3.9.2 *WSDL Message(s)*

8.3.9.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HeaderMessage”, “CloseSessionRequestMessage”, and “CloseSessionResponseMessage” in E132-1-V0305-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 76.

Table 76 CloseSession Messages

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header CloseSessionRequestMessage → auth:CloseSessionRequest CloseSessionResponseMessage → auth:CloseSessionResponse

8.3.9.3 *WSDL Operation*

8.3.9.3.1 See the portType operation definition for "CloseSession" in E132-1-V0305-SessionManager-PortType.wsdl. Key information is shown for convenience in Table 77.

Table 77 CloseSession PortType Operation

Input Message Name	CloseSessionRequestMessage
Output Message Name	CloseSessionResponseMessage

8.3.9.4 *WSDL Operation Binding*

8.3.9.4.1 See the operation binding for "CloseSession" in E132-1-V0305-SessionManager-Binding.wsdl. Key information is shown for convenience in Table 78.

Table 78 CloseSession Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.SessionManagerBinding:CloseSession
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

8.4 *SessionClient Interface*

8.4.1 *Session Identifier*

8.4.1.1 Equipment initiated SOAP messages such as equipment notifications are not required by SEMI E132 to be send over an SSL channel, though equipment is still required to include the session identifier in all messages. Since sending the session ID in plaintext is undesired, this specification defines a hashed session identifier to be included in the SOAP header. Figure 41 shows the XML Schema diagram for the E132HashHeader schema element. The hashed session identifier is modeled as the XML element SessionIDHash of type xsd:base64Binary in the E132HashHeader.

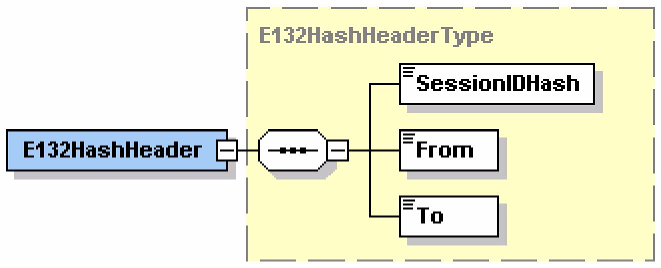

Figure 41
E132HashHeader Schema Element

8.4.1.2 The value of the SessionIDHash element is computed by first applying the SHA-1 hash function to the session identifier, the resulting hashed output is then based 64 encoded. The session identifier must be converted to a UTF-8 string prior to applying the hash. See SHA documentation reference in ¶4.4 for algorithm specification. Base64Encoding is as specified in the XML Schema standard (part 2) referenced in ¶4.3.

8.4.1.3 All equipment initiated SOAP messages and corresponding responses over SEMI E132 authenticated sessions shall include the E132HashHeader in the SOAP header.

8.4.2 The From/To fields are as described in ¶8.2.8.

8.4.3 *XML Schema and WSDL Files*

8.4.3.1 The XML Schema and WSDL defined by this specification for the SessionClient interface is contained in the following documents:

Table 79 XML Schema

File Name	E132-1-V0305-Schema.xsd

Table 79 XML Schema

File Name	E132-1-V0305-Schema.xsd
Target Namespace	urn:semi-org:xsd.E132-1.V0305.auth
Imported/Referenced Namespaces	http://www.w3.org/2001/XMLSchema urn:semi-org:xsd.CommonComponents.V0305.ccs
Description	This file defines all of the SEMI E132 data types and elements used by the SecurityAdmin, SessionManager and SessionClient interfaces.

Table 80 SessionClient PortType Definitions

File Name	E132-1-V0305-SessionClient-PortType.wsdl
Target Namespace	urn:semi-org:ws.E132-1.V0305.sessClient-portType
Imported/Referenced Namespaces	urn:semi-org:xsd.E132-1.V0305.auth http://schemas.xmlsoap.org/wsdl/ http://www.w3.org/2001/XMLSchema
Description	This file defines all of the input/output messages and operations for the SessionClient interface, based on the data types defined in the SEMI E132 XML Schema.

Table 81 SessionClient Binding Definitions

File Name	E132-1-V0305-SessionClient-Binding.wsdl
Target Namespace	urn:semi-org:ws.E132-1.V0305.sessClient-binding
Imported/Referenced Namespaces	urn:semi-org:ws.E132-1.V0305.sessClient-portType http://schemas.xmlsoap.org/wsdl/soap/ http://schemas.xmlsoap.org/wsdl/
Description	This file binds the abstract portType definition to HTTP and SOAP, specifying required SOAP and HTTP headers for each operation and the XML encoding style for the SessionClient interface.

8.4.3.2 These documents are a part of this specification and should accompany this document. The contents of these documents constitute the core part of this specification.

8.4.4 *WSDL Port Type Overview*

8.4.4.1 The SessionClient WSDL portType definition organizes the SEMI E132.1 XML Schema data types into a collection of named operations with inputs and outputs that correspond to the UML operations that are defined for the SessionClient interface in SEMI E132. The WSDL portType itself is named after the SEMI E132 interface, and each WSDL portType operation is named after the corresponding UML operation defined for the interface. Each WSDL operation consists of two WSDL message definitions corresponding to the input and output (request and response) for the UML operation defined in SEMI E132. Table 82 describes the mapping between the UML operations and WSDL operations. The WSDL message and operation definitions are described in more detail in ¶¶8.4.7 through 8.4.9 .

Table 82 SessionClient Port Type

SEMI E132 Class Name	SessionClient
WSDL Port Type Name	SessionClient
SEMI E132 Operation → WSDL Operation	SessionPing → SessionPing SessionFrozen → SessionFrozen SessionClosed → SessionClosed

8.4.5 *WSDL Binding Overview*

8.4.5.1 The WSDL SessionClient binding definition specifies a message and transport protocol to use for a given portType (SOAP and HTTP for SEMI E132.1), the interface style used (document style for SEMI E132.1). These settings are shown in Table 83. For each WSDL portType operation, the binding defines any SOAP headers used and whether or not they are required, the HTTP SOAPAction header value to use for that operation, and the XML encoding to use (literal for SEMI E132.1). The binding definitions for each portType operation are described in ¶¶8.4.7 through 8.4.9.

Table 83 SessionClient Binding

SEMI E132 Class Name	SessionClient
WSDL Binding Name	SessionClientBinding
SOAP Binding Style	document
SOAP Transport	http://schemas.xmlsoap.org/soap/http

8.4.6 *WSDL Service Overview*

8.4.6.1 This specification does not provide a WSDL service definition. WSDL service definitions provide one way to define and locate the endpoint(s) at which a given interface can be accessed by a client. Such information cannot be determined until the equipment has been installed on a factory network, and therefore is out of scope for this specification. A sample SessionClient service definition is provided in Figure 42 for reference only.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="SessionClient"
       xmlns="urn:semi-org:ws.E132-1.V0305.sessClient-service"
       targetNamespace="urn:semi-org:ws.E132-1.V0305.sessClient-service"
       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
       xmlns:SCBind="urn:semi-org:ws.E132-1.V0305.sessClient-binding">

    <wsdl:import namespace="urn:semi-org:ws.E132-1.V0305.sessClient-binding"
location="E132-1-V0305-SessionClient-Binding.wsdl"/>

       <wsdl:service name="SessionClient">
              <wsdl:port name="SessionClientPort"
binding="SCBind:SessionClientBinding">
                     <soap:address location="http://www.someplace/SessionClient"/>
              </wsdl:port>
       </wsdl:service>

</wsdl:definitions>
```

Figure 42
Sample SessionClient WSDL Service definition

8.4.7 *SessionPing Operation*

8.4.7.1 *XML Schema Types*

8.4.7.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the SessionPing operation.

8.4.7.1.2 *SessionPing Request*

8.4.7.1.2.1 The service request is mapped to the XML Schema element SessionPingRequest of type SessionPingRequestType. There are no input parameters to this operation, so the schema element has no content. See SessionPing operation of the SessionManager interface.

8.4.7.1.3 *SessionPing Response*

8.4.7.1.3.1 The service response is mapped to the XML Schema element SessionPingResponse of type SessionPingResponseType. Table 84 shows how the output parameters are mapped to XML. See SessionPing operation of the SessionManager interface for more details.

Table 84 Translation Table for SessionPing Output Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
clientId	Text	Element	ID: xsd:string
error	Structured data, of type Error, defined in the SEMI Specification for Semiconductor Common Components	Element	Error: ErrorType

8.4.7.2 *WSDL Message(s)*

8.4.7.2.1 Three messages are defined for this operation corresponding to the SEMI E132 Header, Request and Response messages. See the message definitions for “E132HashHeaderMessage”, “SessionPingRequestMessage”, and “SessionPingResponseMessage” in E132-1-V0305-SessionClient-PortType.wsdl. Key information is shown for convenience in Table 85.

Table 85 SessionPing Messages

Message Name → Schema Element Name	E132HashHeaderMessage →auth:E132HashHeader SessionPingRequestMessage → auth:SessionPingRequest SessionPingResponseMessage → auth:SessionPingResponse

8.4.7.3 *WSDL Operation*

8.4.7.3.1 See the portType operation definition for “SessionPing” in E132-1-V0305-SessionClient-PortType.wsdl. Key information is shown for convenience in Table 86.

Table 86 SessionPing PortType Operation

Input Message Name	SessionPingRequestMessage
Output Message Name	SessionPingResponseMessage

8.4.7.4 *WSDL Operation Binding*

8.4.7.4.1 See the operation binding for “SessionPing” in E132-1-V0305-SessionClient-Binding.wsdl. Key information is shown for convenience in Table 87.

Table 87 SessionPing Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.sessClient-binding:SessionPing
Input Headers (WSDL Message, Required)	E132HashHeaderMessage, required
Output Headers (WSDL Message, Required)	E132HashHeaderMessage, required

8.4.8 *SessionFrozen Operation*

8.4.8.1 *XML Schema Types*

8.4.8.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the SessionFrozen operation. See GetDefinedPrivileges operation regarding schema conventions used.

8.4.8.1.2 *SessionFrozen Notification*

8.4.8.1.2.1 The SessionFrozen notification is mapped to the XML Schema element SessionFrozenNotification of type SessionFrozenNotificationType. There are no input parameters for this notification, so the schema element has no content.

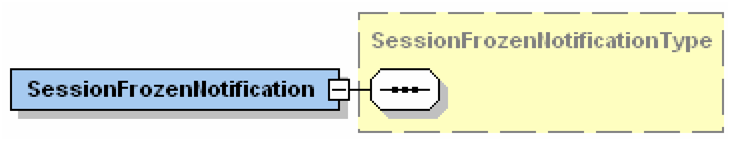

Figure 43
SessionFrozenNotification XML Schema Element

8.4.8.2 *WSDL Message(s)*

8.4.8.2.1 Two messages are defined for this operation corresponding to the SEMI E132 Hash Header, and the Notification message. See message definitions for "E132HashHeaderMessage" and "SessionFrozenInMessage" in E132-1-V0305-SessionClient-PortType.wsdl. Key information is shown for convenience in Table 88.

Table 88 SessionFrozen Messages

Message Name → Schema Element Name	E132HashHeaderMessage → auth:E132HashHeader SessionFrozenInMessage → auth:SessionFrozenNotification

8.4.8.3 WSDL Operation

8.4.8.3.1 See the portType operation definition for "SessionFrozen" in E132-1-V0305-SessionClient-PortType.wsdl. Key information is shown for convenience in Table 89. There is no output message defined for notifications.

Table 89 SessionFrozen PortType Operation

Input Message Name	SessionFrozenInMessage

8.4.8.4 *WSDL Operation Binding*

8.4.8.4.1 See the operation binding for "SessionFrozen" in E132-1-V0305-SessionClient-Binding.wsdl. Key information is shown for convenience in Table 90.

Table 90 SessionFrozen Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.sessClient-binding:SessionFrozen
Input Headers (WSDL Message, Required)	E132HashHeaderMessage, required

8.4.9 *SessionClosed Operation*

8.4.9.1 *XML Schema Types*

8.4.9.1.1 This section describes the XML mappings for the classes defined in SEMI E132 for the SessionClosed operation. See GetDefinedPrivileges operation regarding schema conventions used. This operation is just a notification only so there is no response defined.

8.4.9.1.2 *SessionClosed Notification*

8.4.9.1.2.1 The SessionClosed notification is mapped to the XML Schema element SessionClosedNotification of type SessionClosedNotificationType. Table 91 shows how the input parameters are mapped to XML.

Table 91 Translation Table for SessionClosed Input Parameters

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
sessionId	Text	Element	SessionID: xsd:string

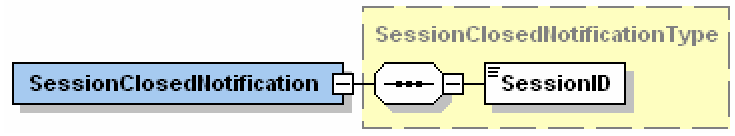

Figure 44
SessionClosedNotification XML Schema Element

8.4.9.2 *WSDL Message(s)*

8.4.9.2.1 Two messages are defined for this operation corresponding to the SEMI E132 Hash Header, and the Notification message. See the message definitions for "E132HashHeaderMessage", and "SessionClosedInMessage" in E132-1-V0305-SessionClient-PortType.wsdl. Key information is shown for convenience in Table 92.

Table 92 SessionClosed Messages

Message Name → Schema Element Name	E132HashHeaderMessage → auth:E132HashHeader SessionClosedInMessage → auth:SessionClosedNotification

8.4.9.3 *WSDL Operation*

8.4.9.3.1 See the portType operation definition for "SessionClosed" in E132-1-V0305-SessionClient-PortType.wsdl. Key information is shown for convenience in Table 93. There is no output message defined for notifications.

Table 93 SessionClosed PortType Operation

Input Message Name	SessionClosedInMessage

8.4.9.4 *WSDL Operation Binding*

8.4.9.4.1 See the operation binding for "SessionClosed" in E132-1-V0305-SessionClient-Binding.wsdl. Key information is shown for convenience in Table 94.

Table 94 SessionClosed Operation Binding

SOAPAction	urn:semi-org:ws.E132-1.V0305.sessClient-binding:SessionClosed
Input Headers (WSDL Message, Required)	E132HashHeaderMessage, required

SEMI E133-0705
PROVISIONAL SPECIFICATION FOR AUTOMATED PROCESS CONTROL SYSTEMS INTERFACE

This provisional specification was technically approved by the global Information & Control Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on April 7, 2005. It was available at www.semi.org in June 2005 and on CD-ROM in July 2005. Originally published March 2004; previously published November 2004.

1 Purpose

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

2 Scope

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

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

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

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

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

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

2.5 Note that conformance with this standard as defined in §11 is not enforceable until the complete data structures have been defined for at least one functional group.

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

3 Limitations

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

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

4 Referenced Standards and Documents

4.1 *SEMI Standards*

SEMI E81 — Provisional Specification for CIM Framework Domain Architecture

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

SEMI E125 — Provisional Specification for Equipment Self Description (ESD)

SEMI E134 — Specification for Data Collection Management

4.2 *Other Sources*

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

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

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

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

IEC 61158-5, "Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 5: Application layer service definition," Final Draft International Standard, 2003.

ISO 8601 — Data elements and interchange formats—Information interchange—Representation of dates and times, International Organization for Standardization, (December 2000).

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

5 Terminology

5.1 *Abbreviations & Acronyms*

5.1.1 *XML* — eXtensible Markup Language

5.2 *Definitions*

5.2.1 *Advanced Process Control (APC)*[1] — the manufacturing discipline for applying control strategies and/or employing analysis and computation mechanisms to recommend optimized machine settings and detect faults and determine their cause.

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

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

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

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

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

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

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

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

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

5.2.11 *Process Control System (PCS)*[1] — a system capable of performing Process Control, which includes one or more of R2R Control, FD, FC, FP, SPC, or any future Process Control functionality defined in this standard for a PCS functional group.

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

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

6 Background

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

7 Conventions

7.1 This section defines conventions followed in this document.

7.2 *Object Modeling*

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

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

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

7.2.3 *Class Diagrams* — UML Class Charts.

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

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

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

Table 1 Attribute Table Format

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

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

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

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

7.2.6.4 *Constraints*

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

7.2.7 *UML Associations*

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

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

Table 2 Association Table Format

Association Role Name	*Role Definition*	*Comments*

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

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

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

7.2.9 *Method Definition* — The methods (in other words, the messages that comprise the external interface to an object) supported by a PCS functional group are defined in table format as illustrated below.

Table 3 Method Definition Table Format

#	*Method*	*Arguments*	*Returns*	*Requirement*	*Behavior/Comment*
				Y or N	

7.2.9.1 *Method* — Method name.

7.2.9.2 *Arguments* — Input parameters.

7.2.9.3 *Returns* — Output parameters.

7.2.9.4 *Requirement* — Is this method required of a compliant implementation? Y – Yes, or N – No.

7.2.9.5 *Behavior/Comment* — Summary description of the method's purpose and behavior, explanation of expected system response to this method, etc.

7.3 *Data Types*

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

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

7.4 *Sequence Behavior*

7.4.1 Sequence behavior description is specified in the form of UML Sequence Diagrams.

7.5 *State Behavior*

7.5.1 State behavior description is specified in the form of UML State Charts.

8 Overview

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

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

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

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

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

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

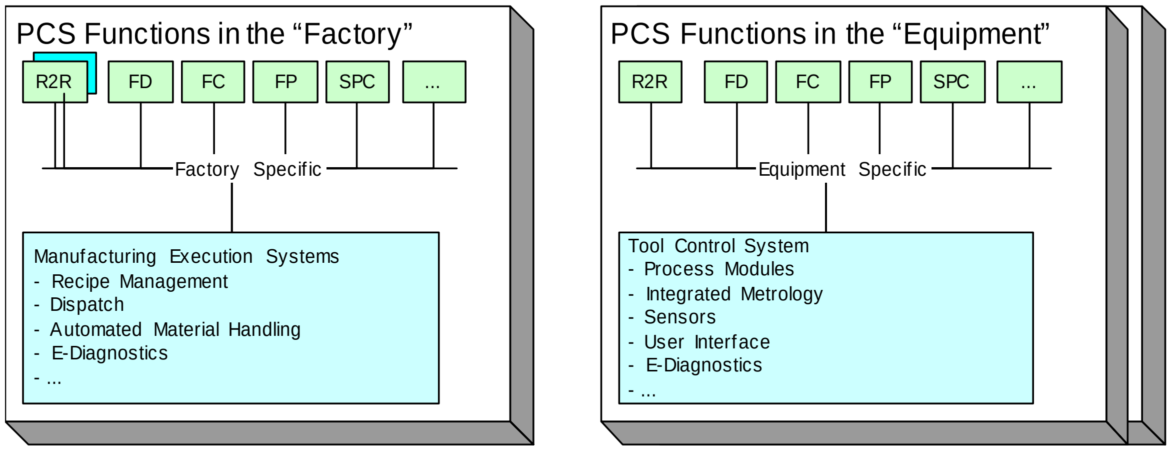

Figure 1
PCS Functions

8.4.1 Process Control Systems may consist of a variety of capabilities or functions at both the factory and equipment level as shown in Figure 1. Moreover, this may include multiple instances of the same function type from different suppliers. In the example above, several run-to-run controllers are running at the factory level with differences that are only visible as internal special capabilities. This interface standard will facilitate easier collaboration between the tool level functions and factory level systems independent of the supplier that developed the internal function's capabilities.

9 Process Control Systems Functional Group Capabilities and Interface Data Structures

9.1 *Functional Group Partitioning*

9.1.1 Process control systems are partitioned into functional groups using the criteria defined in §8. This standard defines the interface data structures, services and behavior associated with these functional groups. In this section the capabilities of the PCS functional groups are defined in more detail. First of all, the basic capabilities common to all functional groups are presented. Then the additional basic capabilities unique to each functional group are presented. A high level description of the specifications of data structures, services and behavior are described in §10.

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

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

9.2 *PCS Analysis Engine*

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

9.2.2 *General Capability Description*

9.2.2.1 All PCS Analysis Engines shall have inputs and outputs as shown in Figure 2. This means that the PCS Analysis Engines shall support the consumption of the specified inputs and production of the specified outputs as necessary to achieve the required capabilities of that analysis engine as specified in this standard. These inputs and outputs are specified as aggregate data structures. The structures are aggregated into "collection types" according to the category of data. A definition of these collection types is provided in Table 4. The PCS Analysis Engine shall have the general behavior of utilizing specified inputs, performing tasks based to some degree on these inputs, and generating outputs. Any additional structure and behavior of various PCS Analysis Engines are outlined in the remainder of this section and specified in detail in ¶10.4.

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

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

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

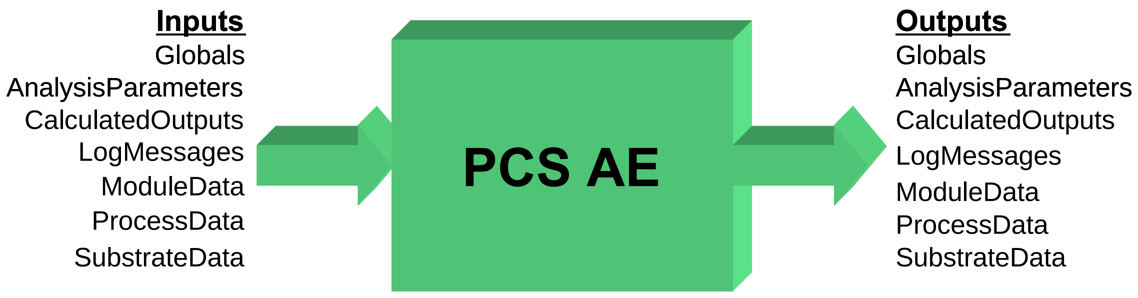

Figure 2
PCS Analysis Engine Common Input/Output Structure

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

Table 4 PCS Analysis Engine Inputs and Outputs Collection Type Definitions

Collection Type	*Definition*	*Comments*
Globals	Information that applies globally across an entire job.	A lot id, for example could be specified once as a global so that it would not have to be specified with many other data sets.
AnalysisParameters	Information that identifies numeric data used by the Analysis Engine to perform various calculations.	Equation coefficients, feedforward data, etc.
CalculatedOutputs	Data representing the results of analysis from an Analysis Engine.	A calculation derived from various inputs to the Analysis Engine indicating wafer state, process state or model state information, recommended machine settings, etc.
LogMessages	Data describing the activity of an Analysis Engine as it executes its tasks.	Data describing the activities of the Analysis Engine. Note that this does not include log information on the communication protocol with the AE. An example that is acceptable is “Algorithm #3 with model parameters ‘A’, ‘B’, and ‘C’”.
ModuleData	Information representing the current or last known physical state of a process module that will be used by an Analysis Engine to process material.	Machine, or module E10 state, RF hours, number of wafers processed, etc.
ProcessData	Information representing the process itself, such as the name and type of the process and the current or last known process state based on measurements from sensors of the processing environment.	Name of the process (CMP), in-situ process measurements, trace data, and summary statistics etc.
SubstrateData	Information about the substrate being associated with the Analysis Engine. This could include data representing a single wafer, multiple wafers, or one or more lots, as well as attributes of the material.	Profile or topology data, and other characteristics measured physically or electrically.

Table 5 PCS Analysis Engine Functional Group Capabilities

#	*Description*	*Req'd*	*Comments*
3.1	**Provide a PCS Analysis Engine capability** PCS Analysis Engine shall be capable of performing one or more of R2R control, FD, FC, FP, SPC, or any future functionality defined in this standard as a PCS functional group.	Y	
3.2	**Provide the required PCS Analysis Engine methods** The PCS analysis engine shall provide the required interface methods as defined in this standard. See ¶10.4.1.	Y	These include Execute and Update.
3.3	**Provide the required PCS Analysis Engine method capabilities** The PCS analysis engine shall provide the required interface method capabilities as defined in this standard. See ¶10.4.1.	Y	These include Log Data, Exception Handling, Get Topic List, Subscribe to Topic, Get Topic Data.
3.4	**Maintain models of selected process control system environment** The PCS Analysis Engine shall maintain explicit or implicit models of the selected process control environment. These models shall be utilized to provide the analysis or control capability of the Analysis Engine.	Y	The specific form and behavior of these models is beyond the scope of this standard.
3.5	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action** The PCS Analysis Engine shall support an input mechanism whereby the analysis engine can be instructed to take a specific unique action.	Y	The mechanism could be a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model and generation of a PCS result, update of a model, or communication of model parameters.
3.6	**Interface with multiple external systems** The PCS Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.
3.7	**Edit configuration of models** The PCS Analysis Engine shall provide the capability to create/add, edit configuration, and delete models.	Y	This capability will be supported through the Analysis Engine interface.
3.8	**Interface with other Analysis Engines** The PCS Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	May be a requirement of Analysis Engines belonging to a specific functional group.
3.9	**Advanced Model Management** The PCS Analysis Engine may provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	
3.10	**Provide the optional PCS Analysis Engine method capabilities** The PCS Analysis Engine shall provide all or a subset of the optional interface method capabilities as defined in this standard. See ¶10.4.1.	N	

9.2.3 *Common Interface Data Structures*

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

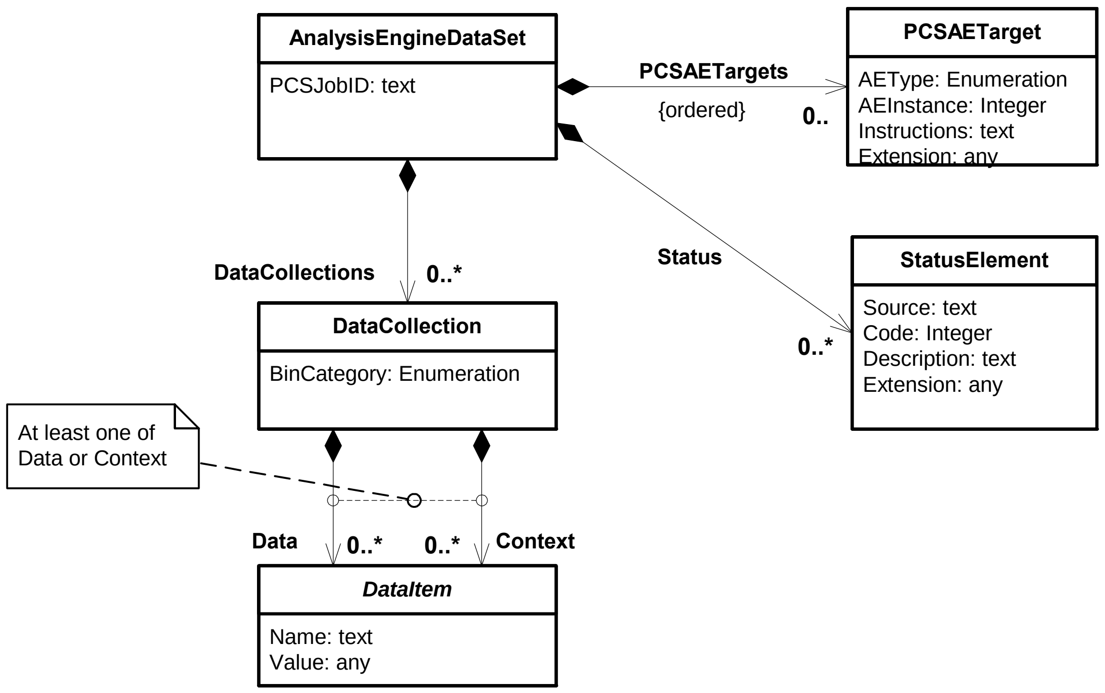

Figure 3
Analysis Engine Data Structure Model for a Complete Message Associated with a PCS Job

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

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

Table 6 Attributes of AnalysisEngineDataSet

Attribute Name	*Description*	*Access*	*Reqd*	*Form*
PCSJobID	A unique identifier of a Process Control System Job.	NA	C	ObjID—reference SEMI E39
PCSAETargets	An attribute that indicates a list of AE types targeted by the AnalysisEngineDataSet	NA	Y	An ordered list of structured data of type PCSAETarget; see below.
Status	An attribute formatted as an extensible data structure that relates properties of the success or failure of a PCS AE, identified in PCSAETargets, to execute a PCSJob	NA	C	An unordered list of structured data of type StatusElement; see below.

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

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

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

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

9.2.3.2.1.2.1.1 This attribute has the following form:

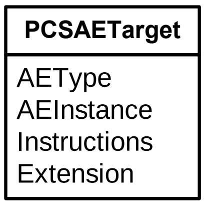

Figure 4
PCSAETarget Data Structure

Table 7 PCSAETarget Attribute Definition

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

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

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

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

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

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

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

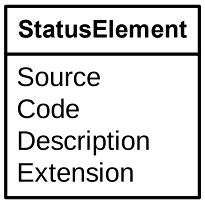

Figure 5
StatusElement Data Structure

Table 8 StatusElement Attribute Definition

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

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

- The message is an output from an AE.

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

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

Table 9 Attributes of DataCollection

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

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

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

Table 10 Attributes of DataItem

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

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

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

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

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

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

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

9.2.3.3.2 The first attribute of any DataCollection shall be BinCategory.

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

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

```
•   PCSJobID == “[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}”
•   PCSAETargets
        APCSAETarget: PCSAETarget
                AEType == “R2R”
                AEInstance == 1
                Instructions
                        Instruction ==  “R2RCalculateControlAdvice”
                Extension == “execute serially”
•   Status
        AStatusElement: StatusElement
                Source == “urn:semi-org:E132”
                Code == 0
•   ADataCollection: DataCollection
            BinCategory == “Globals”
                    Role == “Context”
                            ADataItem: DataItem
                                    AName: (type == text)
                                    AValue: (type == any)
                            ADataItem
                                    AName: (type == text)
                                    AValue: (type == any)
            BinCategory == “ModuleData”
                    Role == “Context”
                            ADataItem: DataItem
                                    AName: (type == text)
                                    AValue: (type == any)
                            …
                    Role == “Data”
                            ADataItem: DataItem
                                    AName: (type == text)
                                    AValue: (type == any)
                            …
            BinCategory == “ProcessData”
                    Role == “Context”
                            ADataItem: DataItem
                                    AName: (type == text)
                                    AValue: (type == any)
                            …
                    Role == “Data”
                            ADataItem: DataItem
                                    AName: (type == text)
                                    AValue: (type == any)
                            …
            …
```

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

9.2.3.4 *Additional Requirements of an AnalysisEngineDataSet*

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

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

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

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

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

9.2.3.5.1 PCSAETarget minimum cardinality is zero.

9.2.4 *Common Interface Services*

9.2.4.1 All PCS Analysis Engines shall support the following common interface services:

- A set of services shall exist to support input of data into the engine,
- A set of services shall exist to support access to data from the engine,
- A set of services shall exist to support logging and exception handling, and
- A set of services shall exist to support execution and model update.

9.2.4.2 These services will be defined further in §10. Each class in the interface could be part of an input or an output message.

9.2.5 *Common Behavior*

9.2.5.1 All PCS Analysis Engines shall support the following common behavior:

- Accept inputs, and
- Provide outputs.

9.2.6 *Analysis Engine General Capability Description*

9.2.6.1 An AE module should accept the following inputs and outputs using the common interface structure described in Figure 3. Not all inputs and outputs are required depending on the functionality needed by the AE.

9.2.7 *Inputs*—The following data types and data values are used in AE input messages.

9.2.7.1 No data types or data values are specified for the general AE.

9.2.8 *Outputs*—The following data types and data values are used in AE output messages.

9.2.8.1 No data types or data values are specified for the general AE.

9.3 *Run-to-Run Control Functional Group*

9.3.1 This section defines the common capabilities and interface data structures of the R2R Control functional group.

9.3.2 Table 9 below describes the specialization for the R2R Control functional group of each inherited general capability described in Table 5.

9.3.3 Table 10 describes the additional capabilities unique to the R2R Control functional group.

Table 11 R2R Control Functional Group Inherited Capabilities

#	*Description*	*Req'd*	*Comments*
R2R.I.1.	**Provide a R2R control capability as defined in §5** The R2R Control Analysis Engine shall be able to accept process quality data, and utilize this data to generate advice on how to improve future processing; specifically, a R2R implementation shall provide a capability for modifying recipe parameters or the selection of control parameters between runs to improve processing performance. A "run" can be a batch, lot, or an individual wafer (specified in the context).	Y	Based on definition of R2R control.
R2R.I.2.	**Provide the required PCS Analysis Engine Methods** The R2R Control Analysis Engine shall provide the required interface methods as defined in this standard.	Y	Execute, Update.
R2R.I.3.	**Provide the required PCS Analysis Engine method capabilities** The R2R Control Analysis Engine shall provide the required interface method capabilities as defined in this standard.	Y	Log Data, Exception Handling, Get Topic List, Subscribe to Topic, Get Topic Data.
R2R.I.4.	**Maintain R2R models of selected process control system environment** The R2R Control Analysis Engine shall maintain explicit or implicit R2R models of the selected process control environment. These models shall be utilized to provide the R2R control capability. These models shall dynamically represent the state of the system based on information received by the Analysis Engine relating to the selected process control environment as well as configuration information.	Y	The specific form and behavior of these models is beyond the scope of this standard; however they generally model aspects of the process(es) or tool(s) being controlled.
R2R.I.5.	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action** The R2R Control Analysis Engine shall support an input mechanism whereby the Analysis Engine can be instructed to take a specific unique action.	Y	The mechanism could be a a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model and generation of a PCS result, update of a model, or communication of model parameters.
R2R.I.6.	**Interface with multiple external systems** The R2R Control Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.
R2R.I.7.	**Basic Model Management** The R2R Control Analysis Engine shall provide the capability to create/add, edit configuration, and delete R2R control models.	Y	This capability will be supported through the Analysis Engine interface.
R2R.I.8.	**Interface with other Analysis Engines** The R2R Control Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	The R2R Control Analysis Engine is not required to interface with other Analysis Engines, however examples of use of interface capabilities would be utilizing SPC and/or FD information to determine model parameter updates or controller actions.
R2R.I.9.	**Advanced Model Management** The R2R Control Analysis Engine shall provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	

#	Description	Req'd	Comments
R2R.I.10.	**Provide the optional PCS Analysis Engine method capabilities** The R2R Control Analysis Engine shall provide all or a subset of the optional interface method capabilities as defined in this standard. See ¶10.4.1.	N	

Table 12 Additional R2R Control Functional Group Capabilities

#	Description	Req'd	Comments
R2R.A.1.	**Accept and utilize feedback data** The R2R Control Analysis Engine shall be able to accept historical information about the target process control environment, and advice adjustments for improving processing based on this data; typical feedback data is post process metrology information from previous process events on the target process. In the context of R2R control, feedback data is defined as data that is utilized to improve capability for future process runs.	Y	The controller must utilize feedback information to maintain dynamic models.
R2R.A.2.	**Automatically configure internally for control of a target process** The R2R Control Analysis Engine shall be able to request and/or accept information about the specific process control environment as necessary, and automatically configure to provide control of the process. For example, the R2R Control Analysis Engine may request and utilize experimental data to determine optimal baseline control models for the target process.	C [#1]	Item R2R.A.2 and/or R2R.A.3, required.
R2R.A.3.	**Support manual configuration for control of a target process** The R2R Control Analysis Engine shall be able to request and/or accept configuration information so that the Analysis Engine may be configured by an external system to provide control of the process.	C [#1]	Item R2R.A.2 and/or R2R.A.3, required.
R2R.A.4.	**Accept and utilize feedforward data** The R2R Control Analysis Engine shall be able to accept information about current material to be processed and adjust recommendations for improving processing based on this data. In the context of R2R control, feedforward data is defined as data that is utilized to improve capability for the current process run.	N	

[#1] C indicates "conditional" with the conditions specified in the Comment column.

9.3.4 *R2R General Capability Description*

9.3.4.1 A R2R module should accept the following inputs and outputs using the common interface structure described in Figure 5. Not all inputs and outputs are required depending on the functionality needed by the R2R model. For example, R2R function may only need feedback data in the form of Substrate Data.

9.3.5 *Inputs*—The following data types and data values are used in R2R control input messages:

9.3.5.1 The R2R Controller analysis engine inherits the input data structure and data structure requirements from the PCS Analysis Engine.

9.3.5.2 The following data types and data type values shall be supported by a R2R control analysis engine when included in an input data structure, where supported means that the AE will be able to utilize the information provided by the data type and perform any indicated action associated with that data:

9.3.5.2.1 *PCSAETarget Class*

9.3.5.2.1.1 Attribute AEType with a value of "R2R"

9.3.5.2.1.2 A list element of attribute Instructions with a value of "R2RCalculateControlAdvice"

9.3.5.2.1.3 A list element of attribute Instructions with a value of "R2RUpdateModel".

9.3.6 *Outputs* — The following data types and data type values are used in R2R control output messages:

9.3.6.1 The R2R Controller analysis engine inherits the output data structure and data structure requirements from the PCS Analysis Engine.

9.3.6.2 The following data types and data type values shall be supported by a R2R control analysis engine when included in an output data structure, where supported means that the AE will be able to provide these data types and values when behavioral conditions warrant, as indicated below.

9.3.6.2.1 *DataCollection Class* — The following DataItem Names or Name/Value pairs shall be supported for DataItems in the specified BinCategory.

Table 13 Data Types Required to be Supported by an Input Message for an Analysis Engine

BinCategory	*DataItem Name*	*Value*	*Description*	*Role* #1	*Reqd*	*Form*
Calculated Outputs	ControlAdvice	N.A. #2	A set of name-value pairs that constitute a control advice. Context associated with the AnalysisEngineDataSet Input may identify the system to which the R2R control recipe advice parameters are expected to be applied.	D	Y	any

#1 Where, in the Role column, 'D' indicates "data" and 'C' indicated "context"

#2 N.A. indicates "not applicable", because specification is on DataItem as opposed to a specific value of DataItem.

9.3.6.2.2 *ControlAdvice* — This attribute contains the control advice, structured as an unordered list of name-value pairs. The method in which the control advice is utilized is beyond the scope of this document.

9.4 *Fault Detection Functional Group*

9.4.1 This section defines the common capabilities of the FD functional group.

9.4.2 Table 11 describes the specialization for FD of each inherited general capability described in Table 5.

9.4.3 Table 12 describes the additional capabilities unique to the FD functional group.

Table 14 FD Functional Group Inherited Capabilities

#	*Description*	*Req'd*	*Comments*
FD.I.1	**Provide a Fault Detection capability as described in §5** The fault detection system shall analyze provided input data with the purpose of detecting anomalies/faults.	Y	Detailed inputs and outputs to be defined in a future version of the standard. However it is expected that the FD engine will provide output sufficient to analyze why the engine determined there was a fault.
FD.I.2	**Provide the required PCS Analysis Engine methods** The FD Analysis Engine shall provide the required interface methods as defined in this standard.	Y	Execute and Update.
FD.I.3	**Provide the required PCS Analysis Engine method capabilities** The FD Analysis Engine shall provide the required interface method capabilities as defined in this standard. .	Y	Log Data, Exception Handling, Get Topic List, Subscribe to Topic, Get Topic Data.
FD.I.4	**Maintain models of selected process control system environment(s)** The FD Analysis Engine shall maintain explicit or implicit models of the selected process control environment. These models shall be used to detect anomalies in the supplied data.	Y	The specific form and behavior of these models is beyond the scope of this standard.

#	Description	Req'd	Comments
FD.I.5	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action** The FD Analysis Engine shall support an input mechanism whereby the analysis engine can be instructed to take a specific unique action.	Y	The mechanism could be a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model and generation of a FD result.
FD.I.6	**Interface with multiple external systems** FD Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.
FD.I.7	**Basic Model Management** The FD Analysis Engine shall provide the capability to create/add models, edit configuration of models, and delete models.	Y	This capability will be supported through the FD Analysis Engine interface.
FD.I.8.1	**Interface with FC Analysis Engine** The FD Analysis Engine shall be capable of providing output that can be used by a PCS-compliant FC Analysis Engine.	Y	Output to FC is required. Other interfaces are optional.
FD.I.8.2	**Interface with other Analysis Engines** FD Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	E.g., using R2R outputs to distinguish between controller changes and faults.
FD.I.9	**Advanced Model Management** The FD Analysis Engine shall provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	
FD.I.10	**Provide the optional PCS Analysis Engine method capabilities** The FD Analysis Engine shall provide all or a subset of the optional interface method capabilities as defined in this standard.	N	

Table 15 Additional FD Functional Group Capabilities

#	Description	Req'd	Comments
FD.A.1	**Provide an interface for automatic notification of detected faults** The FD Analysis Engine shall provide an interface to automatically notify an external system when a fault is detected. The interface must provide enough information about the detected fault for the external system to make decisions about notification.	Y	The external system may for example: page or e-mail an engineer, or take the tool in question off-line. Information provided in the interface should include: context, severity, and any fault details (TBD in a future version of the standard).
FD.A.2	**Ability to test against reference/simulation data** The system should provide the ability to test and evaluate fault detection models against historic, reference or simulated data without affecting the production environment.	N	This capability could be provided off-line, i.e. as a capability that is available in a mode when other required FD capabilities are not necessarily available.
FD.A.3	**Ability to interdict process/tool** The system should be capable of initiating the interdiction of a process, either directly or indirectly, in a timely manner, in order to prevent or prohibit the creation of out-of-spec material. This is needed when the general event notification and/or interdiction system(s) cannot respond fast enough to protect people, plant, or product.	N	NOTE: This would be in addition to the capability defined in FD.I.1 above.

9.5 *Fault Classification Functional Group*

9.5.1 This section defines the common capabilities of the FC functional group.

9.5.2 Table 13 describes the specialization for FC of each inherited general capability described in Table 5 above.

9.5.3 Table 14 describes the additional capabilities unique to the FC functional group.

Table 16 FC Functional Group Inherited Capabilities

#	*Description*	*Req'd*	*Comments*
FC.I.1	**Provide a Fault Classification capability as described in §5** The Fault Classification system shall analyze provided input detected or predicted fault data with the purpose of determining the root cause of the fault.	Y	Detailed inputs and outputs to be defined in a future version of the standard.
FC.I.2	**Provide the required PCS Analysis Engine methods** The FC Analysis Engine shall provide the required interface methods as defined in this standard.	Y	Execute and Update.
FC.I.3	**Provide the required PCS Analysis Engine method capabilities** The FC Analysis Engine shall provide the required interface method capabilities as defined in this standard.	Y	Log Data, Exception Handling, Get Topic List, Subscribe to Topic, Get Topic Data.
FC.I.4	**Maintain models of selected process control system environment(s)** FC Analysis Engine shall maintain explicit or implicit models of the selected process control environment. These models shall be used to classify the input fault data into probable root cause categories.	Y	The specific form and behavior of these models is beyond the scope of this standard.
FC.I.5	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action.** The FC Analysis Engine shall support an input mechanism whereby the analysis engine can be instructed to take a specific unique action.	Y	The mechanism could be a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model and generation of a FC result.
FC.I.6	**Interface with multiple external systems** FC Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.
FC.I.7	**Basic Model Management** The FC Analysis Engine shall provide the capability to create/add models, edit configuration of models, and delete models.	Y	This capability will be supported through the FD Analysis Engine interface.
FC.I.8.1	**Interface with FD Analysis Engine** The FC Analysis Engine shall be capable of accepting the appropriate output of a PCS-compliant FD Analysis Engine.	Y	FC analysis engine may require additional inputs to perform classification operation i.e., FD output may not be sufficient as sole input.
FC.I.8.2	**Interface with FP Analysis Engine** The FC Analysis Engine shall be capable of accepting the appropriate output of any PCS-compliant FP Analysis Engine.	N	FC analysis engine may require additional inputs to perform classification operation i.e., FP output may not be sufficient as sole input.
FC.I.8.3	**Interface with other Analysis Engines** The FC Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	
FC.I.9	**Advanced Model Management** The FC Analysis Engine shall provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	

#	Description	Req'd	Comments
FC.I.10	**Provide the optional PCS Analysis Engine method capabilities** The FC Analysis Engine shall provide all or a subset of the optional interface method capabilities as defined in this standard.	N	

Table 17 Additional FC Functional Group Capabilities

#	Description	Req'd	Comments
FC.A.1	**Ability to perform automated classification** Automated classification allows for classification to occur automatically after a fault is detected or predicted (in-line). This ability results in faults classified with some degree of confidence without user intervention.	C	At least one of Items #FCA.1 and #FCA.2 is required.
FC.A.2	**Ability to perform manual classification** Manual classification allows classification to occur manually after a fault is detected or predicted (inline). This ability may result in a new classification being defined or a new association to an existing classification established.	C	At least one of Items #FC.A.1 and #FC.A.2 is required.
FC.A.3	**Ability to perform remote reclassification of production and archived faults** Re-classification should be required offline. This allows for new association to be made that may not be the result of an online classification.	N	
FC.A.4	**Ability to communicate references to troubleshooting guides (TSG)** Classification should allow users to link to an existing TSG. The result may be a reference link or a description from a list of recommended actions.	N	
FC.A.5	**Ability to tag unclassified faults for future classification** To avoid inline delays it is necessary to defer classification until further studies are made. This may be the result of automated, manual, and re-classification tasks.	N	
FC.A.6	**Produce list of classifications by a method such as confidence or probability** Some metric to identify the classification confidence must be the result of classification. Optionally all the classifications that have any nominal confidence should be listed.	N	

9.6 *Fault Prediction Functional Group*

9.6.1 This section defines the common capabilities of the FP functional group.

9.6.2 Table 15 describes the specialization for FP of each inherited general capability described in Table 5 above.

9.6.3 Table 16 describes the additional capabilities unique to the FP functional group.

Table 18 FP Functional Group Inherited Capabilities

#	Description	Req'd	Comments
FP.I.1	**Provide a Fault Prediction capability as described in §5** The Fault Prediction system shall analyze provided input data with the purpose of determining if a fault will occur in a future run.	Y	Detailed inputs and outputs to be defined in a future version of the standard. However, it is expected that the FP output will provide enough information to determine why and when the engine determined a fault will occur.

#	Description	Req'd	Comments
FP.I.2	**Provide the required PCS Analysis Engine methods** The FP Analysis Engine shall provide the required interface methods as defined in this standard.	Y	Execute and Update.
FP.I.3	**Provide the required PCS Analysis Engine method capabilities** The FP Analysis Engine shall provide the required interface method capabilities as defined in this standard.	Y	Log Data, Exception Handling, Get Topic List, Subscribe to Topic, Get Topic Data.
FP.I.4	**Maintain models of selected process control system environment(s)** FP Analysis Engine shall maintain explicit or implicit models of the selected process control environment. These models shall be used to predict faults based on the supplied input data.	Y	The specific form and behavior of these models is beyond the scope of this standard.
FP.I.5	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action** The FP Analysis Engine shall support an input mechanism whereby the Analysis Engine can be instructed to take a specific unique action.	Y	The mechanism could be a a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model and generation of a FP result.
FP.I.6	**Interface with multiple external systems** FP Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.
FP.I.7	**Basic Model Management** The FP Analysis Engine shall provide the capability to create/add models, edit configuration of models, and delete models.	Y	This capability will be supported through the FD Analysis Engine interface.
FP.I.8.1	**Interface with FC Analysis Engine** The FP Analysis Engine shall be capable of providing output that can be used by any PCS-compliant FC Analysis Engine.	Y	
FP.I.8.2	**Interface with other Analysis Engines** The FP Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	
FP.I.9	**Advanced Model Management** The FP Analysis Engine shall provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	
FP.I.10	**Provide the optional PCS Analysis Engine method capabilities** The FP Analysis Engine shall provide all or a subset of the optional interface method capabilities as defined in this standard.	N	

Table 19 Additional FP Functional Group Capabilities

#	Description	Req'd	Comments
FP.A.1	**Provide an interface for automatic notification of predicted faults** The system must provide an interface to automatically notify an external system when a fault is predicted. The interface must provide enough information about the predicted fault for the external system to make decisions about notification**.**	Y	The external system may for example: page or e-mail an engineer, or take the tool in question off-line. Information provided in the interface should include: context, severity, and any fault details (TBD in a future version of the standard).

#	*Description*	*Req'd*	*Comments*
FP.A.2	**Ability to perform advanced statistical/prediction algorithms on data to enable fault prediction after a specific number of runs, tool time or products** What is the probability that a specific fault will happen, when, what are action plans that can be taken, which tools, what is the predictive classification of a specific fault.	N	
FP.A.3	**Ability to build a prediction model from historical data** Based on the historical data, a learning-based model may be developed. A specific pattern of historical data may give information about a future fault possibility.	N	
FP.A.4	**Ability to predict preventive maintenance scheduling based on tool/substrate information** This will reduce tool downtime.	N	

9.7 *Statistical Process Control Functional Group*

9.7.1 This section defines the common capabilities of the SPC functional group.

9.7.2 Table 17 below describes the specialization for SPC of each inherited general capability described in Table 5 above.

9.7.3 Table 18 describes the additional capabilities unique to the SPC functional group.

Table 20 SPC Functional Group Inherited Capabilities

#	*Description*	*Req'd*	*Comments*
SPC.I.1	**Provide a Statistical Process Control capability as described in §5** The Statistical Process Control system shall analyze provided input data with the purpose of determining if statistically significant deviations from normal have occurred.	Y	Detailed inputs and outputs to be defined in a future version of the standard. However, it is expected that the SPC output will provide enough information to determine why the engine determined a deviation occurred.
SPC.I.2	**Provide the required PCS Analysis Engine methods** The SPC Analysis Engine shall provide the required interface methods as defined in this standard.	Y	Execute and Update.
SPC.I.3	**Provide the required PCS Analysis Engine method capabilities** The SPC Analysis Engine shall provide the required interface method capabilities as defined in this standard. .	Y	Log Data, Exception Handling, Get Topic List, Subscribe to Topic, Get Topic Data
SPC.I.4	**Maintain models of selected process control system environment(s)** SPC Analysis Engine shall maintain explicit or implicit models of the selected process control environment. These models shall be used to determine if statistically significant deviations have occurred.	Y	The specific form and behavior of these models is beyond the scope of this standard.
SPC.I.5	**Provide a mechanism to uniquely instruct the Analysis Engine to perform an action.** The SPC Analysis Engine shall support an input mechanism whereby the analysis engine can be instructed to take a specific unique action.	Y	The mechanism could be a single key variable, e.g., model name or a complex set of context data utilized internally to perform context matching, etc. The action could include application of a specific model (SPC rule) and generation of a SPC result.
SPC.I.6	**Interface with multiple external systems** SPC Analysis Engine shall be capable of accepting inputs from, or generating outputs to multiple external systems.	Y	The definition of these systems will be a function of the application environment; these could include process tool or metrology systems, or user interfaces.

#	Description	Req'd	Comments
SPC.I.7	**Basic Model Management** The SPC Analysis Engine shall provide the capability to create/add models, edit configuration of models, and delete models.	Y	This capability will be supported through the FD Analysis Engine interface.
SPC.I.8	**Interface with other Analysis Engines** The SPC Analysis Engine shall be capable of either accepting outputs from other Analysis Engines as inputs, or producing outputs that are utilized by other Analysis Engines.	N	
SPC.I.9	**Advanced Model Management** The SPC Analysis Engine may provide advanced model management capabilities including but not limited to: secure access to models, and automatic updating of models based on process control environment changes.	N	
SPC.I.10	**Provide the optional PCS Analysis Engine method capabilities** The PCS Analysis Engine shall provide all or a subset of the optional interface method capabilities as defined in this standard.	N	

Table 21 Additional SPC Functional Group Capabilities

#	Description	Req'd	Comments
SPC.A.1	**Ability to perform trend analysis** Automated analysis of data for the identification of undesired trends using industry standard run rules such as (for example) Western Electric. One result of these calculations and comparisons should be a decision of whether or not the new data caused the "chart" to go out of control. The mathematics available must include those items listed in #SPC.A.4 below.	Y	
SPC.A.2	**Ability to support standard control charts** Provide ability to plot data as well as display traditional variables charts, and attributes charts. The system should also provide access to the data to enable a third party system to display control charts.	Y	
SPC.A.3	**Ability to perform statistical analysis** These tools may include the ability to calculate statistical metrics, normal analysis, and non-normal analysis.	Y	
SPC.A.4	**Ability to calculate control limits** The calculation of control limits for control charts from historical data using industry standard techniques for calculating upper and lower control limits for variables and attributes charts based on sub-group size, sample size, etc.	Y	
SPC.A.5	**Provide an interface for automatic notification of processing anomalies** The system must provide an interface to automatically notify an external system when an assignable cause statistical anomaly is detected. The interface must provide enough information about the detected problem for the external system to make decisions about notification.	Y	The external system may for example: page or e-mail an engineer, or take the tool in question off-line. Information provided in the interface should include: context, severity, and any fault details.
SPC.A.6	**Ability to provide a reference to an Out-of-Control Action Plan** The system should store the action plan that is to be put into effect in the case of the trend analysis determining that an out of control condition exists. Ideally this would have at least a full page of storage and an optional URL linking capability (reference being web link).	Y	

#	Description	Req'd	Comments
SPC.A.7	**Ability to store user problem resolution comments** Provide a place for users to put in comments on how the problem was corrected. This may also be used by the statistical analysis system to store information related to the OOC point to speed diagnosis and problem resolution.	N	

10 Method Capabilities and Behavior

10.1 All PCS functional groups inherit certain capabilities, methods, and behavior from the general PCS Interface. This interface provides support primarily for general execution and publish/subscribe communications. Publish/subscribe is a type of communications where data produced by an application process can be consumed by a number of application processes that belong or "subscribe to" the communication environment. Additional information on the publish/subscribe communications approach can be found in reference §4.

10.2 This standard specifies both required 'methods' as well as 'capabilities', which must be supported by any PCS Analysis Engine. Support for the required methods must use the defined method names, as well as the specified argument and return data types. Most Arguments and Returns must be of the type AnalysisEngineDataSet. The AnalysisEngineDataSet type is a set of aggregate data structures as defined in §9. Support for the stated "methods" must be provided as specified. Support for "method capabilities" must be provided by any PCS-conformant functional group, but how these capabilities are supplied – including specific arguments and returns, method, argument and return names, and argument and return data types - are either beyond the scope of this standard or specified elsewhere in this standard, e.g., in specification of the interface data structure. The data types used as the Arguments and Returns for each capability are general data types, and are guidelines only.

10.3 The general PCS Interface required methods and capabilities are summarized in the table below and described more fully in the accompanying sections.

10.4 *Common Analysis Engine Method Capabilities and Behavior*

10.4.1 *Common Analysis Engine Method Capabilities*

10.4.1.1 The capabilities described in the following section and Table XYZ are required in any PCS-compliant Analysis Engine. However, the form of their implementation is not a part of this standard. The approach by which the method capability is accessed by a client, be it via method invocation, database queries and triggers, etc, is subject to the decisions of the implementer. Also, the exact data types used for the Arguments and Returns are up to the developer. Additional arguments or return types may also be required depending on the implementation technology.

10.4.1.2 These method capabilities are generalizations of "Publish/Subscribe" interfaces. Note that there is not necessarily a one-to-one correspondence between method capabilities as defined herein and methods in an implementation of these capabilities.

Table 22 Common PCS Analysis Engine Method Capabilities

#	Capability	Req'd	Behavior/Comment
1	Produce Log Data	Y	Allows a client to have access to logging information conveyed through the log messages of the PCS compliant analysis engine. Arguments passed with a log data request could relate the date range and context associated with the request, while information returned could include the status along with the requested log data.
2	Produce Exception Data	Y	Allows a client to access or be informed of exceptions raised by the PCS system. The exception information would be returned by the AE to the requestor, and could include a date range, similar to log data.

10.4.2 *Common Analysis Engine behavior*

10.4.2.1 No additional common AE behavior is specified.

10.5 *R2R Control AE Method Capabilities and Behavior*

10.5.1 *R2R Control AE Method Capabilities*

10.5.1.1 No method capabilities for this AE type are specified.

10.5.2 *R2R Control AE Behavior*

10.5.2.1 The R2R control behavior defines the operation of the R2R Control AE in providing capabilities. The interface (methods, capabilities and behavior) is inherited from the common PCS interface. Inherited methods are identical in form to the methods required in the Analysis Engine interface. However, their implementation is specific to R2R Control analysis engines.

10.5.2.2 If a R2R AE receives an input message of type AnalysisEngineDataSet for a process control job, and, in a PCSAETarget attribute of that message, one of the list of instructions in the Instructions attribute has a value of "R2RCalculateControlAdvice", then the R2R control AE shall provide an output message of type AnalysisEngineDataSet that contains the data item name "ControlAdvice" in a DataCollection where BinCategory == "CalculatedOutputs".

10.5.2.3 If a R2R AE receives an input message of type AnalysisEngineDataSet for a process control job, and, in a PCSAETarget attribute of that message, one of the list of instructions in the Instructions attribute has a value of "R2RUpdateModel", then the R2R control AE shall utilize information provided as part of the associated PCS job to update control models as necessary. The determination of need to update models and the method in which these models are updated is beyond the scope of this standard.

11 Compliance to PCS Standard

11.1 This section defines requirements placed on process control systems for claiming compliance with this standard and verifying compliance with this standard. The standard does NOT include specific verification test plans, but there should be enough information in the standard to generate these.

11.2 *Approach*

11.2.1 This standard specifies the interfaces for PCS functional groups. Compliance with this standard is verified through testing these interfaces to verify compliance of data structures and behavior to the extent specified in this standard.

11.2.2 As noted in §9, all PCS Analysis Engines are characterized as having a set of capabilities, and a collection of inputs and outputs, with the outputs being a function of the inputs. The general approach to compliance testing then is to first document the capabilities, and then test the inputs and outputs by (1) generating inputs for the PCS Analysis Engine to consume and (2) observing corresponding outputs of the engine. It is understood that there are a number of behavior models that an Analysis Engine shall use to consume inputs and produce outputs, such as active polling by the engine or a request message from an outside source to the Analysis Engine. The compliance testing method should utilize a mechanism that works with the behavioral model of the Analysis Engine for subscription to (i.e., consumption of) inputs and publication of (i.e., production of) outputs.

11.2.3 Note also that all PCS Analysis Engines inherit a set of common requirements. Thus, the approach for testing any PCS Analysis Engine should be to first verify compliance to the common requirements, and then verify compliance to the requirements specific to the particular functional group.

11.3 *Scope of Compliance Verification*

11.3.1 Compliance Verification shall address the following:

- Documentation of functional group capabilities

11.3.2 Future version of this standard will also address the following compliance verification:

- Capability to accept all required inputs
- Capability to accept properly formatted interface data structures
- Capability to produce all required outputs; these outputs may be in response to the inputs as specified herein
- Capability to produce properly formatted output data structures

11.3.3 Compliance verification will not address the following:

- Engine response to improperly formatted inputs
- Quality of Service (QoS) of data in output structures produced by engine
- Quality of Performance (QoP) of engine in producing output structures (e.g., in response to receipt of input structures)

11.4 *Statement of Compliance (SoC): Functional Group Capabilities Data Sheet*

11.4.1 All PCS Analysis Engines claiming compliance with this standard are required to be delivered with a statement of compliance sheet on functional group capabilities. This sheet specifies for the user which functional capabilities, specified as required or conditional in this standard, are available with this engine. The format of the SoC sheet is given in Table 21, which includes a description of the fields of this sheet:

Table 23 PCS Analysis Engine SoC Functional Group Capabilities Data Sheet

Company	*Company Name*			
Product	*Product Name*			
Version	*Software Revision Number, etc.*			
Functional Group(s)	*Functional Group(s) to which Compliance is claimed*			
Functional Group Capabilities				
- Functional Group #1				
#	**Description**	**Requirement**	**Supported**	**Comments**
- Functional Group #2				
#	**Description**	**Requirement**	**Supported**	**Comments**

11.4.2 *Company* — Name of company supplying product.

11.4.3 *Product* — Name of product being supplied.

11.4.4 *Version* — Any additional descriptors that supplier applies to product (such as software revision number) that, along with Product Name, uniquely identifies product.

11.4.5 *Functional Group(s)* — The functional groups, as defined in this standard, to which compliance is being claimed.

11.4.6 *Functional Group Capabilities* — The functional group capabilities as specified in this standard for the functional group(s) to which compliance is being claimed (as indicated in the "Functional Group(s)" field). Note that *all* capabilities specified in this standard for the functional groups, to which compliance is being claimed, must be listed, and are organized by the fields below.

11.4.7 *Functional Group #n* — The name of the 'n-th' functional group to which compliance is being claimed, e.g., "R2R", "FD", "FC", "FP", or "SPC".

11.4.8 *#* — The index number (copied from the "#" field for that capability).

11.4.9 *Description* — The functional group capability description copied verbatim from this standard (copied from the "Description" field for that capability), for the PCS analysis engine functional group listed.

11.4.10 *Requirement* — The requirement of the capability as defined in this standard (copied from the "Requirement" field for that capability).

11.4.11 *Supported* — This field contains the PCS analysis engine provider's claim of support of this capability. The claim could be "Yes" ("Y"), "No" ("N"), "Conditional" ("C"), or "Optional" ("O"). Note that if the capability is listed as a requirement then it must be supported.

11.4.12 *Comment* — This field contains any additional information that the PCS analysis engine provider may wish to provide. Note that if the “Supported” field is listed as “Conditional” or “Optional”, the conditionality or optionality conditions must be delineated in this field.

11.5 *PCS Analysis Engine Compliance Verification Procedure*

11.5.1 The following procedure should be applied to all PCS Analysis Engines:

11.5.2 *Compliance: Functional Group Capabilities Data Sheet Verification Test:*

- Is a properly formatted SoC data sheet supplied with product?

Pass Criteria:

- A properly formatted SoC data sheet, as specified in ¶11.3 is provided with product.
- The Product + Version fields uniquely identify product.
- At least one functional group is claimed, i.e., the engine declares to belong to at least one functional group as specified in this standard.
- All functional group capabilities for each functional group claimed in the “Functional Group” field are listed as defined in §9, and the requirement field matches that provided for each capability in §9.
- For each capability listed as required, it must also be supported (i.e., if Requirement == ‘Y’, then Supported ==’Y’).
- For each capability listed as conditional, the conditions, defined in §9, must be met.
- For each capability listed as optional (i.e., not required), a declaration of whether the requirement is supported must be made (i.e., the “Supported” field for this requirement must be filled in).

11.6 *Condition of Compliance*

11.6.1 A PCS Analysis Engine can be considered to be compliant with this standard if and only if it passes all of the compliance tests for the Analysis Engine in ¶11.5.

SEMI E134-0305
SPECIFICATION FOR DATA COLLECTION MANAGEMENT

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 July 11, 2004 and August 16, 2004. Initially available at www.semi.org September 2004; to be published November 2004. Originally published July 2004.

E This standard was editorially modified in November 2004 to correct editorial errors in Figures 21 and 22.

NOTICE: The designation of SEMI E134 was updated during the 0305 publishing cycle to reflect the creation of SEMI E134.1.

1 Purpose 2
2 Scope 2
2.1 In-scope 2
2.2 Out-of-scope 2
3 Limitations....2
3.1 Abstract Model 2
4 Referenced Standards 3
4.1 Non-SEMI Standards....3
4.2 SEMI Standards....3
5 Terminology 3
5.1 Abbreviations and Acronyms....3
5.2 Definitions 3
6 Conventions....4
6.1 Terms 4
6.2 Notation 4
7 Background....6
7.1 Equipment Data Collection 6
8 Overview 8
8.1 Data Collection Concepts....8
8.2 Trace Data....8
8.3 Events 8
8.4 Exceptions....9
8.5 Data Collection Plans....9
8.6 Equipment Operational Performance Monitoring....9
8.7 Ad-hoc Data Requests....9
9 Data Collection Plan Management 10
9.1 DataCollectionManager Interface....10
10 DCP Privilege Model....27
10.1 Privileges....27
11 Data Collection Plan Definition....29
11.1 DataCollectionPlan 29
12 Data Collection Plan State Models 35
12.1 DataCollectionPlan State Model 35
12.2 DataCollectionPlan-Active State Model....38
12.3 Trace Request State Model 41
13 Operational Performance Monitoring....44
13.1 Operational Performance Conditions State Model 44
14 Data Collection Report Formats 45
14.1 Data Collection Reports 45
14.2 Parameter Values 50
14.3 Primitive Parameter Values....51
14.4 Structured and Enumerated Parameter Values 53
14.5 Time Format 55
15 DCP Consumer Interface....56

15.2 DCPConsumer 56
16 Requirements for Compliance 60

1 Purpose

1.1 This specification describes a method for data acquisition consumers to request process and operational data from equipment to be communicated in an automated fashion by the equipment, or in an ad-hoc request from the consumer. It includes a mechanism for organizing related data into groups to make it more straightforward for enabling or disabling a large number of data sources, and to allow consumers to organize related data into groups according to their purpose.

2 Scope

2.1 *In-scope*

2.1.1 This specification provides a means to acquire event, exception, and trace data from semiconductor equipment through the use of a named data collection plan. This specification defines what form a data collection plan takes, the meaning of its contents, and an interface for managing them that is to be supported by the equipment.

2.1.2 This specification defines the behavior associated with the execution of data collection plans in the form of finite state machines. The formats for data produced as a result of executing a data collection plan are also defined, as is the interface that must be supported by consumers of data collection plan output.

2.1.3 This specification defines a way for the equipment to notify consumers when the combination of activities on the equipment, including data acquisition, are causing the equipment to perform below supplier-defined criteria.

2.1.4 This specification defines a way for consumers to make ad-hoc on-demand requests for data from the equipment, outside of a data collection plan.

2.2 *Out-of-scope*

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

3 Limitations

3.1 *Abstract Model*

3.1.1 This specification does not define a convention for naming or identifying sources, parameters, events, or exceptions. This document assumes that all such entities can be identified separately and represented as text. Such conventions shall be fully specified by any implementation mapping of this specification.

3.1.2 This specification does not define the mechanism used for identifying or authenticating consumers or enforcing privileges. It assumes that consumers can be uniquely identified by a text value, and that the equipment is capable of enforcing the standard privileges defined in this document. Such mechanisms shall be fully specified by any implementation mapping of this specification.

3.1.3 This specification does not define the mechanism used for determining how to locate consumers that are to receive the events defined by this specification. It assumes that the equipment has established a communications context with any consumers that can submit requests and has provided any context necessary to send events to those consumers. Such mechanisms shall be fully specified by any implementation mapping of this specification.

3.1.4 This specification does not define any algorithms or heuristics to use for detecting degrading equipment operational performance. Such techniques are specific to the equipment's internal hardware and software architecture and will very likely be unique to each class and/or model of equipment. It assumes that it is possible to detect such conditions and to identify which systems comprising the equipment are affected.

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

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E40 — Standard for Processing Management

SEMI E94 — Provisional Specification for Control Job Management

4.2 *Non-SEMI Standards*

International Standards Organization (ISO) 11578:1996[1] — Information technology Open Systems Interconnection – Remote Procedure Call definition of UUID

ISO 8601[2] — Representations of dates and times, 1988-06-15

ISO 8601 Draft Revision — Representations of dates and times, draft revision, 2000

Unified Modeling Language (UML) Specification, Version 1.4, OMG Specification 01-09-67[3]

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

5 Terminology

5.1 *Abbreviations and Acronyms*

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

5.1.2 *DCP* — Data Collection Plan

5.1.3 *UML* — Unified Modeling Language

5.2 *Definitions*

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

5.2.2 Related Information, Section 1, contains useful definitions of UML terms taken directly from the UML standard. UML terms used in this document conform to these definitions. Please refer to this Section as needed.

5.2.3 *collection result* — the set of data values obtained during trace data collection.

5.2.4 *collection frequency* — the rate at which the collection of one or more data values is performed. This is not the same as the frequency with which the equipment internally samples these data from its components.

5.2.5 *data source* — a physical or logical entity associated with the equipment that is capable of providing data values independently of other equipment entities.

5.2.6 *event source* — a physical or logical entity associated with the equipment that is capable of generating events independently of other equipment entities

5.2.7 *exception source* — a physical or logical entity associated with the equipment that is capable of generating exceptions independently of other equipment entities

5.2.8 *parameter source* — a physical or logical entity associated with the equipment that is capable of providing parameters independently of other sources. This term may be used interchangeably with 'data source'.

5.2.9 *source id* — a name or other token that uniquely identifies a specific origin or producer of information from among possible sources.

1 http://www.iso.ch/cate/d2229.html

2 http://www.iso.ch/cate/d26780.html

3 http://www.omg.org/technology/documents/modeling_spec_catalog.htm