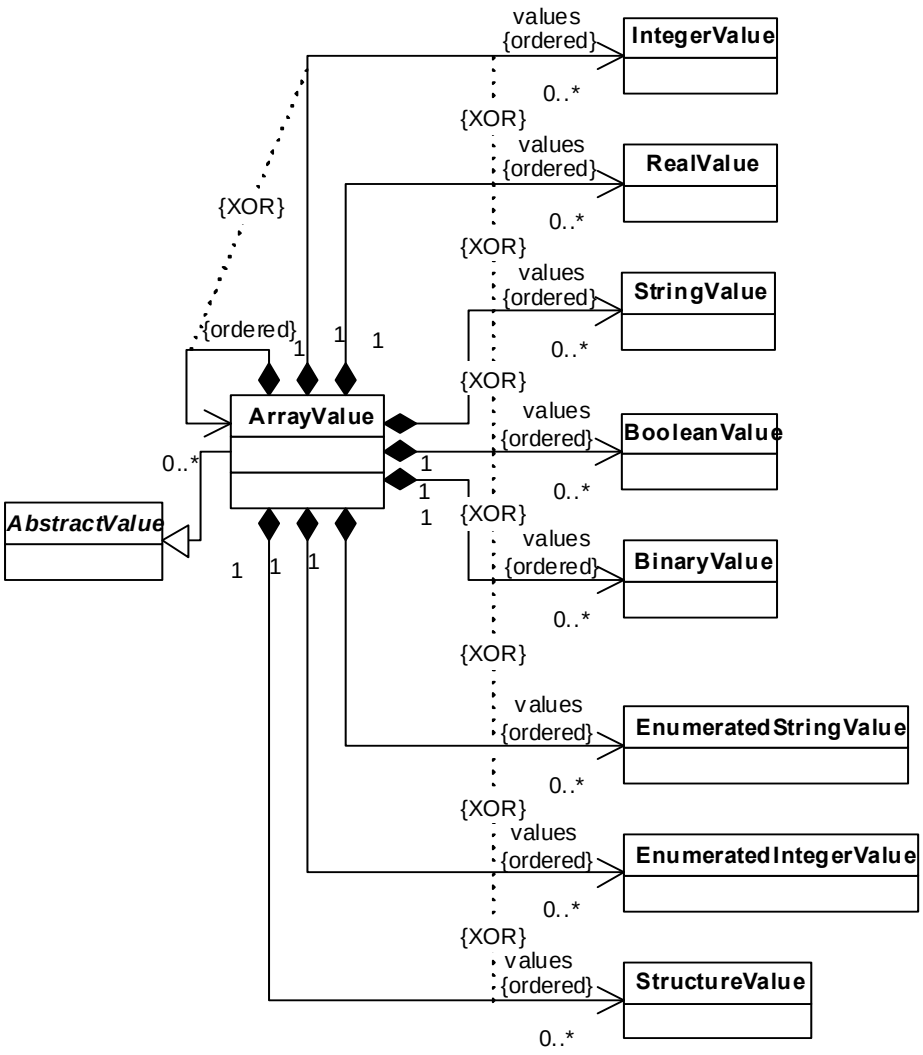

Figure 31
Array Values

14.4.3 Figure 31 shows how to represent array values. An array is an ordered collection of zero or more elements, each element of the same type. In this case the UML {XOR} constraint is used to show that arrays must contain values of one type only (integers, reals, strings, etc.). Note also that array values can be nested (each element of an array can be an array).

14.4.3.1 *ArrayValue* — Represents an array of one or more values of the same type.

14.4.3.2 *ArrayValue Attribute Definition Table*

Table 75 ArrayValue Attribute Definition

Attribute Name	*Definition*	*Form*
values	The values of the array.	Ordered list of elements, each of the same specific primitive, enumerated, structure, or array type.

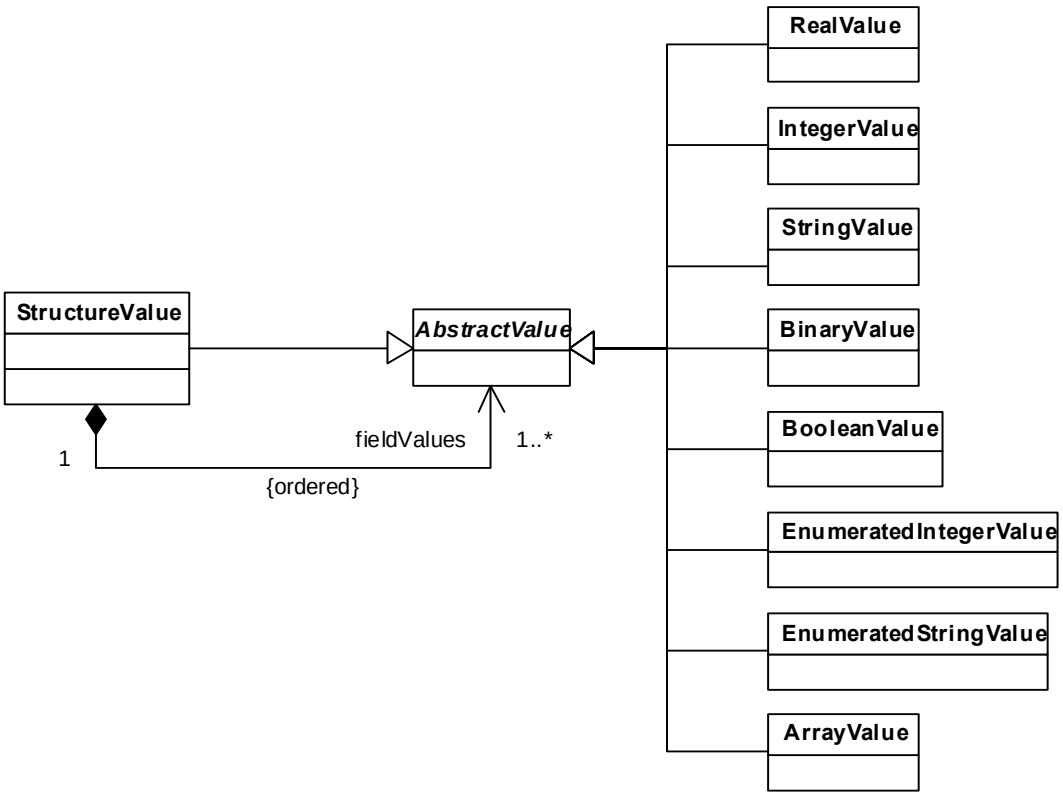

Figure 32
Structure Values

14.4.4 Figure 32 shows how to represent structure values. A structure is an ordered collection of one or more elements, each element of a potentially different type. Note also that structure values can be nested (any element of a structure can be a structure).

14.4.4.1 *StructureValue* — Represents a structure of one or more values, each of which may be a different type.

14.4.4.2 *StructureValue Attribute Definition Table*

Table 76 StructureValue Attribute Definition

Attribute Name	*Definition*	*Form*
fieldValues	The values of the structure.	Ordered list of elements, each of a specific, potentially different, primitive, enumerated, structure, or array type.

14.5 *Time Format*

14.5.1 This specification uses the ISO 8601 format for representing instants in time. ISO 8601 permits a variety of possible formats for the same time value, so it is necessary to establish a convention for the format used by time values in this document.

14.5.2 For representing time values, this document restricts the string representation of time to the form CCYY-MM-DDThh:mm:ss.fff[-+]hh:mm. 'CC' is the century (first century starts at '00'), 'YY' is the year, 'MM' is the month, and 'DD' is the day, each represented as a two-digit number with leading zeroes as needed. For years beyond 9999, additional digits can be added to the 'CC' field. 'T' separates date information from time information. 'hh' is the hour, 'mm' are the minutes, 'ss' are the seconds, and 'fff' are the sub-seconds, supporting up to millisecond precision in local time. If equipment can't measure time with millisecond precision, it shall fill in the sub-second field with zeroes as appropriate up to the measurable precision (for example, if time can be measured

with centi-second precision, all time values would contain a zero in the third column of the sub-second field). '[-+]hh:mm' describes the difference from Coordinated Universal Time expressed as either +hh:mm or –hh:mm.

15 DCP Consumer Interface

15.1 Figure 33 shows the interface used to communicate data collection results and other notifications to a DCP consumer, as well as the data types of the arguments provided in each operation.

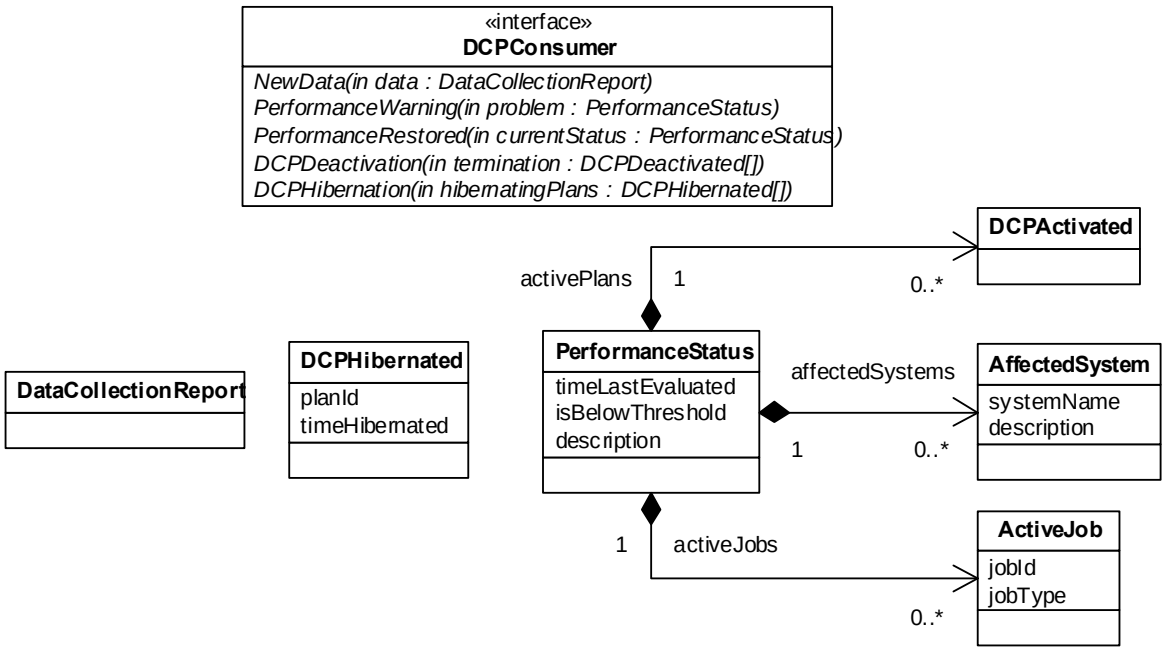

Figure 33
DCPConsumer Interface and Argument Types

15.2 *DCPConsumer*

15.2.1 The DCPConsumer interface allows the equipment to provide DCP consumers with data collection results, operational performance warnings, and DCP deactivation notifications. All DCP consumers must implement this interface.

15.2.2 The notifications defined for the DCPConsumer interface are Fire-and-Forget messages. No replies from consumers are specified. The equipment is not required to take action on any protocol-level replies or errors received from consumers when sending these notifications, unless otherwise required by an implementation specification. The equipment is not required to retry or buffer notifications if it determines that such notifications are not being successfully delivered to a consumer. The equipment shall not deactivate any consumer's DCPs if it detects an error in the delivery of these notifications.

15.2.3 *DCPConsumer Operations*

Table 77 DCPConsumer Operation Definition

Operation	*Description*	*Type*	*Requestor/Sender*	*Responder/ Receiver*
NewData	Communicates data collection results from an active DCP to the consumer that activated the DCP.	FF	Equipment	Consumer
PerformanceWarning	Warns the consumer when the equipment has detected degradation of its performance (see Section 13).	FF	Equipment	Consumer

Operation	*Description*	*Type*	*Requestor/Sender*	*Responder/ Receiver*
PerformanceRestored	Notifies the consumer when the equipment has detected a return to normal performance conditions (see Section 13).	FF	Equipment	Consumer
DCPDeactivation	Notifies the consumer when the equipment is deactivating a DCP previously activated by that consumer.	FF	Equipment	Consumer
DCPHibernation	Notifies the consumer when the equipment is placing one or more persistent DCPs in a hibernation state as part of a shutdown sequence.	FF	Equipment	Consumer

15.2.4 *NewData* — The equipment shall send this notification whenever data from an active DCP is available for the consumer that activated the DCP.

15.2.4.1 *NewData Operation Arguments*

Table 78 NewData Argument Definitions

Argument	*Description*	*Kind*	*Form*
dataCollectionReport	The data originating from the active DCP.	in	Structured data, of type DataCollectionReport, described in Section 14.1.

15.2.5 *PerformanceWarning* — The equipment shall send this notification to all consumers with ManageOnlyAuthoredDCPs privilege or higher whenever operational performance has degraded below a supplier-specified threshold (see Section 13.1.7).

15.2.5.1 *PerformanceWarning Operation Arguments*

Table 79 PerformanceWarning Argument Definitions

Argument	*Description*	*Kind*	*Form*
warning	Information describing the context and the equipment components affected by the performance problem.	in	Structured data, of type PerformanceStatus, described in Section 15.1.6.1.1.

15.2.6 *PerformanceRestored* — The equipment shall send this notification to all consumers with ManageOnlyAuthoredDCPs privilege or higher when operational performance has returned above a supplier-specified threshold (see Section 13.1.7).

15.2.6.1 *PerformanceRestored Operation Arguments*

Table 80 PerformanceRestored Argument Definitions

Argument	*Description*	*Kind*	*Form*
status	Information describing the current performance status.	in	Structured data, of type PerformanceStatus, described in Section 15.1.6.1.1.

15.2.6.1.1 *PerformanceStatus* — This class, shown in Figure 33, describes the current status of the equipment's ability to perform its intended function. It includes the time the performance status was last evaluated, whether or not the performance is below the equipment-defined threshold, and a description of the current status. See Section 13 for further details.

15.2.6.1.2 *DegradedPerformance Attribute Definition Table*

Table 81 DegradedPerformance Attribute Definition

Attribute Name	*Definition*	*Form*
timeLastEvaluated	The time at which the equipment performance was last evaluated.	Text, formatted according to Section 14.5.
isBelowThreshold	Whether or not the performance falls below the equipment-defined threshold.	Boolean.
description	If performance is below threshold, a human-readable description of any details known about the root cause.	Text.

15.2.6.1.3 *PerformanceStatus Association Definition Table*

Table 82 PerformanceStatus Association Definition

Association Role Name	*Definition*	*Comments*
activePlans	List of DCPs active at the time the performance was last evaluated.	Unordered list of elements of type DCPActivated, described in Section 9.1.2.8.4.
activeJobs	List of jobs active at the time the performance was last evaluated.	Unordered list of elements of type ActiveJob, described in Section 15.1.6.1.4.
affectedSystems	List of equipment systems affected by the performance issue, if below threshold.	Unordered list of zero or more elements of type AffectedSystem, described in Section 15.1.6.1.6. If performance is below threshold, at least one AffectedSystem must be listed.

15.2.6.1.4 *ActiveJob* — This class, shown in Figure 33, represents an active job that can be uniquely identified among all active jobs on the equipment.

15.2.6.1.5 *ActiveJob Attribute Definition Table*

Table 83 ActiveJob Attribute Definition

Attribute Name	*Definition*	*Form*
jobId	The unique identifier of the job.	Text, formatted according to the convention used to identify jobs.
jobType	The type of job identified by jobId.	Text, equal to any of the following values: “urn:semi-org:E94:ControlJob”, “urn:semi-org:E40:ProcessJob”, “urn:semi-org:E30:ppSelect”, “urn:semi-org:E30:rcpSelect”

15.2.6.1.6 *AffectedSystem* — This class, shown in Figure 33, represents an equipment component that is negatively impacted by the decrease in performance, and provides a description of how it is being affected.

15.2.6.1.7 *AffectedSystem Attribute Definition Table*

Table 84 AffectedSystem Attribute Definition

Attribute Name	*Definition*	*Form*
equipmentComponent	The unique name of the affected equipment component.	Text, formatted according to the convention used for identifying an equipment component.
description	A description of how the system is being affected by the performance decrease.	Text.

15.2.6.2 *DCPDeactivation* — The equipment shall send this notification to a consumer whenever one or more DCPs that were activated by that consumer have been deactivated. Note that the consumer should make no assumption regarding the timing of the receipt of this notification and the receipt of the last transmitted data for any deactivated DCPs provided in the notification. The consumer may receive one or more data collection reports from the named DCPs after receiving this notification.

15.2.6.2.1 *DCPDeactivation Operation Arguments*

Table 85 DCPDeactivation Argument Definitions

Argument	*Description*	*Kind*	*Form*
deactivationNotice	Information describing the DCPs that have been deactivated.	in	List of one or more structured data elements, each of type DCPDeactivated, described in Section 9.1.2.7.6.

15.2.6.3 *DCPHibernation* — The equipment shall send this notification to a consumer whenever one or more DCPs that were activated by that consumer are going into the hibernating state. Note that the consumer should make no assumption regarding the timing of the receipt of this notification and the receipt of the last transmitted data for any deactivated DCPs provided in the notification. The consumer may receive one or more data collection reports from the named DCPs after receiving this notification.

15.2.6.3.1 *DCPHibernation Operation Arguments*

Table 86 DCPHibernation Argument Definitions

Argument	*Description*	*Kind*	*Form*
hibernatingPlans	Information describing the DCPs that are going into hibernation.	in	List of one or more structured data elements, each of type DCPHibernated, described in Section 15.1.6.3.2.

15.2.6.3.2 *DCPHibernated* — This class, shown in Figure 33, describes a persistent DCP that the equipment has placed in hibernation.

15.2.6.3.3 *DCPHibernated Attribute Definition Table*

Table 87 DCPHibernated Attribute Definition

Attribute Name	*Definition*	*Form*
planId	Identifies the DCP that is in hibernation.	Text, equal to the 'id' attribute of the DataCollectionPlan that is hibernating.
timeHibernated	The time at which the plan was placed in hibernation.	Text, formatted according to Section 14.5.

16 Requirements for Compliance

16.1 Table 88 provides a checklist for Data Collection Management compliance. Note that the DCP State Models described in Section 12 are internal to the equipment and have no SEMI-specified technology mapping.

Table 88 Data Collection Management Compliance Statement

Fundamental Requirements	*Section*	*Implemented using SEMI technology mapping*	*Implementation complies with specification*	*Implementation complies with technology mapping*
DCM Interface	9	□ Yes □ No	□ Yes □ No	□ Yes □ No
DCP Privileges	10	□ Yes □ No	□ Yes □ No	□ Yes □ No
DCP Definition	11	□ Yes □ No	□ Yes □ No	□ Yes □ No
DCP State Models	12	N/A	□ Yes □ No	N/A
Operational Performance Monitoring	13	□ Yes □ No	□ Yes □ No	□ Yes □ No
Data Representation	14	□ Yes □ No	□ Yes □ No	□ Yes □ No
DCP Notifications	15	□ Yes □ No	□ Yes □ No	□ Yes □ No

RELATED INFORMATION 1
DATA COLLECTION CONTEXT

NOTICE: This related information is not an official part of SEMI Exxx and was derived from the work of the originating committee. This related information was approved for publication by full letter ballot procedures.

R1-1 Overview

R1-1.1 This Related Information discusses some of the key requirements and expectations of the equipment for applications that make use of Data Collection Management services for performing near-real-time analysis of time series (trace) data from equipment. The essential goal is for the equipment to provide sufficient, correct, and timely contextual data to such applications in order to facilitate correct and timely analysis of trace data.

R1-1.2 *Typical Analysis Application Example*

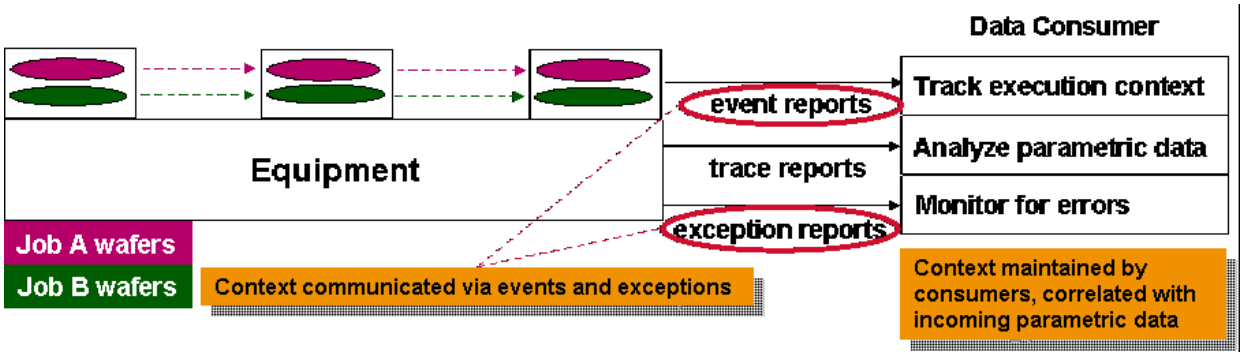

Figure R1-1
Typical Time-Series Data Analysis Example

R1-1.2.2 Consumers that collect trace data from the equipment need to be able to quickly make sense of incoming data in order to determine the proper analysis and/or limits to apply. In addition to the raw low level data values from sensors, actuators, and other devices of interest, applications need to be able to quickly determine the executing context associated with that data. The equipment needs to be able to provide sufficient context information to the application to allow it to determine at all times:

- What module is the data coming from?
- What recipe is that module running?
- What step is that recipe executing?
- What material is in that module?
- What job is responsible for that material?

R1-1.2.3 In addition to the information in Section R1-1.2.2 , typical consumers must also know the manufacturing execution context (for example, whether this an engineering lot, an experimental run, a characterization, a rework lot, a production run, a specific product, etc.) in order to properly analyze data from the equipment. While it is the consumer's responsibility to manage this manufacturing context data, it is the equipment's responsibility to provide the context information described in Section R1-1.2.2 .

R1-1.2.4 In many cases, this context data may consist of a non-trivial amount of descriptive information. It is typically not practical for all of this contextual data to be provided with each TraceReport sent by the equipment. Trace data may also be collected at rates up to, for example, 10Hz; much more frequently than the rate at which the job-related context information changes. Because of this situation, it is normally sufficient for the equipment to communicate context changes when they happen by defining events that are generated whenever any of the context described in Section R1-1.2.2 changes (see Figure R1-1).

R1-1.2.5 Along with such events, the equipment must also make available sufficient data with those events to completely describe the new context, as well as providing a way for consumers to query the current context at any time in an ad-hoc fashion. Additionally, some subset of this context data can be made available for inclusion with each TraceReport, such as the id's of the material at each location in the equipment.

R1-1.2.6 Some of this context data can be provided directly through the SEMI Data Collection Management and related specifications. Specifically, the Data Collection Management specification requires that all parameters, events, and exceptions have both a source and a name. The source can be used to specify the physical equipment component that produces the parameter, event, or exception. This information can be made known to consumers, for example, through the use of SEMI E120 (Common Equipment Model) and E125 (Equipment Self Description). In this way, consumers are at all times aware of the physical equipment component from which the data, event, or exception is coming.

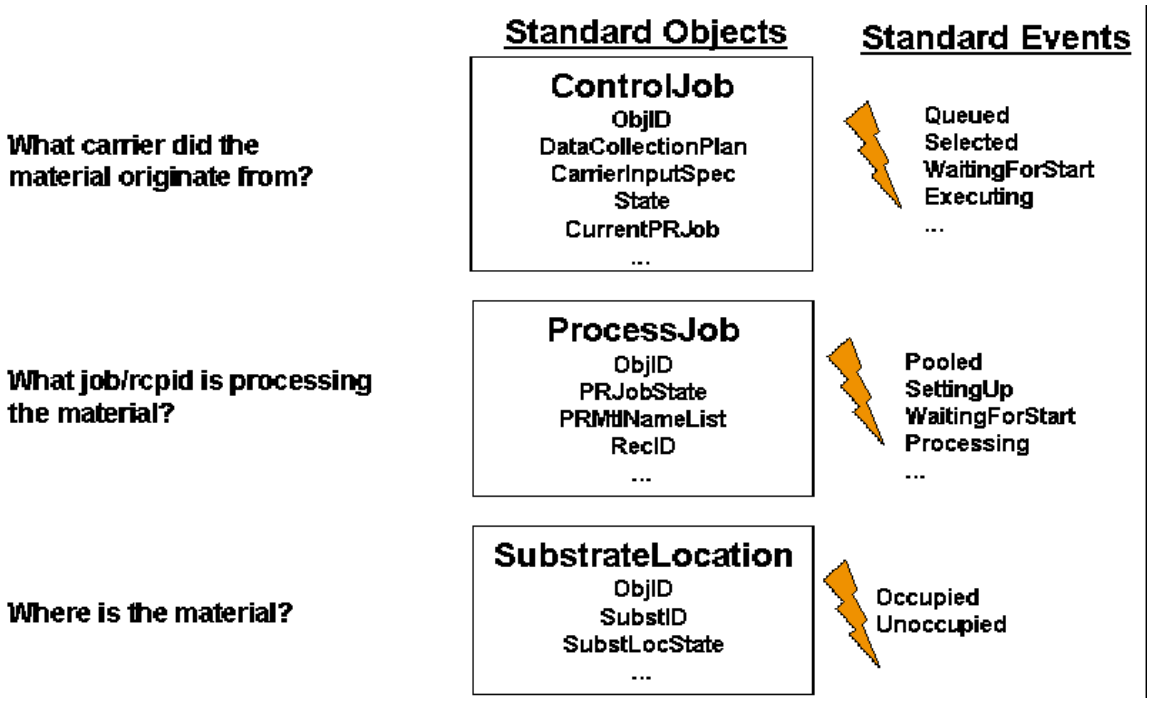

Figure R1-2
SEMI Standard Context Data

R1-1.2.7 Additional context data can be obtained from the SEMI 300mm standards (SEMI E40, SEMI E94, and SEMI E90 data are shown as examples in Figure R1-2). However, at the time of this writing, some context data is not available in standard form, yet is often essential for most analysis applications. For example, SEMI E40 does not provide a means for consumers to determine which sub-recipes might be executing on each module of the equipment, and does not provide a way for consumers to determine which recipe step is currently being executed on each sub-recipe. This information is essential for fault detection applications, which need to analyze and apply limits to process data that may change depending on which recipe step is being executed on which component of the equipment. Unless and until this information becomes standardized, equipment suppliers will need to provide this information to applications using proprietary events, exceptions and parameters.

R1-1.2.8 Putting the standards data together with equipment-specific data to cover the gaps, we can see how a consumer can take an incoming Data Collection Management TraceReport and determine much of the processing context. A typical example might involve the following high-level steps:

- Inspect the self-descriptive information available from the equipment via SEMI E125 to identify the low-level sensor/actuator and process Parameters from each SEMI E120 equipment component that are of significance to the purpose of the analysis application (diagnostic, fault detection, etc.)

- Create a DataCollectionPlan that includes at least one TraceRequest at the desired frequency. The TraceRequest should include a ParameterRequest for each Parameter of interest from the previous step.
- Inspect the E125 data from the equipment to determine the SEMI standard carrier management, control job, process job, and substrate tracking events, exceptions, and available Parameters that provide key material and processing status context.
- In the DataCollectionPlan created earlier, include an EventRequest and ExceptionRequest for each context event and exception of interest. For each EventRequest, include a ParameterRequest for each context data variable available for that event.
- Inspect the E125 data from the equipment to determine the supplier-defined events and data that communicate what sub-recipes and which steps are executing on which components of the equipment, and any other supplier-specific context data of significance to the purpose of the analysis application.
- In the DataCollectionPlan created earlier, include an EventRequest and ExceptionRequest for each context event and exception identified in the previous step. For each EventRequest, include a ParameterRequest for each context data variable available for that event.
- Submit the DCP to the equipment using the DataCollectionManager "DefinePlan" operation, and activate using the "ActivatePlan" operation to begin receiving trace data and context events/exceptions.

R1-1.2.9 Once such a DCP has been activated, consumers can begin tracking the location of each substrate on the equipment, so that their physical locations are always known. Since timestamps of all data are referenced to the tool, synchronization of the events and data can be correlated correctly.

R1-1.2.10 Events from the equipment that inform what recipes are executing on which modules, and which step is currently executing can be tracked and stored by the consumer, so it is always known which module is executing which step in which recipe on which substrate.

R1-1.2.11 By tracking Control and Process Job events and data, a consumer can always determine to which job a given substrate at a given location belongs.

R1-1.2.12 The consumer is responsible for managing the manufacturing context of each job, and so knows which analysis needs to be performed on data from material from those jobs.

R1-1.2.13 Finally, because the consumer knows what the contents of the DCP are, it is also known which physical equipment component produces each trace data variable. So, the data from a TraceReport can be correlated to a physical equipment component, for which the consumer already knows the material located at that component, the recipe executing on the component, the step the recipe is executing, the job the material belongs to, and the manufacturing context for that job. Provided all of this information is provided correctly and in a timely fashion, a consumer can always determine which analysis and/or control limits need to be applied to each TraceReport.

RELATED INFORMATION 2
UML TERMINOLOGY

NOTICE: This related information is not an official part of SEMI E125 and was derived from the work of the originating committee. This related information was approved for publication by full letter ballot procedures.

R2-1 Glossary

R2-1.1 In order to help with understanding terminology used in conjunction with object technology, this section shares excerpts from the glossary of the OMG standard for UML — the predominant modeling notation.

R2-1.2 These definitions are taken from B.2 Glossary of Terms of version 1.4 of the OMG UML specification, 01-09-67, available from **http://www.omg.org/technology/documents/modeling_spec_catalog.htm**.

R2-1.2.1 *abstract class* — a class that cannot be directly instantiated. Contrast: *concrete class*.

R2-1.2.2 *aggregate [class]* — a class that represents the "whole" in an aggregation (whole-part) relationship. See: *aggregation.*

R2-1.2.3 *aggregation* — a special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part. See: *composition*.

R2-1.2.4 *association* — the semantic relationship between two or more classifiers that specifies connections among their instances.

R2-1.2.5 *attribute* — a feature within a classifier that describes a range of values that instances of the classifier may hold.

R2-1.2.6 *cardinality* — the number of elements in a set. Contrast: *multiplicity*.

R2-1.2.7 *child* — in a generalization relationship, the specialization of another element, the parent. See: *subclass*, *subtype*. Contrast: *parent*.

R2-1.2.8 *class* — a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. See: *interface*.

R2-1.2.9 *class diagram* — a diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.

R2-1.2.10 *classification* — the assignment of an object to a classifier. See: *dynamic classification*, *multiple classification, static classification*.

R2-1.2.11 *classifier* — a mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, datatypes, and components.

R2-1.2.12 *composition* — a form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive. Synonym: *composite aggregation*.

R2-1.2.13 *concrete class* — a class that can be directly instantiated. Contrast: *abstract class.*

R2-1.2.14 *constraint* — a semantic condition or restriction. Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML. See: *tagged value, stereotype.*

R2-1.2.15 *dependency* — a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).

R2-1.2.16 *diagram* — a graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs (relationships) and vertices (other model elements). UML supports the following diagrams: class diagram, object diagram, use case diagram, sequence diagram, collaboration diagram, state diagram, activity diagram, component diagram, and deployment diagram.

R2-1.2.17 *element* — an atomic constituent of a model.

R2-1.2.18 *feature* — a property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a datatype.

R2-1.2.19 *generalization* — a taxonomic relationship between a more general element and a more specific element. The more specific element is fully consistent with the more general element and contains additional information. An instance of the more specific element may be used where the more general element is allowed. See: *inheritance.*

R2-1.2.20 *inheritance* — the mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior. See: *generalization*.

R2-1.2.21 *instance* — an entity that has unique identity, a set of operations that can be applied to it, and state that stores the effects of the operations. See: *object*.

R2-1.2.22 *interface* — a named set of operations that characterize the behavior of an element.

R2-1.2.23 *interface inheritance* — the inheritance of the interface of a more general element. Does not include inheritance of the implementation. Contrast: *implementation inheritance*.

R2-1.2.24 *message* — a specification of the conveyance of information from one instance to another, with the expectation that activity will ensue. A message may specify the raising of a signal or the call of an operation.

R2-1.2.25 *method* — the implementation of an operation. It specifies the algorithm or procedure associated with an operation.

R2-1.2.26 *model* [MOF] — an abstraction of a physical system with a certain purpose. See: *physical system*. Usage note: In the context of the MOF specification, which describes a meta-metamodel, for brevity the metametamodel is frequently referred to as simply the model.

R2-1.2.27 *multiple inheritance* — a semantic variation of generalization in which a type may have more than one supertype. Contrast: *single inheritance*.

R2-1.2.28 *multiplicity* — a specification of the range of allowable cardinalities that a set may assume. Multiplicity specifications may be given for roles within associations, parts within composites, repetitions, and other purposes. Essentially a multiplicity is a (possibly infinite) subset of the non-negative integers. Contrast: *cardinality*.

R2-1.2.29 *object* — an entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines. An object is an instance of a class. See: *class, instance*.

R2-1.2.30 *object diagram* — a diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a collaboration diagram. See: *class diagram, collaboration diagram.*

R2-1.2.31 *operation* — a service that can be requested from an object to effect behavior. An operation has a signature, which may restrict the actual parameters that are possible.

R2-1.2.32 *package* — a general purpose mechanism for organizing elements into groups. Packages may be nested within other packages.

R2-1.2.33 *parent* — in a generalization relationship, the generalization of another element, the child. See: *subclass*, *subtype*. Contrast: *child*.

R2-1.2.34 *qualifier* — an association attribute or tuple of attributes whose values partition the set of objects related to an object across an association.

R2-1.2.35 *relationship* — a semantic connection among model elements. Examples of relationships include associations and generalizations.

R2-1.2.36 *role* — the named specific behavior of an entity participating in a particular context. A role may be static (for example, an association end) or dynamic (for example, a collaboration role).

R2-1.2.37 *single inheritance* — a semantic variation of generalization in which a type may have only one supertype. Synonym: *multiple inheritance* [OMA]. Contrast: *multiple inheritance*.

R2-1.2.38 *subclass* — in a generalization relationship, the specialization of another class; the superclass. See: *generalization*. Contrast: *superclass*.

R2-1.2.39 *subtype* — in a generalization relationship, the specialization of another type; the supertype. See: *generalization*. Contrast: *supertype*.

R2-1.2.40 *superclass* — in a generalization relationship, the generalization of another class; the subclass. See: *generalization.* Contrast: *subclass*.

R2-1.2.41 *supertype* — in a generalization relationship, the generalization of another type; the subtype. See: *generalization*. Contrast: *subtype*.

R2-1.2.42 *type* — a stereotyped class that specifies a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects. A type may not contain any methods, maintain its own thread of control, or be nested. However, it may have attributes and associations. Although an object may have at most one implementation class, it may conform to multiple different types. See: *implementation class.* Contrast: *interface*.

R2-1.2.43 *visibility* — an enumeration whose value (public, protected, or private) denotes how the model element to which it refers may be seen outside its enclosing namespace.

SEMI E134.1-0305
PROVISIONAL SPECIFICATION FOR SOAP BINDING OF DATA COLLECTION MANAGEMENT (DCM)

This provisional specification was technically approved by the Global Information and Control Committee and is the direct responsibility of the North American Information and Control Committee. Current edition approved by the North American Regional Standards Committee on December 10, 2004. Initially available at www.semi.org February 2005; to be published March 2005.

Table of Contents

List of Figures

List of Tables

1 Purpose

1.1 *SOAP Implementation Mapping*

1.1.1 The purpose of this specification is to provide an implementation mapping of the SEMI E134 specification for Data Collection Management to the SOAP 1.1 protocol. This document provides a description of the XML Schema data types used to support the UML classes defined in SEMI E134, and also describes the WSDL port type and binding definitions used to support the operations defined by the interfaces specified in SEMI E134.

2 Scope

2.1 *Specification Scope*

2.1.1 The scope of this specification is the faithful representation of the DCM model in an XML Schema and corresponding WSDL port types and bindings. It will not add new domain information or concepts to the SEMI E134 model. The only additions made are those needed to render useful WSDL or XML Schema.

2.1.2 This specification requires SEMI E132 authentication, and defines requirements for the application of SEMI E132 concepts to the SEMI E134 specification.

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

3 Limitations

3.1 *Provisional Specification*

3.1.1 This specification is provisional pending the approval of the SEMI specifications Common Components (SEMI E138), SEMI E120.1, SEMI E125.1, and SEMI E132.1. This specification relies on XML Schema types defined by these specifications, and cannot be fully implemented without these types. Finalization of these specifications is a condition for the removal of the provisional status of this document.

3.1.2 This specification is provisional pending approval of the SEMI specification for Units for the Semiconductor Industry. This specification relies on the unit of measure symbols defined in that specification. Finalization of this specification is a condition for the removal of the provisional status of this document.

4 Referenced Standards

4.1 *SEMI Standards*

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

SEMI E120.1 — XML Schema for the Common Equipment Model

SEMI E132.1 — Specification for SOAP Implementation of Equipment Client Authentication and Authorization

SEMI E125.1 — Specification for SOAP Implementation of Equipment Self Description

SEMI E138 — XML Semiconductor Common Components

SEMI Specification for the Representation of Measurement Units in XML

4.2 *OMG Standards*[1]

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

4.3 *W3C Standards*[2]

1 Object Management Group, Inc., 250 First Ave. Suite 100, Needham, MA 02494, U.S.A., Phone: 781.444.0404, Fax:781.444.0320, website: www.omg.org.

2 World Wide Web Consortium, Massachusetts Institute of Technology (MIT), Computer Science and Artificial Intelligence Laboratory (CSAIL), 32 Vassar Street Room 32-G515, Cambridge, MA 02139, USA, Telephone: 617.253.2613, Fax: 617.258.5999, website: www.w3.org.

Extensible Markup Language (XML) 1.0 (Second Edition) — W3C, 6 October 2000 (http://www.w3.org/TR/2000/REC-xml-20001006/).

Namespaces in XML — W3C, 14 January 1999 (http://www.w3.org/TR/1999/REC-xml-names-19990114/).

XML Schema Part 0: Primer — W3C, 2 May 2001 (http:// www.w3.org /TR/xmlschema-0/).

XML Schema Part 1: Structures — W3C, 2 May 2001 (http:// www.w3.org /TR/xmlschema-1/).

XML Schema Part 2: Datatypes – W3C, 2 May 2001 (http:// www.w3.org /TR/xmlschema-2/).

XML Path Language (Xpath) — W3C, 16 November 1999 (http:// www.w3.org */TR/xpath/).*

Web Services Description Language (WSDL) 1.1 — W3C Note, (http://www.w3.org/TR/wsdl)

Simple Object Access Protocol (SOAP) 1.1 — W3C Note, (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

4.4 *Other Standards*

Secure Sockets Layer (SSL) (available from *http://wp.netscape.com/eng/ssl3/draft302.txt*)

Web Services Interoperability (WS-I): Basic Profile Version 1.0a (available from *http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html*)

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

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *SOAP* — Simple Object Access Protocol

5.1.2 *UML* — Unified Modeling Language

5.1.3 *W3C* — World Wide Web Consortium

5.1.4 *WSDL* — Web Service Description Language

5.1.5 *XML* — eXtensible Markup Language

5.2 *Definitions*

5.2.1 *UML (Unified Modeling Language)* — a notation for representing object-oriented designs and views created by Booch, Rumbaugh, and Jacobson in order to merge their three popular notations plus aspects of other existing notations into a single object-oriented notation intended to be usable by all.

5.2.2 *XML (eXtensible Markup Language)* — a markup language used for representing data rich with context and content in documents and in communications. XML is an extension of SGML, a document-oriented markup language. It was created by W3C for use on the Internet.

6 Conventions

6.1 *Translating UML to XML Schema*

6.1.1 The reader is expected to have a working knowledge of the UML, XML, and Schema specifications (See ¶¶0and 4.3). This document does not provide tutorial information on these subjects.

6.1.2 This document follows the guidelines for XML as outlined in SEMI E121.

6.1.3 Some of the key guidelines followed in this document are summarized here:

- Attributes of UML classes from SEMI E125 are generally represented as XML attributes, to improve efficiency in transferring metadata descriptions from the equipment. Exceptions to this convention are UML attributes that correspond to data structures or other constructs that cannot be represented as XML attributes, or may include text values that may need to include non-parsed character data (CDATA).
- Composition associations are mapped to a single XML element (or multiple such elements if the specified multiplicity is greater than 1).

- Aggregations are modeled as contained elements or arrays of references to one or more unique attributes of the target type.
- Inheritance is mapped to using complexType extension. In cases where a reference to an abstract base class occurs in the UML model, a choice compositor for each of the possible derived types is used in order to restrict instance documents from including arbitrary extension through type substitution.

6.1.4 The translation of a UML class to XML Schema types is documented using a table format illustrated by Table 1. Operations defined for a UML class are translated into WSDL operations.

6.1.5 *Translation Table Column Header Description*

- *Attribute or Role Name* — If an attribute, the name of the attribute is placed here. If an association (including aggregation or composition), the role name from the UML diagram is placed here. Compositions are often not assigned role names. In that case "none" is placed here.
- *UML Name/Type* — If an attribute, the data type of the UML attribute is place here. If an association, the type of association is placed here. The possible association types are "Composition", "Aggregation", or the basic "Association". UML defines these three types of association.
- *XML Element or Attribute* — Lists the type of XML construct used to represent the UML attribute or association.
- *XML Name/Type* — Provides the name and data type of the resulting XML construct. The type may be a built-in type (for example, xs:string), or a named type defined within the XML Schema.

Table 1 Example Translation Table

Attribute or Role Name	*UML Name/Type*	*XML Element or Attribute*	*XML Name/Type*
friend	association	element	Friend: HumanReferenceArray
employees	aggregation	element	Employees: HumanArray
family	composition	element	Family: HumanArray
name	string	element	Name: xs:string

6.1.6 *Translating Operations to XML Schema*

6.1.6.1 The translation of an operation defined for UML interface classes to XML Schema types is documented using a table format illustrated in Table 2. One row is provided for each possible input/output argument for the operation being described.

Table 2 Example Translation Table for Operation Input/Output Arguments

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
<argument name from abstract specification>	<argument format from abstract specification>	<whether the argument is modeled as an Element or Attribute>	<namespace-qualified name of the attribute/element>

6.2 *Documenting XML Schema and WSDL Files*

6.2.1 Associated with this document are XML Schema and WSDL files that are Common Components of the specification. Each such document is described using a table format illustrated by Table 3.

Table 3 Example Schema/WSDL Document Description Table

File Name	<.xsd or .wsdl file name>
Target Namespace	<target namespace defined in the schema or WSDL file>
Imported/Referenced Namespaces	<namespaces imported or used by the schema or WSDL file>
Description	<brief description of the purpose/function of the schema or WSDL file>

6.3 *Documenting XML Schema With Diagrams*

6.3.1 This document provides graphical representations of the included XML Schema data type and element definitions. Although no standard graphical notation for XML Schema could be found, various XML tools have their own notation. This document will use the notation provided by XMLSpy from Altova Corporation Table 1 shows a sample XML Schema diagram that will be used to provide a basis for explanation of the schema graphical notation used in the rest of the document.

6.3.2 In the diagram, rectangular boxes represent XML element definitions. Ownership or containment is read from left to right in the diagrams. In the sample diagram, ParentType contains Child1, Child2, Child 3, and Child 4. In turn, Child2 contains Child2a, Child2b, and Child2c. The additional symbols (8-sided boxes) represent sequences or choices. See Table 4 for an explanation of these symbols.

Table 4 Altova XMLSPY Schema Diagram Symbols

	Denotes a required ordered sequence of the right hand elements with a cardinality of one for each element. (sequence)
	Denotes an optional ordered sequence of the right hand elements with a cardinality of one for each element. (sequence)
	Denotes a required, but unordered, sequence of the right hand elements with a cardinality of one for each element. (all)
	Denotes a required choice of the right hand elements. Exactly one or two of the right hand elements must be present. (choice)
	Denotes an element that contains parsed character data
	Denotes a reference to a group or element defined elsewhere

6.3.3 A graphic using a solid line is a required element; using a dashed line represents an optional element. Numbers or ranges in the lower right hand corner represent cardinality. The default cardinality is one.

6.3.4 To simplify a diagram or help focus on a particular aspect, detail may be hidden. The 8-sided symbols have a small square on the right end. If a minus sign "-" is in the box, then all detail is shown. If the box contains a plus sign "+", then all detail to the right of that symbol is hidden. The example has no hidden detail.

6.3.5 The yellow (or grey if printed in monochrome) boxes indicate the use of other defined types. So, Child4 is of type "Child4Type". Child4Type defines Child4a and Child4b. This detail may be hidden in the diagram. Object oriented inheritance is typically represented in XML as type extension. In Figure 1, ParentType extends Child1and2Type by adding a sequence that includes Child3 and Child4.

6.3.6 Reading Figure 1 would yield the following additional information:

1. Child1and2Type is an ordered sequence of two items: Child2 and Child1.
2. Child1 contains an optional ordered sequence of Child1a, one or more Child1b, and (optionally) Child1c.
3. Child2 contains a choice of one or two of the following: Child2a, Child2b, and Child2c.
4. Child3 contains an unordered sequence of Child3a and Child3b.

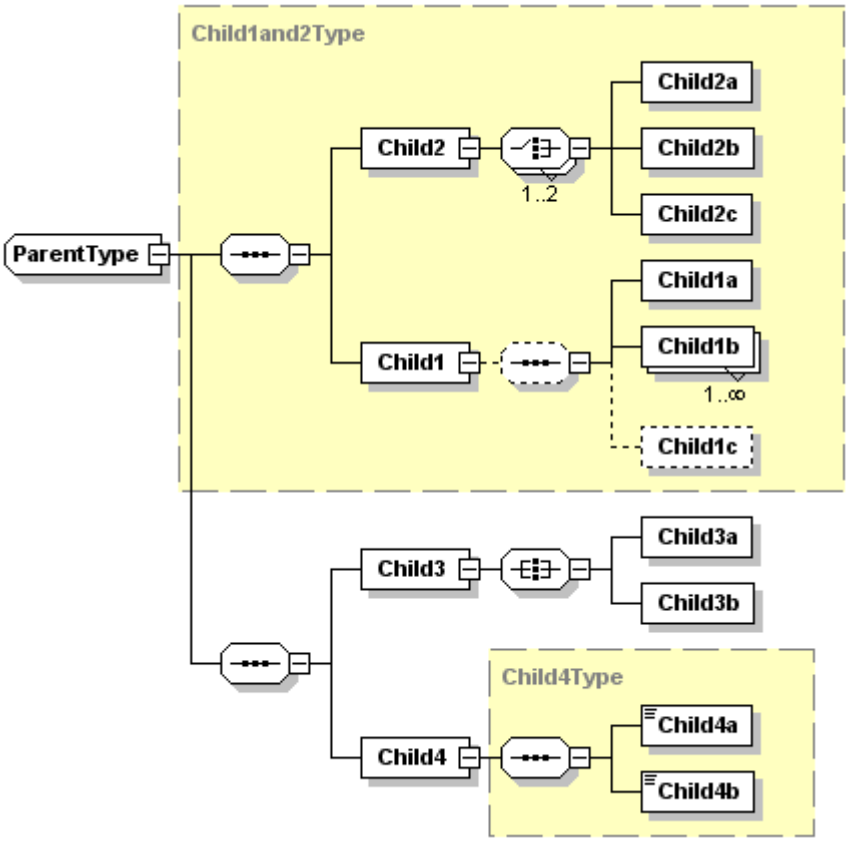

Figure 1
XML Schema Example Diagram

6.4 *XML Schema Sample*

6.4.1 The sample XML Schema for the example shown in Figure 1, is presented below. Refer to the XML documentation referenced in ¶4.3 for a complete description of the syntax and semantics of XML Schema.

```
<xs:element name="Parent">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Child1"/>
            <xs:complexType>
               <xs:sequence minOccurs="0">
                  <xs:element name="Child1a"/>
                  <xs:element name="Child1b" maxOccurs="unbounded"/>
                  <xs:element name="Child1c" minOccurs="0"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="Child2">
            <xs:complexType>
               <xs:choice maxOccurs="2">
                  <xs:element name="Child2a"/>
                  <xs:element name="Child2b"/>
                  <xs:element name="Child2c"/>
               </xs:choice>
            </xs:complexType>
         </xs:element>
         <xs:element name="Child3">
            <xs:complexType>
               <xs:all>
                  <xs:element name="Child3a"/>
                  <xs:element name="Child3b"/>
               </xs:all>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Figure 2
XML for Sample

6.5 *Translating UML to WSDL*

6.5.1 In general, UML interface classes defined in the abstract specification are translated into WSDL as portType definitions, and each portType definition has a corresponding WSDL binding definition. The following tables show the convention used for documenting the WSDL port type and binding definitions for a given UML interface class.

Table 5 Example Interface WSDL Port Type Table

Class Name	<UML interface name from abstract specification>
WSDL Port Type Name	<port type name used in WSDL>
SEMI E125 Operation → WSDL Operation	<operation name from UML interface> → <WSDL port type operation name>

Table 6 Example Interface WSDL Binding Table

SEMI E125 Class Name	<UML interface name from abstract specification>
WSDL Binding Name	<binding name used in WSDL>
SOAP Binding Style	<RPC or document>
SOAP Transport	<transport identifying URI>

6.5.2 Each operation defined for a given UML interface class has a corresponding operation definition and binding. Operations are described using a table format illustrated by Table 7 and Table 8.

Table 7 Example Operation Binding Table

SOAPAction	<the SOAPAction HTTP header value to be used for this operation>
Input Headers (WSDL Message, Required)	<the name of the WSDL message providing input headers, and whether or not the headers are required>
Output Headers (WSDL Message, Required)	<the name of the WSDL message providing output headers, and whether or not the headers are required>

Table 8 Example PortType Operation Table

Input Message Name	<WSDL message name used as input for the WSDL port type operation>
Output Message Name	<WSDL message name used as output for the WSDL port type operation>

7 Mapping of E134 UML to XML Schema and WSDL

7.1 *WSDL Organization*

7.1.1 Each interface definition in SEMI E134 is mapped to a WSDL portType and binding definition. WSDL portType definitions are named after the interface and its operations as they appear in SEMI E134. WSDL binding definitions for each portType are used to specify the SOAP 1.1 envelope contents for each operation, and to define the corresponding XML encoding styles and HTTP header usage. All SEMI E134 WSDL interfaces use document/literal encoding, with the complete SOAP header and body contents defined in XML Schema file(s) via global element definitions.

7.1.2 Figure 3 shows the relationship between the WSDL binding and portType definitions and the XML Schema types used for each interface. The portType definition imports XML Schema type definitions used in each operation via the XML Schema "import" statement. The WSDL binding definition imports the portType definition via the WSDL "import" statement. Figure 4 shows the specific XML Schema and WSDL files defined for SEMI E134.

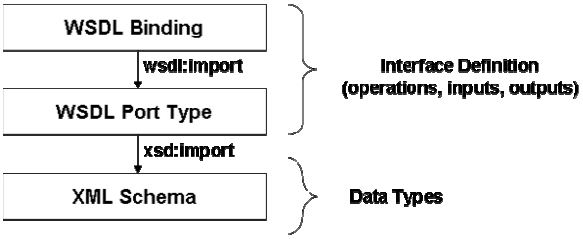

Figure 3
XML Schema and WSDL File Organization

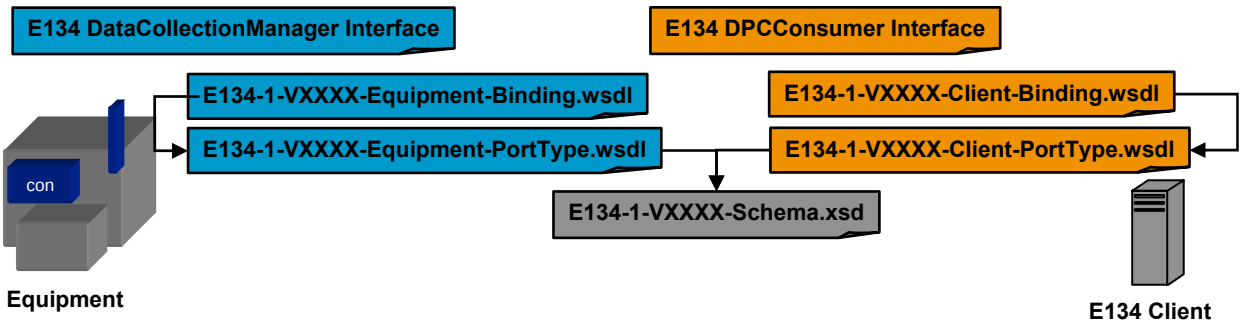

Figure 4
WSDL and XML Schema Files for SEMI E134

7.2 *DataCollectionManager*

7.2.1 *Application of SEMI E132 Sessions*

7.2.1.1 This section is provisional, pending the approval of SEMI E132.1 and the SEMI Specification for Semiconductor Common Components.

7.2.1.2 The DataCollectionManager shall use SEMI E132.1 for authentication and authorization.

7.2.1.3 Use of the interfaces defined in SEMI E134 to collect data can directly affect overall data throughput for the equipment, and may indirectly affect the equipment's ability to perform its intended function. Additionally, data accessible from the equipment via the SEMI E134 interfaces may be proprietary to the equipment supplier and/or the factory that uses the equipment, and should be protected against unauthorized access within the factory.

7.2.1.4 For this reason, all operations defined by the SEMI E134 DataCollectionManager interface can be exchanged using the HTTPS transport (HTTP 1.1 over SSL 3.0) by enabling SSL authentication as specified in SEMI E132.1. Once enabled via SEMI E132.1, the equipment shall require mutual client-server SSL authentication for all SEMI E134 DataCollectionManager operations supported by the equipment, and the equipment shall not provide access to any operations except via the HTTPS transport.

7.2.2 There may be some integration, development, and/or test environments in which it is not practical or feasible to install equipment and/or client certificates in order to communicate with the equipment. As a practical consideration for such environments only, the equipment shall support the ability of the user to disable SSL authentication for the SEMI E134 DataCollectionManager interface, using the equipment configuration specified by E132.1. Users should be aware that disabling SSL authentication opens up complete unrestricted access to SEMI E134 data for any application with network access to the equipment.

7.2.2.1 Regardless of whether or not SSL authentication is enabled, the equipment shall reject all requests from clients that do not provide a valid SEMI E132 session identifier in the SOAP envelope header provided with the request. The SEMI E132 session identifier shall be provided via the E132Header element, as specified by SEMI E132.1. If a client does not provide a recognized session identifier, the equipment shall immediately reject the request using the SEMI Common Error data type, with an error code of '6005' and a source of 'urn:semi-org:E132' as specified in SEMI E132.1. The equipment shall perform no other processing of an SEMI E134 request if the request is found to include an invalid SEMI E132 session identifier, or if no session identifier is provided.

7.2.3 *Session and DCP State Interaction*

7.2.3.1 The equipment shall maintain DCP activity on a per-Session basis. For example, the same SEMI E132 Principal may establish one or more SEMI E132 Sessions on the equipment. If only one of those Sessions activates a DCP, the equipment shall only enable the DCP for that specific session. If two separate SEMI E132 sessions attempt to activate the same DCP, the equipment shall activate the DCP for both sessions. Similarly if one active SEMI E132 session attempts to activate the same DCP twice, the equipment shall respond to the second request with the DCPIsActive error, as defined in SEMI E134.

7.2.3.2 As described in ¶7.2.1.3 , the equipment shall use the URL provided with the HTTPEndpoint information provided with the SEMI E132 EstablishSession request to send all SEMI E134 communication.

7.2.3.3 If a consumer's SEMI E132.1 Session is closed for any reason (either expressly closed or through a communication failure), the equipment shall stop sending data from all DCP's previously activated by that consumer. This shall not affect data transmission from those DCP's to consumers that still have SEMI E132 Sessions in the Established state.

7.2.3.3.1 If one or more of these DCP's has more than one consumer, the equipment shall take the actions defined in transition 5 of the E134 DataCollectionPlan State model. These DCP's will remain active, and the equipment shall remove the consumer's session from the active consumers list and continue to send data to the other consumers for those DCP's.

7.2.3.3.2 If one or more of these DCP's has only one consumer, the equipment shall take the actions defined in transition 8 of the SEMI E134 DataCollectionPlan State model. The equipment shall remove the consumer's session from the active consumers list and change the state of these DCP's to Inactive.

7.2.3.4 If a DCP is returning from the Hibernating state and there is at least one consumer with an SEMI E132 persistent Session that had previously requested activation of the DCP, then the equipment shall transition the DCP to the Active state, and resume sending data to each consumer with an SEMI E132 persistent Session.

7.2.3.5 If a DCP is returning from the Hibernating state and there are no consumers with SEMI E132 persistent Sessions, then the equipment shall transition the DCP to the Inactive state.

7.2.4 Authentication and Authorization Using SEMI E132.1

7.2.4.1 The equipment shall implement the SEMI E134 privilege levels as SEMI E132 privileges.

7.2.4.2 The equipment shall implement the SEMI E134 privilege levels naming convention as specified in Table 9. The privileges listed are of the format urn:semi-org:priv.<priv-name>.

Table 9 SEMI E134 to SEMI E132 Privilege mapping

SEMI E134 Privilege Level	*SEMI E132.1 PrivilegeId Entry*
ManageOnlyAuthoredDCPs	urn:semi-org:priv.ManageOnlyAuthoredDCPs
UseAnyDCP	urn:semi-org:priv.UseAnyDCP
ManageAnyDCP	urn:semi-org:priv.ManageAnyDCP

7.2.5 *Using SEMI E125 Metadata to Identify Sources, Events, Exceptions, and Parameters*

7.2.5.1 Sources shall be named using a valid SEMI E120 Locator value (see SEMI E120, Related Information 2, and the SEMI E120.1 LocatorType). Events/exceptions are identified by their SEMI E125 “id” attribute, and Parameters are identified by their SEMI E125 “name” attribute. If the events, exceptions, or parameters are defined by an SEMI E39-compliant objType, then use the following naming conventions to distinguish between information from any instance vs. a specific instance.

Table 10 Source/Event/Exception/Parameter Naming Convention

SEMI E125 Element	*Naming Convention*
Parameter	SEMI E125 Parameter Class Name attribute
Event	SEMI E125 Event Class Id attribute
Exception	SEMI E125 Exception class Id attribute
ObjType Attribute (any instance)	urn:semi-org:objType:<typeName>:objAttr:<parameter name>
ObjType Event (any instance)	urn:semi-org:objType:<typeName>:<eventId >
ObjType Exception (any instance)	urn:semi-org:objType:<typeName>:<exceptionId>
ObjType Attribute (specific instance)	urn:semi-org:objType:<instName>:objAttr:<parameterName >
ObjType Event (specific instance)	urn:semi-org:objType:<instName>:<eventId >
ObjType Exception (specific instance)	urn:semi-org:objType:<instName>:<exceptionId>

7.2.5.2 <typeName> and <instName> referenced in Table 10 shall be of the following format. “typeName” shall be the E39 objType name for the object type of interest. “instName” shall be equal to a string of the form <typeName>-<E39 objId>, where “typeName” specifies the object type of interest, and “objId” specifies the particular object of interest, equal to the value of that instance’s SEMI E39 objId attribute.

7.2.6 *Use of Common Error Type*

7.2.6.1 This section is provisional, pending the approval of the SEMI Specification for Semiconductor Common Components.

7.2.6.2 Table 11 below provides a mapping of the SEMI E134 defined errors to its common error code. The SEMI E134 mapped error codes below are in addition to any common error codes defined in the Common Components Specification. SEMI E134 Error codes shall be denoted in the common error type structure by defining the “source” name of “urn:semi-org:E134”.

7.2.6.3 SEMI E134.1 implements the SEMI E134 Unauthorized Operation Class as the SEMI E132.1 UnauthorizedOperation class and will return the SEMI E132.1 error code “6005” and a source of “urn:semi-org:E132” Operation not authorized in the Common Components Error structure.

Table 11 SEMI E134 Error Codes

SEMI E134 Error	*Common Error Code*
Operation not authorized.	6005 (as defined in SEMI E132.1)
InvalidPlan	8000
NoSuchPlan.	8001
DCPIsActive.	8002
DCPNotActive.	8003

7.2.7 *XML Schema and WSDL Files*

7.2.7.1 This section is provisional, pending the approval of SEMI E120.1, SEMI E125.1, and SEMI E132.1.

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

Table 12 XML Schema

File Name	E134-1-V0305-Schema.xsd
Target Namespace	urn:semi-org:xsd.E134-1.V0305.DCM
Imported/Referenced Namespaces	urn:semi-org:xsd.E120-1.V0704.CommonEquipmentModel http://www.w3.org/2001/XMLSchema
Description	This file defines all of the SEMI E134 data types and elements used by both the DataCollectionManager and DCPConsumer interfaces.

Table 13 DataCollectionManager PortType Definitions

File Name	E134-1-V0305-Equipment-portType.wsdl
Target Namespace	urn:semi-org:ws.E134-1.V0305.DCMEqp-portType
Imported/Referenced Namespaces	urn:semi-org:xsd.E132-1.V0305.auth urn:semi-org:xsd.E134-1.V0305.DCM 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 DataCollectionManager interface, based on the data types defined in the SEMI E134 XML Schema.

Table 14 DataCollectionManager Binding Definitions

File Name	E134-1-V0305-Equipment-binding.wsdl
Target Namespace	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding
Imported/Referenced Namespaces	urn:semi-org:ws.E134-1.V0305.DCMEqp-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.

7.2.7.3 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.

7.2.7.4 E134-1-V0305-Schema.xsd imports types from the SEMI E132.1 Equipment Client Authentication and Authorization XML Schema namespace, specifically the E132Header and E132HashHeader types.

7.2.7.5 E134-1-V0305-Schema.xsd imports types from the SEMI 3570.1 Specification for XML Semiconductor Common Components XML Schema namespace, specifically the ErrorType type.

7.2.8 WSDL Port Type Overview

7.2.8.1 The DataCollectionManager WSDL portType definition organizes the SEMI E134.1 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 DataCollectionManager interface in SEMI E134, ¶9.1. The WSDL portType itself is named after the SEMI E134 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 E134. Figure 5 illustrates the relationship between the SEMI E134 UML interface and the WSDL portType definition. The WSDL message and operation definitions are described in ¶¶7.2.10 through 7.2.20.

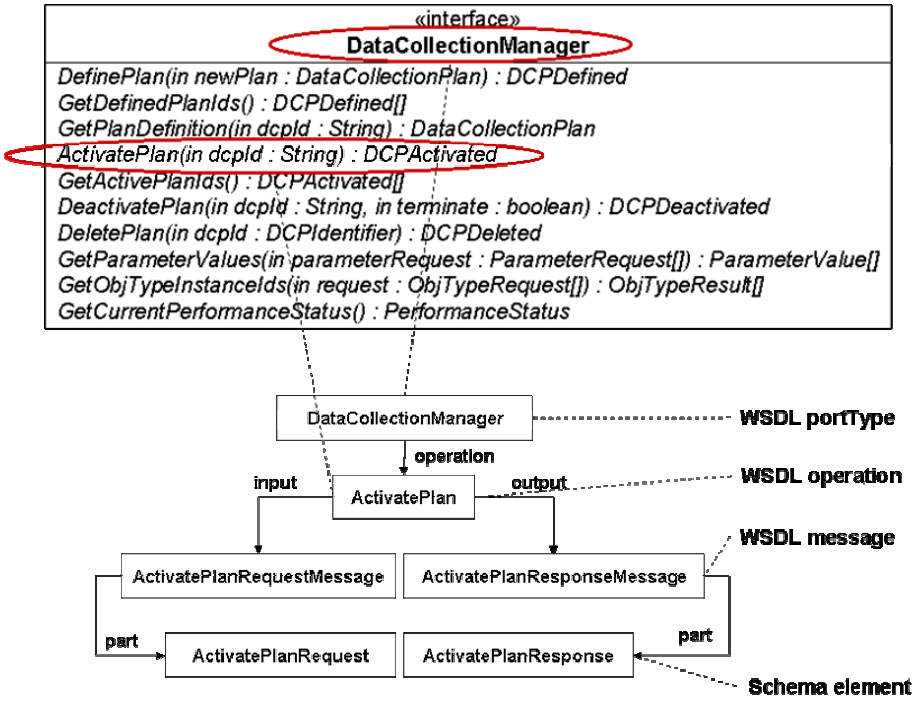

Figure 5
Mapping the DataCollectionManager to a WSDL portType

7.2.9 *WSDL Binding Overview*

7.2.9.1 The WSDL DataCollectionManager binding definition specifies a message and transport protocol to use for a given portType (SOAP and HTTP for SEMI E134.1), the interface style used (document style for SEMI E134.1). 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). The binding definitions for each portType operation are described in ¶¶7.2.10 through 7.2.20.

7.2.10 *WSDL Service Overview*

7.2.10.1 This specification does not provide a WSDL service definition. WSDL service definitions define the address(es) 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 SEMI E134 equipment services in the factory

7.2.11 *DefinePlan*

7.2.11.1 *WSDL Operation Binding*

7.2.11.1.1 See the operation binding for "DefinePlan" in SEMI E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 15 DefinePlan Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:DefinePlan
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.11.2 *WSDL Operation*

7.2.11.2.1 See the portType operation definition for "DefinePlan" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 16 DefinePlan PortType Operation

Input Message Name	DefinePlanRequest
Output Message Name	DefinePlanResponse

7.2.11.3 *WSDL Message(s)*

7.2.11.3.1 See the message definitions for "E132HeaderMessage", "DefinePlanRequest", and "DefinePlanResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 17 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header DefinePlanRequest → dcm:DefinePlanRequestType DefinePlanResponse → dcm:DefinePlanResponseType

7.2.11.4 *XML Schema Types*

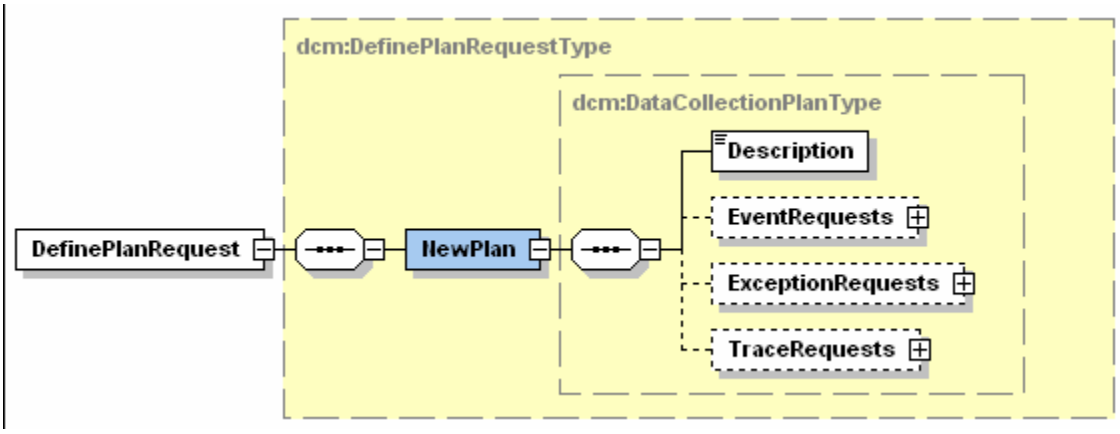

Figure 6
DefinePlanRequest

7.2.11.4.1 DefinePlanRequest - this global element acts as a top-level container entity representing the SOAP body of a DefinePlan request. As defined in SEMI E134, this operation has one input argument: the new DCP that is being defined. This argument is modeled as a single element named "NewPlan" of complexType "DataCollectionPlanType".

Table 18 Translation Table for Input Arguments for the DefinePlan Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
newPlan	Structured data, of type DataCollectionPlan	element	NewPlan: dcm:DataCollectionPlanType

7.2.11.4.2 *DataCollectionPlanType* — This global complexType is the XML Schema representation of the SEMI E134 Data Collection Plan class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 19 Translation Table for DataCollectionPlanType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
name	Text	attribute	name: xsd:string
id	Text	attribute	id: xsd:string
intervalInMinutes	Positive Integer >=0	attribute	intervalInMinutes: xsd:int
isPersistent	Boolean	attribute	isPersistent: xsd:boolean
description	Text	Element	Description: xsd:string
eventRequests	List of structured data, of type EventRequest	Element	EventRequest: dcm:EventRequestType
exceptionRequests	List of structured data of type ExceptionRequest	Element	ExceptionRequests: dcm:ExceptionRequestType
traceRequests	List of structured data of type TraceRequest	Element	TraceRequests: dcm:TraceRequestType

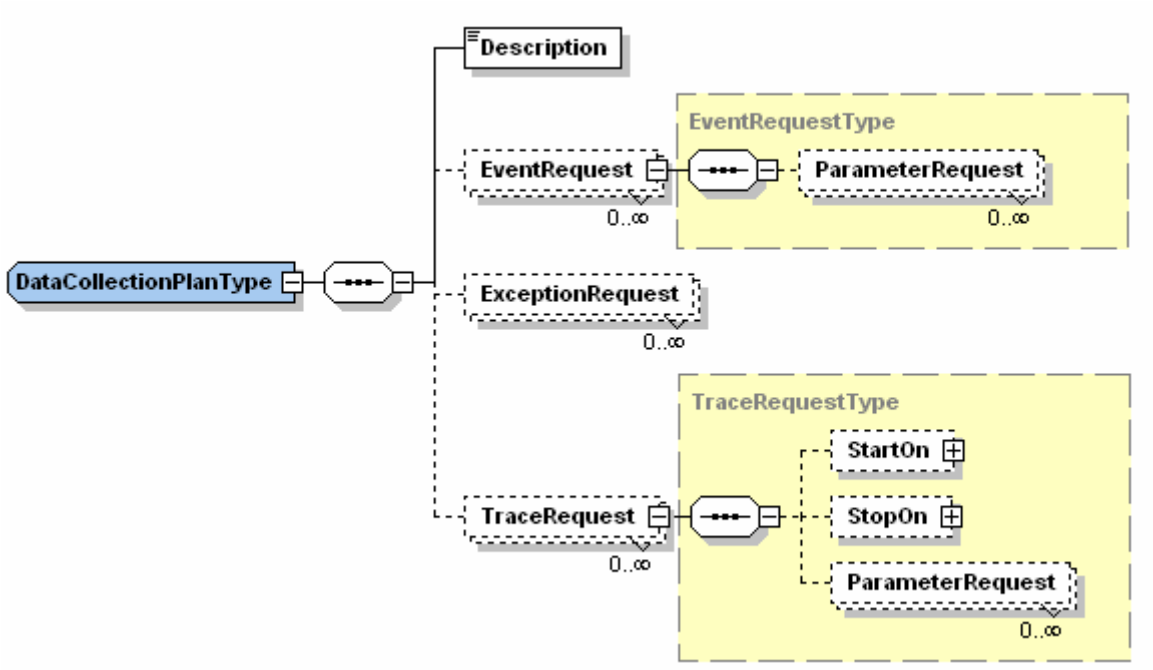

Figure 7
DataCollectionPlanType

7.2.11.4.3 *EventRequestType* — This global complexType is the XML Schema representation of the SEMI E134 Event Request class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 20 Translation Table for EventRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
eventId	Text	Attribute	eventId: xsd:string
parameterRequests	Ordered list of structured data, of type ParameterRequest	Element	ParameterRequests: dcm:ParameterRequest

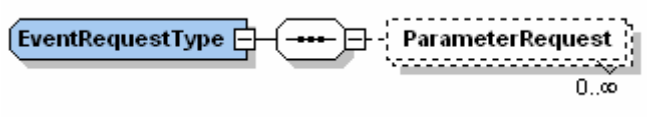

Figure 8
EventRequestType

7.2.11.4.4 *ParameterRequestType* – This global complexType is the XML Schema representation of the SEMI E134 Event Request class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 21 Translation Table for ParameterRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	xsd:string
parameterName	Text	Attribute	xsd:string

7.2.11.4.5 *ExceptionRequestType* — This global complexType is the XML Schema representation of the SEMI E134 Exception Request class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 22 Translation Table for ExceptionRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
exceptionId	Text	Attribute	exceptionId: xsd:string
severity	Text	Attribute	severity: xsd:string

7.2.11.4.6 *TraceRequestType* — This global complexType is the XML Schema representation of the SEMI E134 Trace Request class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 23 Translation Table for TraceRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
id	Integer	Attribute	id: xsd:int
intervalInSeconds	Real number.	Attribute	intervalInSeconds: xsd:float
collectionCount	Positive Integer >=0	Attribute	collectionCount: xsd:int
groupSize	Integer >=0	Attribute	groupSize: xsd:int
isCyclical	Boolean	Attribute	isCyclical: xsd:boolean
startOn	List of structured data of any type subclassed from Trigger	Element	StartOn: dcm:TriggerSelectorType
stopOn	List of structured data of any type subclassed from Trigger	Element	StopOn: dcm:TriggerSelectorType
parameterRequests	Ordered list of structured data, of type ParameterRequest	Element	ParameterRequests: dcm:ParameterRequestType

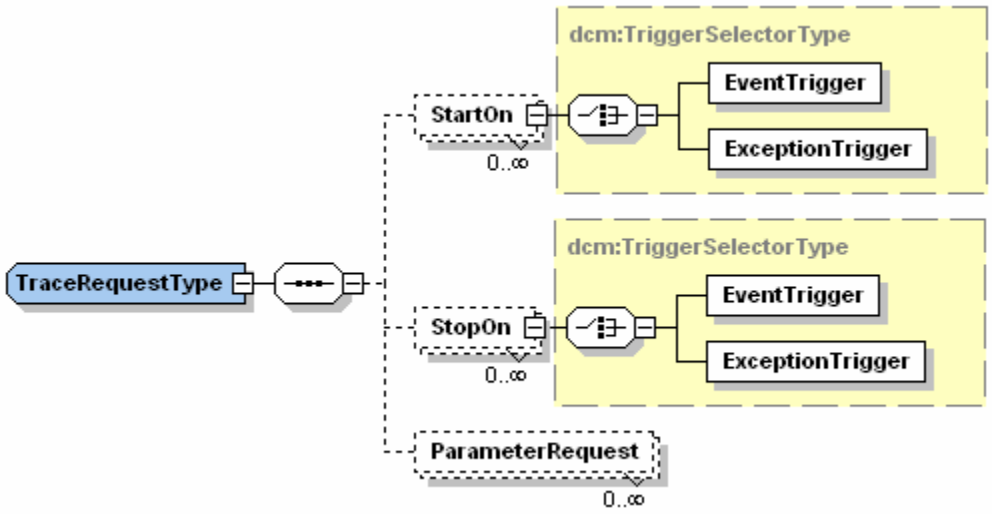

Figure 9
TraceRequestType

7.2.11.4.7 *EventTriggerType* — This global complexType is the XML Schema representation of the SEMI E134 Event Trigger class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 24 Translation Table for EventTriggerType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
eventId	Text	Attribute	eventId: xsd:string

7.2.11.4.8 *ExceptionTriggerType* — This global complexType is the XML Schema representation of the SEMI E134 Exception Trigger class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 25 Translation Table for ExceptionTriggerType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
exceptionId	Text	Attribute	exceptionId: xsd:string
exceptionState	Text.	Attribute	exceptionState: xsd:string

7.2.11.4.9 *ParameterRequestType* — This global complexType is the XML Schema representation of the SEMI E134 Parameter Request class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 26 Translation Table for ParameterRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
parameterName	Text	Attribute	parameterName: xsd:string

7.2.11.4.10 *DefinePlanResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 DefinePlan request. It contains either an element named “PlanDefined” of complexType “DCPDefinedType” or an element named “Error” of complexType “DefinePlanErrorType”. DefinedPlanErrorType contains the Common Components Error class and a choice of either an UnauthorizedOperationError or InvalidPlanError as decribed in the following table.

Table 27 Translation Table for Output Arguments for the DefinePlan Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
planDefined	Structured data, of type DCPDefined	Element	PlanDefined: dcm:DCPDefinedType
error	N/A	Element	ErrorType: ccs:Error
error	Structured data, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType
error	Structured data, of type InvalidPlan	Element	InvalidPlanError: dcm:InvalidPlanType

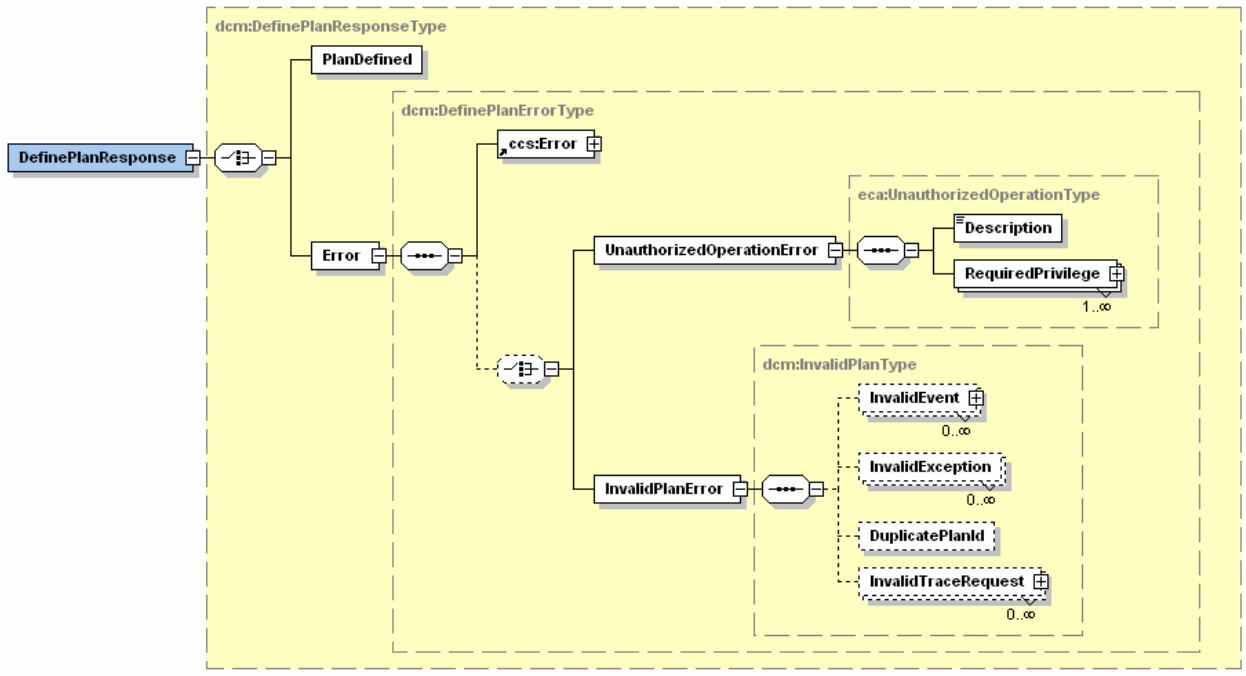

Figure 10
DefinePlanResponseType

7.2.11.4.11 *DCPDefinedType* — This global complexType is the XML Schema representation of the SEMI E134 DCPDefined class. The following table describes the mapping of the UML class to this XML Schema complexType

Table 28 Translation Table for DCPDefinedType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text.	Attribute	planId: xsd:string
timeDefined	Text	Attribute	timeDefined: xsd:datetime
definedBy	Text	Attribute	definedBy: xsd:string

7.2.11.4.12 *UnauthorizedOperationType* — UnauthorizedOperationType is defined and described in SEMI E132.1. Please reference E132.1 for the description of the UnathorizedOperationType.

7.2.11.4.13 *InvalidPlanType* — This global complexType is the XML Schema representation of the SEMI E134 InvalidPlan class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 29 Translation Table for InvalidPlanType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text	Attribute	planId: xsd:string
description	Text	Attribute	description: xsd:string
invalidEvents	Unordered list of elements of type InvalidEventRequest	Element	InvalidEvents: dcm:InvalidEventsRequestType
invalidExceptions	Unordered list of elements of type InvalidExceptionRequest	Element	InvalidExceptions: dcm:InvalidExceptionsRequestType
duplicatePlanId	Structure, of type DCPDefined	Element	DuplicatePlanId: dcm:DCPDefinedType reference ¶7.2.11.4.11
invalidTraceRequests	Unordered list of elements of type InvalidTraceRequest	Element	InvalidTraceRequests: dcm:InvalidTraceRequestsType

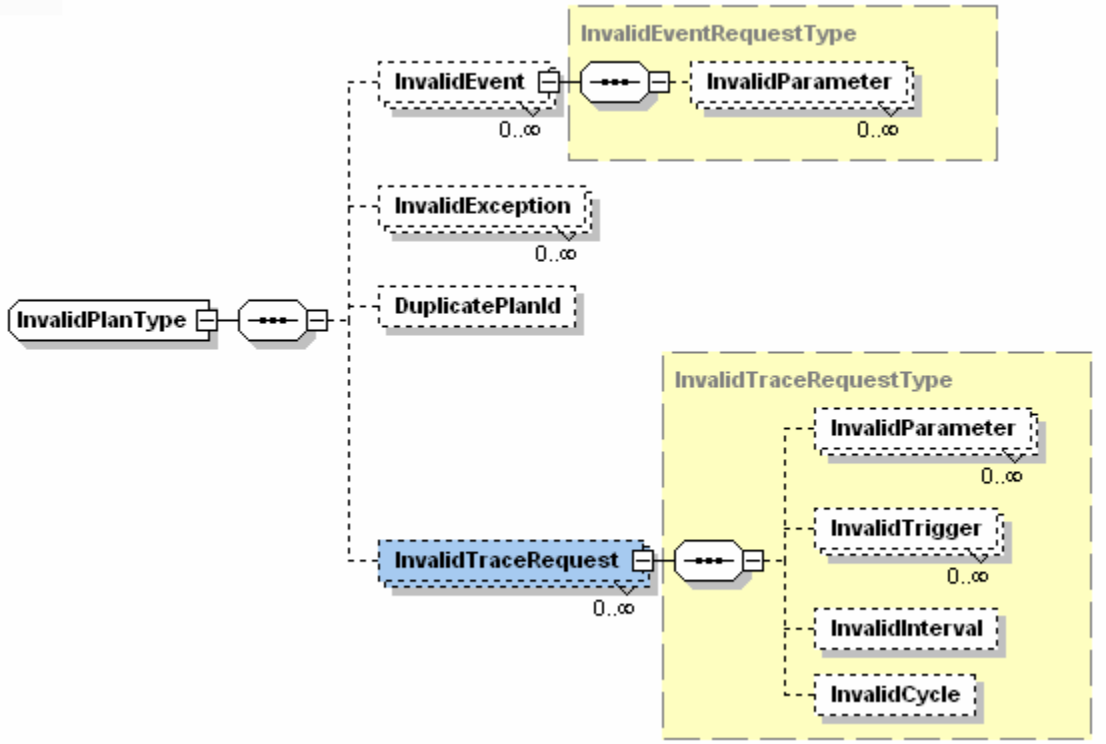

Figure 11
InvalidPlanType

7.2.11.4.14 *InvalidEventRequestType* — This global complexType is the XML Schema representation of the SEMI E134 InvalidEventRequest class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 30 Translation Table for InvalidEventRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
eventId	Text	Attribute	eventId: xsd:string
invalidSourceId	Boolean. .	Attribute	invalidSourceId: xsd:boolean
invalidEventId	Boolean	Attribute	invalidEventId: xsd:boolean
notProducedBySource	Boolean.	Attribute	notProducedBySource: xsd:boolean
isDuplicate	Boolean	Attribute	isDuplicate: xsd:boolean
invalidParameters	Unordered list of zero or more elements of type InvalidParameterRequest	Element	InvalidParameters: dcm:InvalidParameterRequestsType

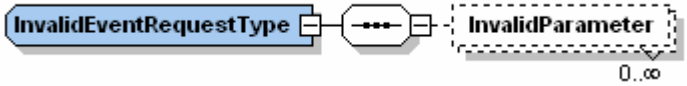

Figure 12
InvalidEventRequestType

7.2.11.4.15 *InvalidExceptionRequestType* — This global complexType is the XML Schema representation of the E134 InvalidExceptionRequest class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 31 Translation Table for InvalidExceptionRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
exceptionId	Text	Attribute	exceptionId: xsd:string
severity	Text	Attribute	severity: xsd:string
invalidSourceId	Boolean	Attribute	invalidSourceId: xsd:boolean
invalidExceptionId	Boolean	Attribute	invalidExceptionId: xsd:boolean
invalidSeverity	Boolean	Attribute	invalidSeverity: xsd:boolean
notProducedBySource	Boolean	Attribute	notProducedBySource: xsd:boolean
isDuplicate	Boolean	Attribute	isDuplicate: xsd:boolean

7.2.11.4.16 *InvalidParameterRequestType* — This global complexType is the XML Schema representation of the E134 InvalidParameterRequest class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 32 Translation Table for InvalidParameterRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
parameterName	Text	Attribute	parameterName: xsd:string
invalidSourceId	Boolean	Attribute	invalidSourceId: xsd:boolean
invalidParameterName	Boolean	Attribute	invalidParameterName: xsd:boolean
notProducedBySource	Boolean	Attribute	notProducedBySource: xsd:boolean
invalidContext	Boolean	Attribute	invalidContext: xsd:boolean

7.2.11.4.17 *InvalidTraceRequestType* — This global complexType is the XML Schema representation of the SEMI E134 InvalidTraceRequest class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 33 Translation Table for InvalidTraceRequestType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
traceId	Integer	Attribute	traceId: xsd:int
duplicateId	Boolean	Attribute	duplicateId: xsd:boolean
invalidParameters	Unordered list of zero or more elements of type InvalidParameterRequest	Element	InvalidParameters: dcm:InvalidParameterRequestsType
invalidTriggers	Unordered list of zero or more elements of type InvalidTrigger	Element	InvalidTriggers: dcm:InvalidTriggersType
invalidInterval	Zero or one element of type InvalidInterval	Element	InvalidInterval: dcm:InvalidIntervalType
invalidCycle	Zero or one element of type InvalidCycle	Element	InvalidCycle: dcm:InvalidCycleType

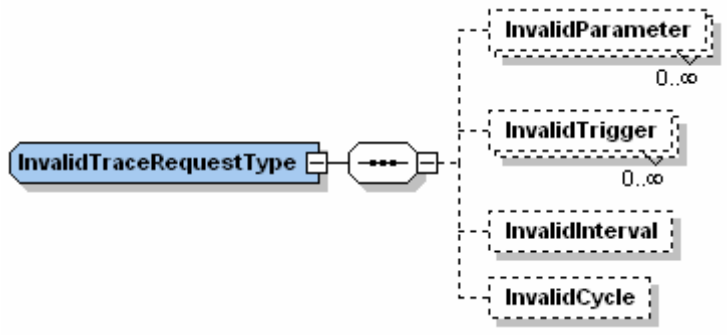

Figure 13
InvalidTraceRequestType

7.2.11.4.18 *InvalidTriggerType* — This global complexType is the XML Schema representation of the SEMI E134 InvalidTrigger class. The following table describes the mapping of the UML class to this XML Schema complexType

Table 34 Translation Table for InvalidTriggerType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
sourceId	Text	Attribute	sourceId: xsd:string
itemId	Text	Attribute	itemId: xsd:string
invalidStartTrigger	Boolean	Attribute	invalidStartTrigger: xsd:boolean
invalidEventTrigger	Boolean	Attribute	invalidEventTrigger: xsd:boolean
invalidExceptionState	Boolean	Attribute	invalidExceptionState: xsd:boolean
invalidSourceId	Boolean.	Attribute	invalidSourceId: xsd:boolean
invalidItemId	Boolean.	Attribute	invalidItemId: xsd:boolean
notProducedBySource	Boolean	Attribute	notProducedBySource: xsd:boolean
isDuplicate	Boolean	Attribute	isDuplicate: xsd:boolean

7.2.11.4.19 *InvalidIntervalType* — This global complexType is the XML Schema representation of the SEMI E134 InvalidInterval class. The following table describes the mapping of the UML class to this XML Schema complexType

Table 35 Translation Table for InvalidIntervalType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
validInterval	Floating point	Attribute	validInterval: xsd:float

7.2.11.4.20 *InvalidCycleType* — This global complexType is the XML Schema representation of the SEMI E134 InvalidCycle class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 36 Translation Table for InvalidCycleType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
needsStartTrigger	Boolean	Attribute	needsStartTrigger: xsd:Boolean
needsStopTrigger	Boolean	Attribute	needsStopTrigger: xsd:Boolean

7.2.12 *GetDefinedPlanIds*

7.2.12.1 *WSDL Operation Binding*

7.2.12.1.1 See the operation binding for "GetDefinedPlanIds" in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 37 GetDefinedPlanIds Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:GetDefinedPlanIds
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.12.2 *WSDL Operation*

7.2.12.2.1 See the portType operation definition for "GetDefinedPlanIds" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 38 GetDefinedPlanIds PortType Operation

Input Message Name	GetDefinedPlanIdsRequest
Output Message Name	GetDefinedPlanIdsResponse

7.2.12.3 *WSDL Message(s)*

7.2.12.3.1 See the message definitions for "E132HeaderMessage", "GetDefinedPlanIdsRequest", and "GetDefinedPlanIdsResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 39 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header GetDefinedPlanIdsRequest → dcm:GetDefinedPlanIdsRequestType GetDefinedPlanIdsResponse → dcm:GetDefinedPlanIdsResponseType

7.2.12.4 *XML Schema Types*

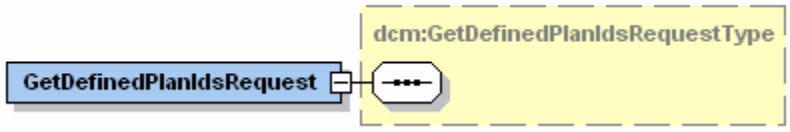

Figure 14
GetDefinedPlanIdsRequest

7.2.12.4.1 *GetDefinedPlanIdsRequest* — this global element acts as a top-level container entity representing the SOAP body of a GetDefinedPlanIds request. As defined in SEMI E134, this operation has no input arguments.

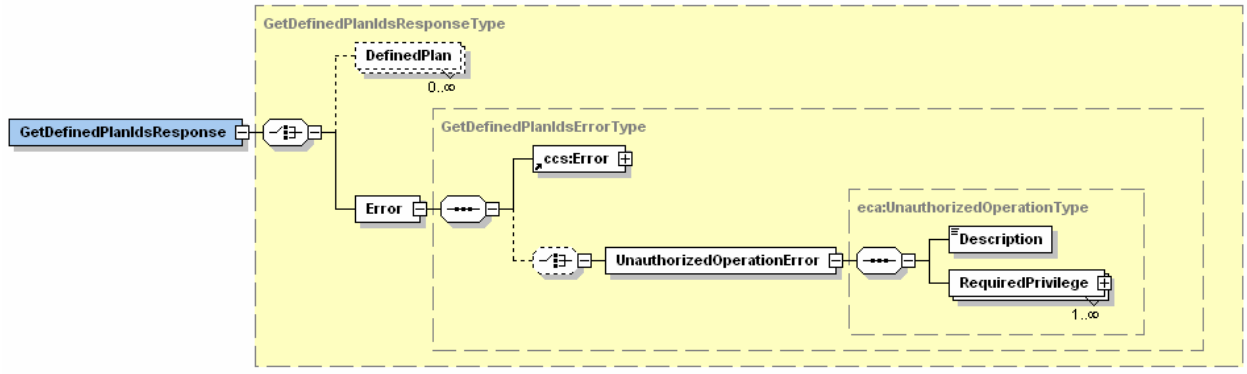

Figure 15
GetDefinedPlanIdsResponse

7.2.12.4.2 *GetDefinedPlanIdsResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 GetDefinedPlanIds request. It contains either an element named "DefinedPlans" of complexType "DCPDefinedType" or an element named "Error" of complexType "GetDefinedPlanIdsErrorType". GetDefinedPlanIdsErrorType contains the Common Components Error class and the ability to specify an UnauthorizedOperationError if appropriate.

Table 40 Translation Table for Output Arguments for the GetDefinedPlanIds Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
definedPlans	Unordered list of structured data, of type DCPDefined	Element	DefinedPlans: dcm:DCPDefinedType as defined in ¶7.2.11.4.11
error	N/A	Element	ErrorType: ccs:Error
error	Structure, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12

7.2.13 *GetPlanDefinition*

7.2.13.1 *WSDL Operation Binding*

7.2.13.1.1 See the operation binding for "GetPlanDefinition" in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 41 GetPlanDefinition Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:GetPlanDefinition
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.13.2 *WSDL Operation*

7.2.13.2.1 See the portType operation definition for "GetPlanDefinition" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 42 GetPlanDefinition PortType Operation

Input Message Name	GetPlanDefinitionRequest
Output Message Name	GetPlanDefinitionResponse

7.2.13.3 *WSDL Message(s)*

7.2.13.3.1 See the message definitions for "E132HeaderMessage", "GetPlanDefinitionRequest", and "GetPlanDefinitionResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 43 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header GetPlanDefinitionRequest → dcm:GetPlanDefinitionRequestType GetPlanDefinitionResponse → dcm:GetPlanDefinitionResponseType

7.2.13.4 *XML Schema Types*

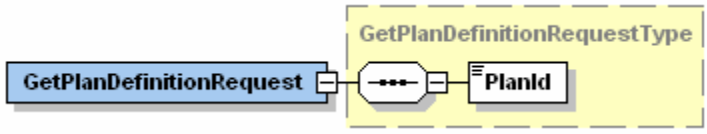

Figure 16
GetPlanDefinitionRequest

7.2.13.4.1 *GetPlanDefinitionRequest* — This global element acts as a top-level container entity representing the SOAP body of a GetPlanDefinition request. As defined in SEMI E134, this operation has one input argument: the Plan id that is being requested. This argument is modeled as a single element named "PlanId" of simpleType "xsd:string".

Table 44 Translation Table for Input Arguments for the GetPlanDefinitionOperation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
planId	Text	element	PlanId: xsd:string

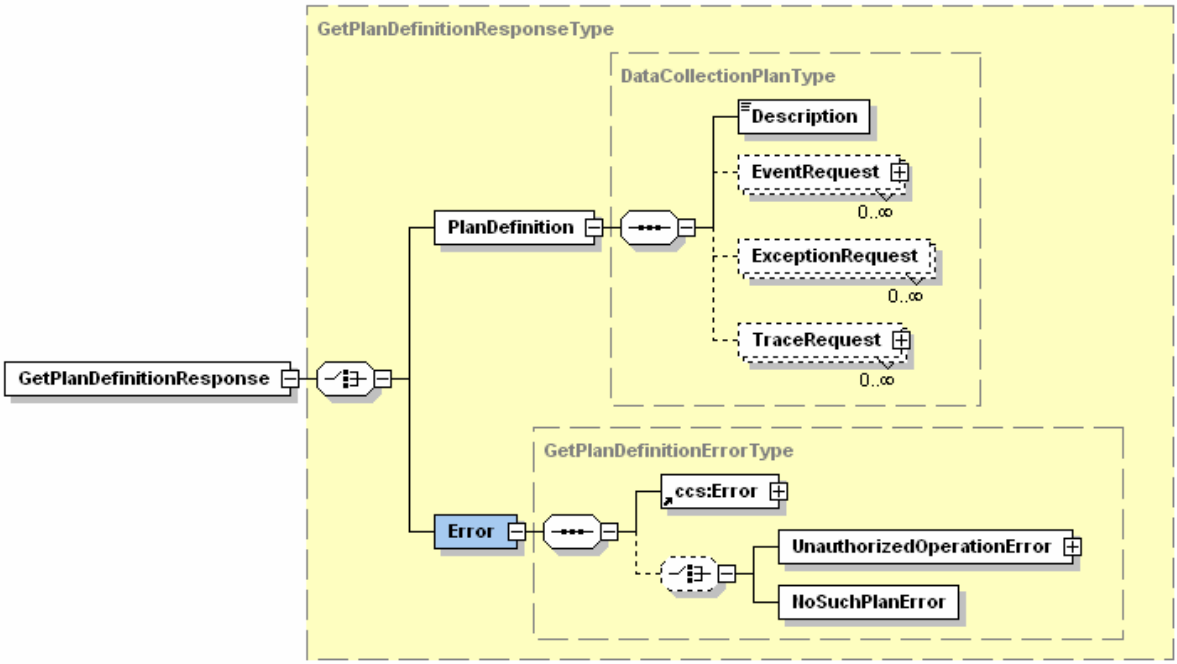

Figure 17
GetPlanDefinitionResponse

7.2.13.4.2 *GetPlanDefinitionResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 GetPlanDefinition request. It contains either an element named "PlanDefinition" of complexType "DataCollectionPlanType" or an element named "Error" of complexType "GetPlanDefinitionErrorType". GetPlanDefinitionErrorType contains the Common Components Error class and the ability to specify an UnauthorizedOperationError or NoSuchPlanError if appropriate.

Table 45 Translation Table for Output Arguments for the GetPlanDefinitionOperation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
planDefinition	Structure, of type DataCollectionPlan	Element	PlanDefinition: dcm:DataCollectionPlanType as defined in ¶7.2.11.4.2
error	N/A	Element	ErrorType: ccs:Error
error	Structure, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12
error	Structure, of type NoSuchPlan	Element	NoSuchPlanError: dcm:NoSuchPlanType

7.2.13.4.3 *NoSuchPlanType* — This global complexType is the XML Schema representation of the SEMI E134 NoSuchPlan class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 46 Translation Table for NoSuchPlanType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text	Attribute	planId: xsd:string

7.2.14 *ActivatePlan*

7.2.14.1 *WSDL Operation Binding*

7.2.14.1.1 See the operation binding for "ActivatePlan" in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 47 ActivatePlan Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:ActivatePlan
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.14.2 *WSDL Operation*

7.2.14.2.1 See the portType operation definition for "ActivatePlan" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 48 ActivatePlan PortType Operation

Input Message Name	ActivatePlanRequest
Output Message Name	ActivatePlanResponse

7.2.14.3 *WSDL Message(s)*

7.2.14.3.1 See the message definitions for "E132HeaderMessage", "ActivatePlanRequest", and "ActivatePlanResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 49 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header ActivatePlanRequest → dcm:ActivatePlanRequestType ActivatePlanResponse → dcm:ActivatePlanResponseType

7.2.14.4 *XML Schema Types*

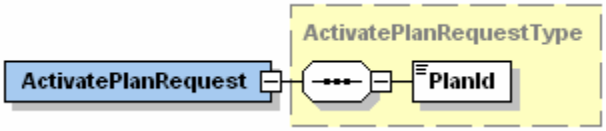

Figure 18
ActivatePlanRequest

7.2.14.4.1 *ActivatePlanRequest* — This global element acts as a top-level container entity representing the SOAP body of a ActivatePlan request. As defined in SEMI E134, this operation has one input argument: the Plan id that is being activated. This argument is modeled as a single element named "PlanId" of simpleType "xsd:string".

Table 50 Translation Table for Input Arguments for the ActivatePlanOperation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
planId	Text	element	PlanId: xsd:string

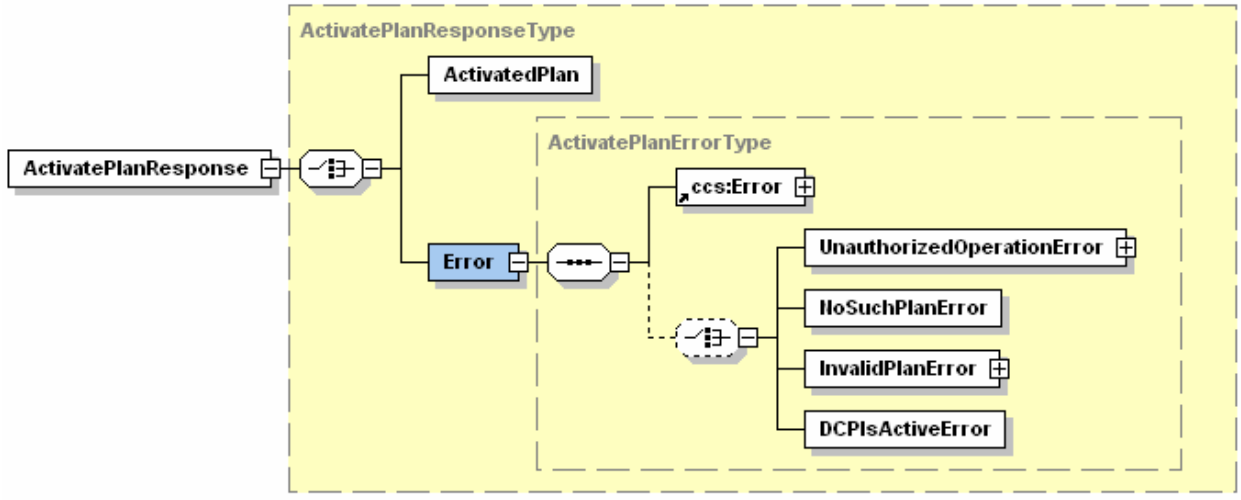

Figure 19
ActivatePlanResponse

7.2.14.4.2 *ActivatePlanResponse* — this global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 ActivatePlan request. It contains either an element named "activatedPlan" of complexType "DCPActivatedType" or an element named "Error" of complexType "ActivatePlanErrorType". GetPlanDefinitionErrorType contains the Common Components Error class and the ability to specify specific errors as described in the following table.

Table 51 Translation Table for Output Arguments for the GetDefinedPlanIds Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
activatedPlan	Structured data, of type DCPActivated	Element	ActivatedPlan: dcm:DCPActivatedType
error	N/A	Element	ErrorType: ccs:Error
error	Structured data, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12
error	Structured data, of type NoSuchPlan	Element	NoSuchPlanError: dcm: NoSuchPlanType, see ¶7.2.13.4.3
error	Structured data, of type InvalidPlan	Element	InvalidPlanError: dcm: InvalidPlanType, see ¶7.2.11.4.12
error	Structured data, of type DCPIsActive	Element	DCPIsActiveError: dcm: DCPActivatedType, see ¶7.2.14.4.3.

7.2.14.4.3 *DCPActivatedType* — This global complexType is the XML Schema representation of the SEMI E134 DCPActivated class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 52 Translation Table for DCPActivatedType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text	Attribute	planId: xsd:string
timeActivated	Text	Attribute	timeActivated: xsd:string
activatedBy	Text	Attribute	activatedBy: xsd:string

7.2.15 *GetActivePlanIds*

7.2.15.1 *WSDL Operation Binding*

7.2.15.1.1 See the operation binding for “GetActivePlanIds” in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 53 GetActivePlanIds Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:GetActivePlanIds
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.15.2 *WSDL Operation*

7.2.15.2.1 See the portType operation definition for “GetActivePlanIds” in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 54 GetActivePlanIds PortType Operation

Input Message Name	GetActivePlanIdsRequest
Output Message Name	GetActivePlanIdsResponse

7.2.15.3 *WSDL Message(s)*

7.2.15.3.1 See the message definitions for “E132HeaderMessage”, “GetActivePlanIdsRequest”, and “GetActivePlanIdsResponse” in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 55 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header GetActivePlanIdsRequest → dcm:GetActivePlanIdsRequestType GetActivePlanIdsResponse → dcm:GetActivePlanIdsResponseType

7.2.15.4 *XML Schema Types*

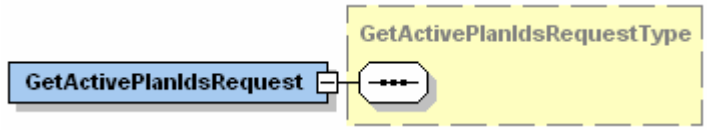

Figure 20
GetActivePlanIdsRequest

7.2.15.4.1 *GetActivePlanIdsRequest* — This global element acts as a top-level container entity representing the SOAP body of a GetActivePlanIds request. As defined in SEMI E134, this operation has no input arguments.

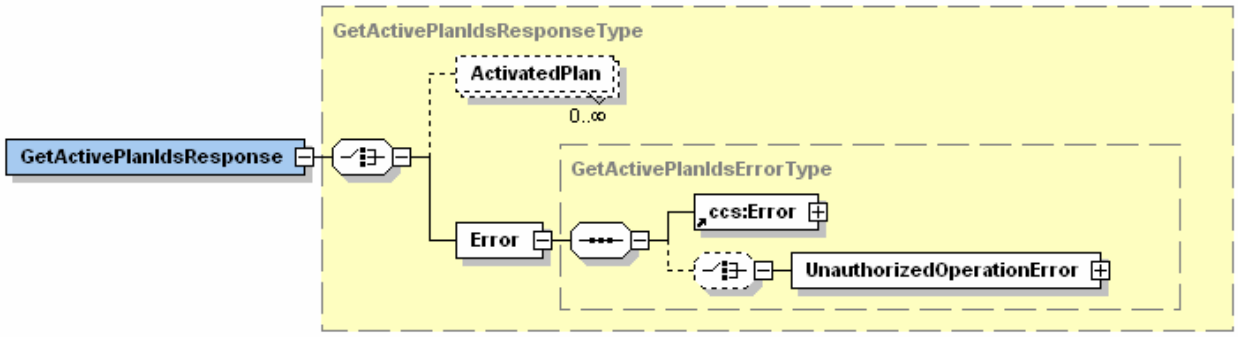

Figure 21
GetActivePlanIdsResponse

7.2.15.4.2 *GetActivePlanIdsResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 GetActivePlanIdsrequest. It contains either an element named "ActivePlans" of complexType "DCPActivatedType" or an element named "Error" of complexType "GetActivatePlanIdsErrorType". GetActivePlanIdsErrorType contains the Common Components Error class and the ability to specify specific errors as described in the following table.

Table 56 Translation Table for Output Arguments for the ActivatePlan Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
activePlans	Structured data, of type DCPActivated	Element	ActivePlans: dcm:DCPActivatedType, see ¶7.2.14.4.3
error	N/A	Element	ErrorType: ccs:Error
error	Structured data, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12

7.2.16 *DeactivatePlan*

7.2.16.1 *WSDL Operation Binding*

7.2.16.1.1 See the operation binding for "DeactivatePlan" in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 57 DeactivatePlan Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:DeactivatePlan
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.16.2 *WSDL Operation*

7.2.16.2.1 See the portType operation definition for "DeactivatePlan" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 58 DeactivatePlan PortType Operation

Input Message Name	DeactivatePlanRequest
Output Message Name	DeactivatePlanResponse

7.2.16.3 *WSDL Message(s)*

7.2.16.3.1 See the message definitions for "E132HeaderMessage", "DeactivatePlanRequest", and "DeactivatePlanResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 59 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header DeactivatePlanRequest → dcm:DeactivatePlanRequestType DeactivatePlanResponse → dcm:DeactivatePlanResponseType

7.2.16.4 *XML Schema Types*

Figure 22
DeactivatePlanRequest

7.2.16.4.1 *DeactivatePlanRequest* — This global element acts as a top-level container entity representing the SOAP body of a DeactivatePlan request. As defined in SEMI E134, this operation has two input arguments: the Plan id that is being deactivated and terminate specifying if the DCP should be deactivated for all consumers. These arguments are modeled as a attributes named "PlanId" of simpleType "xsd:string" and "terminate" of simpleType "xsd:boolean".

Table 60 Translation Table for Input Arguments for the DeactivatePlan Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
planId	Text	Attribute	PlanId: xsd:string
terminate	Boolean	Attribute	terminate: xsd:boolean

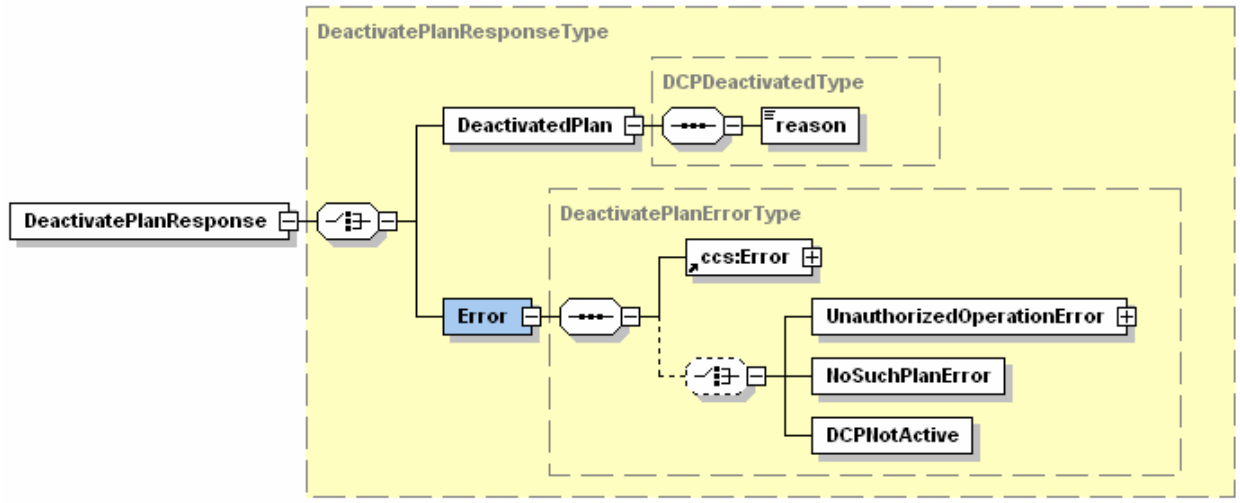

Figure 23
DeactivatePlanResponse

7.2.16.4.2 *DeactivatePlanResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 DeactivatePlan request. It contains either an element named "deactivatedPlan" of complexType "DCPDeactivatedType" or an element named "Error" of complexType "DeactivatePlanErrorType". DeactivatePlanErrorType contains the Common Components Error class and the ability to specify specific errors as described in the following table.

Table 61 Translation Table for Output Arguments for the DeactivatePlan Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
deactivatedPlan	Structure, of type DCPDeactivated	Element	DeactivatedPlan: dcm:DCPDeactivatedType
error	N/A	Element	ErrorType: ccs:Error
error	Structured data, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12
error	Structured data, of type NoSuchPlan	Element	NoSuchPlanError: dcm: NoSuchPlanType, see ¶7.2.13.4.3
error	Structured data, of type DCPNotActive	Element	DCPNotActive: dcm: DCPNotActiveType

7.2.16.4.3 *DCPDeactivatedType* — This global complexType is the XML Schema representation of the SEMI E134 DCPDeactivated class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 62 Translation Table for DeactivatePlanResponseType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text	Attribute	planId: xsd:string
timeDeactivated	Text	Attribute	timeDeactivated: xsd:dateTime
deactivatedBy	Text	Attribute	deactivatedBy: xsd:string
reason	Text	Attribute	reason: xsd:string

7.2.16.4.4 *DCPNotActive* — This global complexType is the XML Schema representation of the SEMI E134 DCPNotActive class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 63 Translation Table for DCPNotActiveType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text	Attribute	planId: xsd:string

7.2.17 *DeletePlan*

7.2.17.1 *WSDL Operation Binding*

7.2.17.1.1 See the operation binding for "DeletePlan" in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 64 DeletePlan Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:DeletePlan
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.17.2 *WSDL Operation*

7.2.17.2.1 See the portType operation definition for "DeletePlan" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 65 DeletePlan PortType Operation

Input Message Name	DeletePlanRequest
Output Message Name	DeletePlanResponse

7.2.17.3 *WSDL Message(s)*

7.2.17.3.1 See the message definitions for "E132HeaderMessage", "DeletePlanRequest", and "DeletePlanResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 66 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header DeletePlanRequest → dcm:DeletePlanRequestType DeletePlanResponse → dcm:DeletePlanResponseType

7.2.17.4 *XML Schema Types*

Figure 24
DeletePlanRequest

7.2.17.4.1 *DeletePlanRequest* — This global element acts as a top-level container entity representing the SOAP body of a DeletePlan request. As defined in SEMI E134, this operation has one input argument: the Plan id that is being deleted. This argument is modeled as a attributes named "PlanId" of simpleType "xsd:string".

Table 67 Translation Table for Input Arguments for the DeletePlan Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
planId	Text	Attribute	PlanId: xsd:string

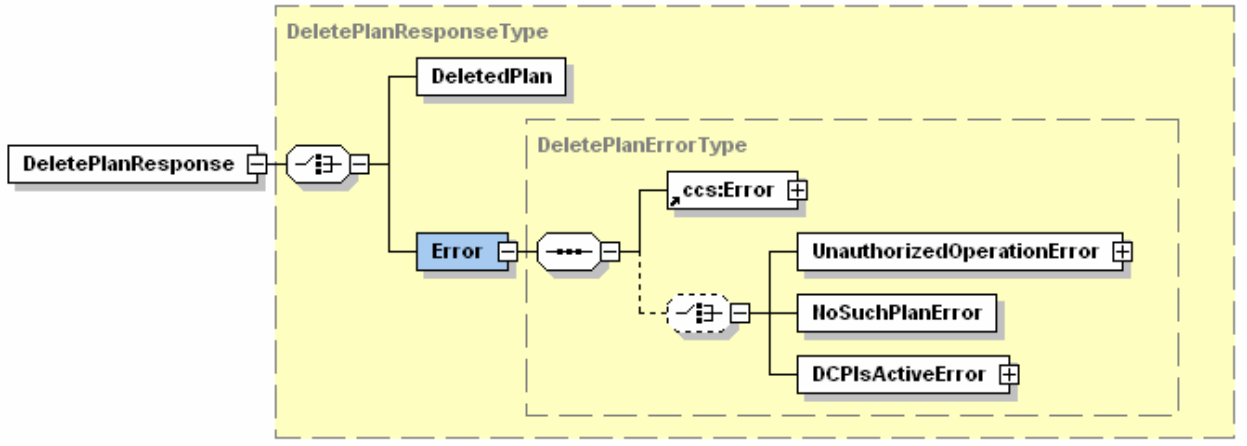

Figure 25
DeletePlanResponse

7.2.17.4.2 *DeletePlanResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 DeletePlan request. It contains either an element named "deletedPlan" of complexType "DCPDeletedType" or an element named "Error" of complexType "DeletePlanErrorType". DeletePlanErrorType contains the Common Components Error class and the ability to specify specific errors as described in the following table.

Table 68 Translation Table for Output Arguments for the DeletePlanOperation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
deletedPlan	Structured data, of type DCPDeleted	Element	DeletedPlan: dcm:DCPDeletedType
error	N/A	Element	ErrorType: ccs:Error
error	Structured data, of type UnauthorizedOperation	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12
error	Structured data, of type NoSuchPlan	Element	NoSuchPlanError: dcm: NoSuchPlanType, see ¶7.2.13.4.3
error	Structured data, of type DCPIsActive	Element	DCPIsActiveError: dcm: DCPActivatedType see ¶7.2.14.4.3

7.2.17.4.3 *DCPDeletedType* — This global complexType is the XML Schema representation of the SEMI E134 DCPDeleted class. The following table describes the mapping of the UML class to this XML Schema complexType.

Table 69 Translation Table for DCPDeletedType

Attribute or Role Name	*UML Type*	*XML Element or Attribute (or Reference)*	*XML Name/Type*
planId	Text	Attribute	planId: xsd:string
timeDeleted	Text	Attribute	timeDeleted: xsd:dateTime
deletedBy	Text	Attribute	deletedBy: xsd:string

7.2.18 *GetParameterValues*

7.2.18.1 *WSDL Operation Binding*

7.2.18.1.1 See the operation binding for "GetParameterValues" in E134-Equipment-Binding.wsdl. Key information is shown for convenience in the following table.

Table 70 GetParameterValues Operation Binding

SOAPAction	urn:semi-org:ws.E134-1.V0305.DCMEqp-binding:GetParameterValues
Input Headers (WSDL Message, Required)	E132HeaderMessage, required
Output Headers (WSDL Message, Required)	E132HeaderMessage, required

7.2.18.2 *WSDL Operation*

7.2.18.2.1 See the portType operation definition for "GetParameterValues" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 71 GetParameterValues PortType Operation

Input Message Name	GetParameterValuesRequest
Output Message Name	GetParameterValuesResponse

7.2.18.3 *WSDL Message(s)*

7.2.18.3.1 See the message definitions for "E132HeaderMessage", "GetParameterValuesRequest", and "GetParameterValuesResponse" in E134-Equipment-PortType.wsdl. Key information is shown for convenience in the following table.

Table 72 DataCollectionManagement Binding

Message Name → Schema Element Name	E132HeaderMessage → auth:E132Header GetParameterValuesRequest → dcm:GetParameterValuesRequestType GetParameterValuesResponse → dcm:GetParameterValuesResponseType

7.2.18.4 *XML Schema Types*

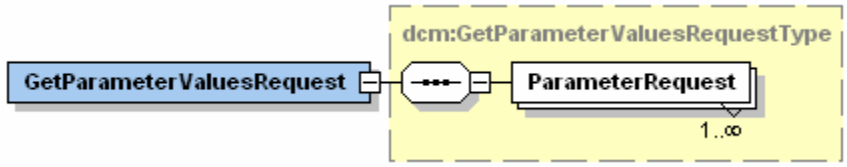

Figure 26
GetParameterValuesRequest

7.2.18.4.1 *GetParameterValuesRequest* — This global element acts as a top-level container entity representing the SOAP body of a GetParameterValues request. As defined in SEMI E134, this operation has one input argument: the Parameters that are being requested. This argument is modeled as a attributes named "ParameterRequests" of complexType "ParameterRequestType".

Table 73 Translation Table for Input Arguments for the GetParameterValues Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
parameterRequests	List of elements of type ParameterRequest	Element	ParameterRequests: dcm:ParameterRequestType as defined in ¶7.2.11.4.4

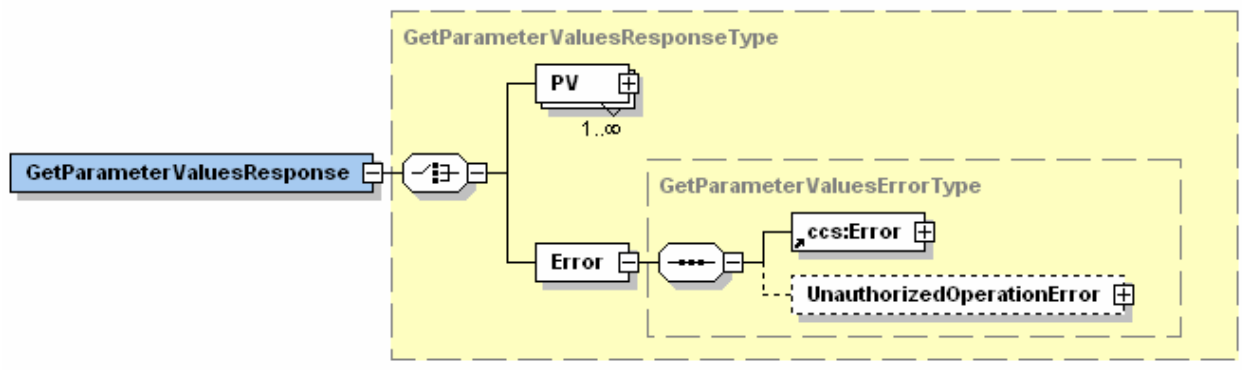

Figure 27
GetParameterValuesResponse

7.2.18.4.2 *GetParameterValuesResponse* — This global element acts as a top-level container entity representing the SOAP body of the response to an SEMI E134 GetParameterValues request. It contains either an element named "PV" of complexType "ParameterValueType" or an element named "Error" of complexType "GetParameterValuesErrorType". GetParameterValuesErrorType contains the Common Components Error class and the ability to specify specific errors as described in the following table.

Table 74 Translation Table for Output Arguments for the GetParameterValues Operation

Argument Name	*Format*	*XML Element or Attribute*	*XML Name/Type*
parameterValues	List of elements of type ParameterValue.	Element	PV: dcm:ParameterValueType
error	N/A	Element	ErrorType: ccs:Error
error	Structured data, of type UnauthorizedOperation.	Element	UnauthorizedOperationError: eca:UnauthorizedOperationType, see ¶7.2.11.4.12

7.2.18.4.3 *ParameterValueType* — This global complexType is the XML Schema representation of the SEMI E134 ParameterValues class. The following table describes the mapping of the UML class to this XML Schema complexType.

7.2.18.4.4 *Extended Parameter Value Types*

7.2.18.4.4.1 This specification adds new parameter value elements to the minimal set defined in SEMI E134. The following table shows how each SEMI E134 element has been mapped to the XML Schema element in this specification, and lists the additional classes not defined in SEMI E134.