

SEMI E134-0923

SPECIFICATION FOR DATA COLLECTION MANAGEMENT

This Standard was technically approved by the Information & Control Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on May 10, 2023. Available at www.semiviews.org and www.semi.org in September 2023; originally published July 2004; previously published October 2022.

NOTICE: The section symbol (§) is used to reference a numbered section, paragraph, or header and all subordinate paragraphs, headers, and embedded material (i.e., EXCEPTIONs, lists) therein. The paragraph symbol (§) is used to reference a particular numbered paragraph and embedded material therein. When duplicated (i.e., §§ and §§) the symbols are used to reference more than one section or paragraph, respectively.

NOTICE: Paragraphs entitled ‘NOTE:’ are not an official part of this Standard or Safety Guideline and are not intended to modify or supersede the official Standard or Safety Guideline. These have been supplied by the global technical committee to enhance the usage of the Standard or Safety Guideline.

Table of Contents

1 Purpose	3
2 Scope	3
3 Limitations	3
4 Referenced Standards and Documents	3
5 Terminology	4
6 Conventions	5
6.1 Terms	5
6.2 Notation	5
7 Clients and Consumers	7
7.3 Client and Consumer Conventions	8
7.4 Client and Consumer Use Case Scenarios	9
7.5 Common Errors	9
8 Background	10
8.1 Equipment Data Collection	10
9 Overview	13
9.1 Data Collection Concepts	13
9.2 Trace Data	13
9.3 Events	13
9.4 Exceptions	13
9.5 Data Collection Plans	14
9.6 Equipment Performance Monitoring	14
9.7 Ad-Hoc Data Requests	14
9.8 Generic Class Definitions	14
10 DataCollectionManager Interface	15
10.3 DefinePlan	16
10.4 GetDefinedPlanIds	24
10.5 ActivatePlan	25
10.6 GetActivePlanIds	26
10.7 DeactivatePlan	27
10.8 DeletePlan	29
10.9 GetParameterValues	31
10.10 GetObjTypeInstancelDs	32
10.11 GetCurrentPerformanceStatus	33
10.12 GetCurrentDateTime	34
10.13 GetCurrentSetExceptions	34
10.14 GetObjTypeAttributes	35

11 DCP Privilege Model.....	37
11.1 Privileges	37
12 DataCollectionPlan Definition.....	39
12.1 DataCollectionPlan	39
12.2 EventRequest	41
12.3 ExceptionRequest.....	42
12.4 TraceRequest	44
12.5 Trigger.....	46
12.6 ParameterReference	48
12.7 ExceptionTrigger	48
12.8 ParameterRequest	49
12.9 CachedDataRequest	49
13 Data Collection Plan State Models	51
13.1 DataCollectionPlan State Model	51
13.2 DataCollectionPlan-Active State Model.....	53
13.3 Trace Request State Model	56
13.4 CachedDataRequest State Model	60
13.5 Reporting Cached Data Results.....	62
14 Performance Monitoring.....	62
14.1 Performance Conditions State Model.....	62
15 Data Collection Report Formats	64
15.1 Data Collection Reports	64
15.2 Time Format	73
16 DCPConsumer Interface.....	73
16.2 DCPConsumer.....	74
17 Requirements for Compliance	78
18 Related Documents.....	78
Related Information 1 Data Collection Context	79
Related Information 2 UML Terminology	82
Related Information 3 Cached Data Examples.....	85
Related Information 4 Eventrequest/Eventreport Examples.....	86

1 Purpose

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

2 Scope

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

2.2 This Specification defines the behavior associated with the execution of data collection plans in the form of finite state machines. The formats for data produced as a result of executing a data collection plan are also defined, as is the interface that shall be supported by an EDA Application of data collection plan output.

2.3 This Specification defines a way for the equipment to notify users when the combination of activities on the equipment, including data acquisition, is causing the equipment to perform below supplier-defined criteria.

2.4 This Specification defines a way for users to make ad-hoc on-demand requests for data from the equipment, outside of a data collection plan.

NOTICE: SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the Documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 *Abstract Model*

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

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

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

3.1.4 This Specification is an abstract model only. Adjunct Specifications shall be developed to map this Specification to an implementation technology.

4 Referenced Standards and Documents

4.1 *SEMI Standards and Safety Guidelines*

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

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

SEMI E39 — Specification for Object Services: Concepts, Behavior, and Services

SEMI E40 — Specification for Processing Management

SEMI E94 — Specification for Control Job Management

SEMI E120 — Specification for the Common Equipment Model (CEM)

SEMI E125 — Specification for Equipment Self Description (EqSD)

NOTE 1: SEMI E125 is being updated by the Information & Control Global Technical Committee.

SEMI E132 — Specification for Equipment Client Authentication and Authorization

NOTE 2: SEMI E132 is being updated by the Information & Control Global Technical Committee.

4.2 ISO® Standards¹

ISO 8601 — Data Elements and Interchange Formats – Information Interchange – Representation of Dates and Times

ISO 11578:1996 — Information Technology – Open Systems Interconnection – Remote Procedure Call (RPC)

4.3 OMG® Standards²

OMG 01-09-67 — Unified Modeling Language®³, v1.4

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

5 Terminology

5.1 Abbreviations and Acronyms

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

5.1.2 DCP — data collection plan

5.1.3 UML — Unified Modeling Language

5.2 Definitions

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

5.2.2 Related Information 2, § R2-1, contains useful definitions of UML terms taken directly from the UML standard. UML terms used in this Standard conform to these definitions. Please refer to this section as needed.

5.2.3 *Client* — an application process that communicates with the equipment to access equipment resources and initiates the connection with the Equipment Server. The Client performs management functionality, and can receive notification and operational messages from the Equipment Server. [SEMI E132]

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

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

5.2.6 *Consumer* — an application process that communicates with the equipment to get data. The Equipment Server initiates the connection to a specific endpoint to communicate with the Consumer. The Consumer does not perform any management functionality, can receive notification messages from the Equipment Server, and always receives operational messages from the Equipment Server when it exists. [SEMI E132]

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

¹ International Organization for Standardization, ISO Central Secretariat, Chemin de Blandonnet 8, CP 401-1214 Vernier, Geneva, Switzerland; Telephone: +41.22.749.01.11, Fax: +41.22.733.34.30, <http://www.iso.org>. ISO trademark is owned by International Organization for Standardization.

² Object Management Group, Inc., 250 First Ave. Suite 100, Needham, MA 02494, USA; Telephone: +1.781.444.0404; Fax: +1.781.444.0320; <http://www.omg.org>. OMG trademark is owned by Object Management Group.

³ Trademark is owned by Object Management Group.

5.2.8 *EDA Application* — software application that implements at least one Client or Consumer. [SEMI E132]

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

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

5.2.11 *notification message* — type of message sent by the Equipment Server to notify the recipient something has changed on the equipment (typically related to status, management, or metadata). [SEMI E132]

5.2.12 *operational message* — type of message sent by the Equipment Server to either a Client or Consumer with collected or analysis data from the equipment. [SEMI E132]

5.2.13 *operational message recipient* — Client or Consumer associated with the session that is receiving operational messages for the session. [SEMI E132]

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

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

6 Conventions

6.1 Terms

6.1.1 *Class, Data Type, Stereotype Name* — This Specification uses the terms ‘class’, ‘data type’, ‘type’, or the class stereotype name interchangeably to refer to a UML class.

6.1.2 *Built-In DCP* — This Specification uses the term ‘built-in’ DCP to refer to pre-defined DCPs that are provided with the equipment. Built-in DCPs can be activated by any Client, but cannot be changed or deleted by any Clients. Refer to ¶ 12.1.3 for details.

6.1.3 *Reporting, Buffering, Sending* — This Specification uses the term ‘reporting’ to refer to the act of collecting event, exception, or trace data values to later be sent to an operational message recipient. The term ‘buffering’ is used to refer to the act of storing a data report on the equipment for some period of time before sending to an operational message recipient. The term ‘sending’ is used to refer to the act of sending a report to an operational message recipient.

6.2 Notation

6.2.1 *Unified Modeling Language Notation* — All class and sequence diagrams in this Specification make use of the Unified Modeling Language notation.

6.2.1.1 *UML Stereotypes* — This Specification uses the following stereotypes in class diagrams:

- <<privilege>> — Classes with this stereotype identify an abstract privilege concept that grants permission to a Client to perform an activity or operation.
- <<interface>> — Classes with this stereotype define a collection of operations and their semantics. It is the responsibility of the equipment and EDA Applications, where specified, to provide implementations of these classes.
- <<enumeration>> — Classes with this stereotype identify data types that have a finite set of discrete values that can be assumed by instances of the class.

6.2.2 *UML Associations* — The mechanism used for realizing UML associations between classes is implementation dependent. This Standard is abstract in nature, and does not specify or imply any such mechanism. Any adjunct standard that provides an implementation of this Specification shall include a description of the mechanism used for representing the UML associations shown in this Standard.

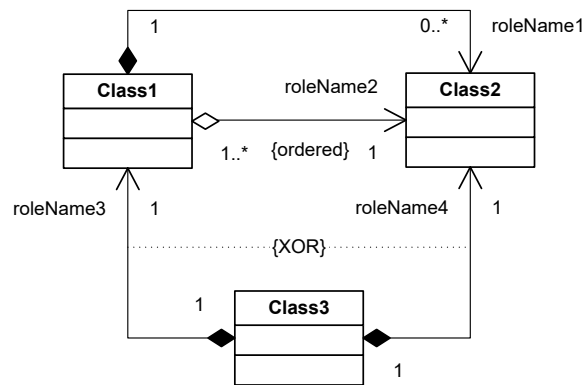


Figure 1
Association Notation

6.2.3 Association Notation — This Standard uses the UML aggregation diamond adornment, role names, end multiplicities, and navigability in all class diagrams specifying associations. Un-adorned associations are not used, for economy of notation. All role names are public by default; therefore the visibility symbols are not used. Refer to the UML 1.4 specification for details.

6.2.3.1 Open diamond adornments indicate that instances of the target class may be shared among aggregate classes. Closed diamond adornments indicate that target instances belong to at most one composite class. The part-whole semantics of the aggregation/composition symbol is not significant in this Specification. Refer to the OMG 01-09-67 specification for further information.

6.2.3.2 Some diagrams use the UML ‘ordered’ or ‘XOR’ constraints where applicable. An ‘ordered’ constraint on an association indicates that the order in which the target instances appear in the aggregation is semantically significant, and is described in the supporting text of the diagram. Refer to the OMG 01-09-67 specification for further information.

6.2.3.3 An ‘XOR’ constraint indicates that only one of many possible associations can be instantiated at any one time for a given instance of the association class. Refer to the OMG 01-09-67 specification for further information.

6.2.4 Association Tables — The table below provides an example of the tables used to list and describe associations between classes defined in this Specification.

Table 1 Association Table Format

<i>Association Role Name</i>	<i>Definition</i>	<i>Comments</i>

6.2.4.1 Association Role Name — The name of the association role being specified.

6.2.4.2 Definition — Describes the function or purpose of the association.

6.2.4.3 Comments — Any additional comments or notes regarding the association.

6.2.5 Attribute Tables — The table below provides an example of the tables used to list and describe attributes of classes defined in this Specification.

Table 2 Attribute Table Format

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
		Refer to list below

6.2.5.1 *Form* — Defines the data type of the attribute. The terms used to describe data types in this column are defined in the SEMI International Standards Compilation of Terms, or are included as part of the Specification. Refer to the compilation of terms for the definition of SEMI type name meanings.

6.2.6 *Operation Definition Tables* — The table below provides an example of the tables used to list and describe the interface operations defined in this Specification.

Table 3 Operation Definition

<i>Operation</i>	<i>Description</i>	<i>Type</i>	<i>Requestor/Sender</i>	<i>Responder/Receiver</i>
		Refer to list below		

6.2.6.1 *Operation* — Specifies the name of the operation.

6.2.6.2 *Type* — Specifies the messaging semantics of the operation. Only Request-Reply (RR) and Fire-and-Forget (FF) semantics are used in this Specification. Request-Reply messages are messages that involve an initiator and a receiver. In a Request-Reply exchange, the initiator sends a single request message to the receiver, and the receiver sends a single reply message to that request back to the initiator. Fire-and-Forget messages are messages that involve a sender and a receiver. In a Fire-and-Forget exchange, the sender sends a single message to the receiver, with no associated response.

6.2.6.3 *Requestor/Sender* — For RR semantics, identifies the entity that makes the request, for FF semantics, identifies the entity that sends the message. Can be either 'Client' or 'Equipment Server'.

6.2.6.4 *Responder/Receiver* — For RR semantics, identifies the entity that responds to the request, for FF semantics, identifies the entity that receives the message. Can be either 'Client', 'Consumer' or 'Equipment Server'.

6.2.7 *Operation Argument Definition Table* — The table below provides an example of the tables used to list and describe arguments for interface operations defined in this Specification.

Table 4 Operation Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
		Refer to list below.	Refer to list below.

6.2.7.1 *Argument* — Specifies the name of the argument.

6.2.7.2 *Kind* — Specifies whether the argument is an 'in', 'out', or 'error' argument for the operation. 'error' arguments always function as 'out' arguments, but indicate that the operation did not complete successfully. All possible errors that could be returned by an operation are listed in these tables. Unless otherwise indicated, only one of the listed errors can be returned by an operation in a single invocation.

6.2.7.3 *Form* — Defines the data type of the argument. The terms used to describe data types in this column are defined in the SEMI International Standards Compilation of Terms, or are included as part of the Specification. Refer to the compilation of terms for the definition of SEMI type name meanings.

7 Clients and Consumers

7.1 With a networked environment, it is possible for EDA applications to be on different machines and fulfill different functions for the equipment users.

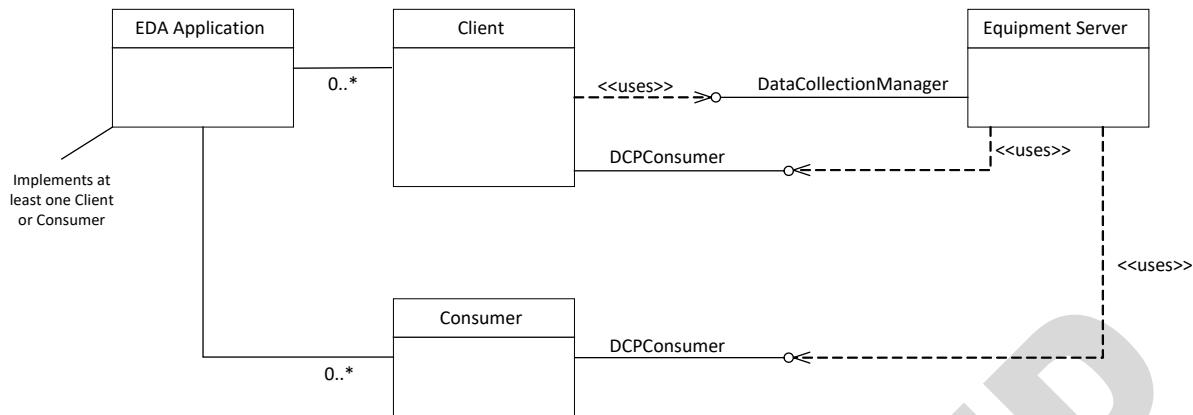


Figure 2
Session Interfaces

7.2 This Specification uses SEMI E132 for authentication and authorization.

7.2.1 Refer to SEMI E132 for details on notification messages, operational messages and how to specify who receives them when establishing a session.

7.3 *Client and Consumer Conventions* — For this Specification, the following conventions are used:

7.3.1 *Client* — This application process initiates the connection to the Equipment Server.

- It is associated with one SessionId.
- It uses the DataCollectionManager interface to perform management functionality (e.g., defining and activating data collection plans).
- The Client can receive notification messages from Equipment Server by implementing the DCPCConsumer interface.
- The Client can receive operational messages sent from the Equipment Server by implementing the DCPCConsumer interface. (When a Consumer is specified for the session, the Client does not receive operational messages.)

7.3.2 *Consumer* — The Equipment Server initiates the connection to this application process.

- It is associated with one SessionId.
- It does not do any management functionality.
- The Consumer always implements the DCPCConsumer interface.
- Can receive notification messages.
- When present, the Consumer always receives operational messages sent from the Equipment Server. (When the Consumer is specified for the session, the Client cannot receive operational messages)

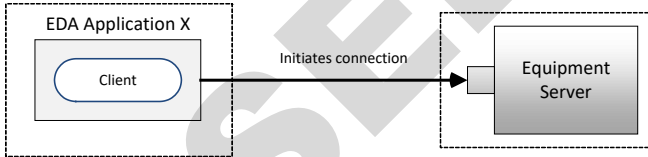
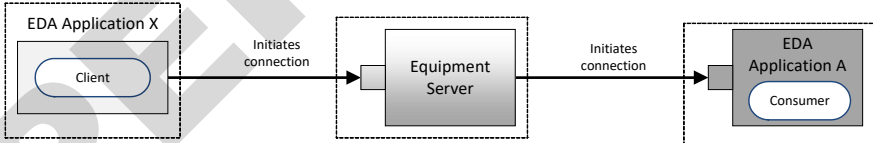
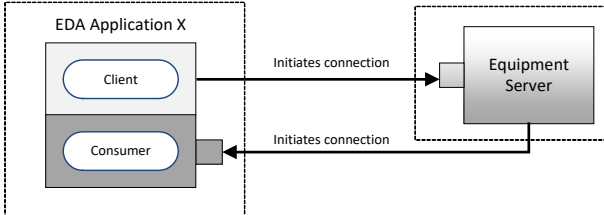
7.3.3 *Equipment Server* — This software application implements the DataCollectionManager interface. It uses the DCPCConsumer interface to send notification and operational messages to the appropriate recipients (i.e., Clients and Consumers).

7.3.4 *EDA Application* — Software application that implements one of the following scenarios:

- one or more Clients
- one or more Consumers
- one or more Clients and one or more Consumers

7.4 Client and Consumer Use Case Scenarios — For this Specification, the following use cases are supported:

Table 5 Clients and Consumer Use Case Scenarios

Scenario	Architecture
1) No one gets operational messages	EDA Application X has the Client; it does not get operational messages. There is no Consumer. Equipment Server does not send SessionPing messages. 
2) Same entity gets operational messages	EDA Application X has the Client that gets operational messages. There is no Consumer. Equipment Server sends SessionPing messages to the Client. This scenario is only supported by implementations that use a protocol that supports bi-directional messages. 
3) Separate entity gets operational messages (Equipment initiates communications)	EDA Application X has the Client. EDA Application A has the Consumer that gets operational messages (through the specified endpoint). Equipment Server sends SessionPing messages to the Consumer.  Variation – EDA Application X has both the Client and the Consumer. The Equipment Server does not treat this variation any differently. 

7.5 Common Errors — Common errors independent of this Specification can be reported when they are applicable and not explicitly defined as SEMI E132-specific errors.

7.5.1 A Subordinate Standard shall identify how common errors are used with the implementation technology described in that Subordinate Standard.

7.5.2 Common errors defined in Table 6 shall be supported by all SEMI E134 operations. (For example, the DefinePlan() operation can return the common error ‘System shutting down’, even though it is not explicitly listed as an error parameter in Table 10, DefinePlan Argument Definitions.)

Table 6 Common Errors Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
error	System not ready.	error	Text, shall include the text ‘System not ready’.
error	System shutting down.	error	Text, shall include the text ‘System shutting down’.
error	Specified configuration is invalid.	error	Text, shall include the text ‘Specified configuration is invalid’, and a description of the invalid configuration.

7.5.3 Common SEMI E132 errors defined in Table 7 shall be supported by all SEMI E134 operations. (For example, the DefinePlan() operation can return the SEMI E132 error ‘Unrecognized session’, even though it is not explicitly listed as an error parameter in Table 10, DefinePlan Argument Definitions.)

Table 7 Common SEMI E132 Errors Parameter Definitions

<i>Parameter</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
error	Unauthorized operation.	error	Structured data, of type SEMI E132 UnauthorizedOperationError
error	Unrecognized session.	error	Text, shall include the text ‘Unrecognized session’ and the bad session id.
error	Unrecognized equipment identifier	error	Text, shall include the text ‘Unrecognized equipment identifier’ and the bad equipment identifier.

8 Background

8.1 Equipment Data Collection

8.1.1 Manufacturing equipment can be comprised of hundreds or thousands of components that interoperate in order for the equipment to perform its intended function. The volume of information that such equipment can produce, and the variation among equipment users in the data used for process control and other applications, makes it impractical to send all equipment data off-tool all the time, or to agree on a fixed set of data to make available for each equipment type. It is therefore important for the equipment to support the ability for Clients to dynamically define the data that is of interest to them (refer to Figure 3).

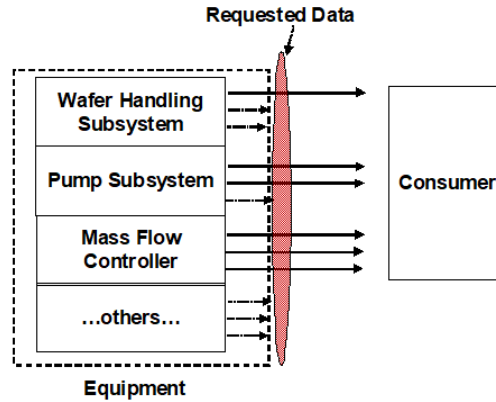


Figure 3
Client Specifies What Data Should be Sent Off-Tool

8.1.2 In addition to declaring interest in specific data that an EDA Application is interested in, different EDA Applications have different requirements for the acceptable latency with which data should be reported off-tool. For example, an EDA Application that is monitoring a trace of several process variables in order to detect an excursion needs to have this data available off-tool almost as quickly as the equipment can produce it (refer to Figure 4). An EDA Application that is monitoring low-level sensor/actuator activity for later diagnosis prefers not to have all of this data transmitted as it is produced, but buffered on the tool and transmitted at periodic intervals (refer to Figure 5).

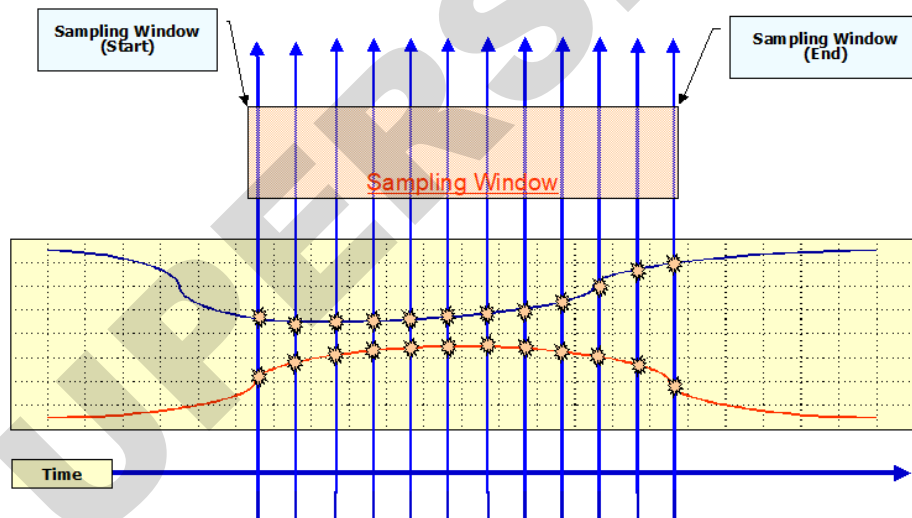


Figure 4
Un-Buffered Data Transmission

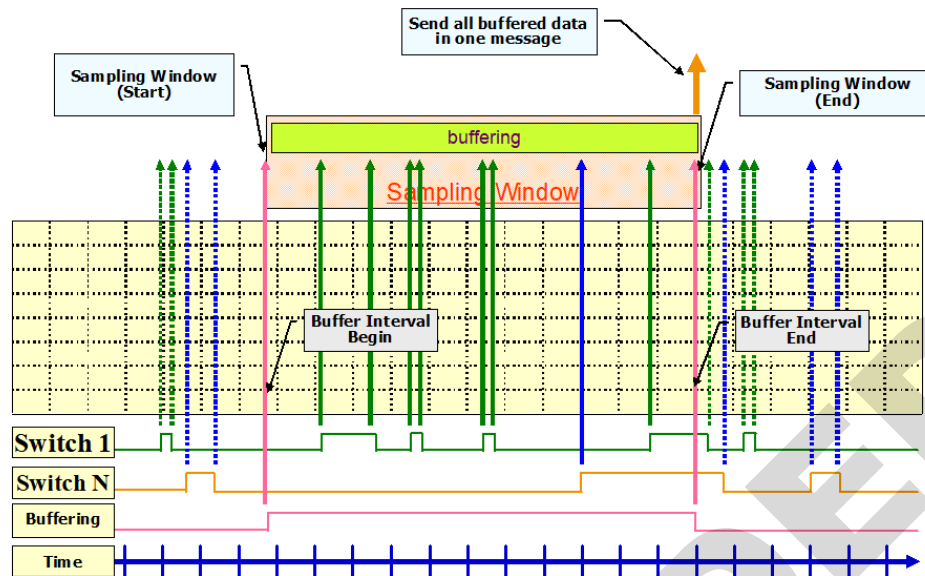


Figure 5
Buffered Data Transmission

8.1.3 Because a single tool is likely to have more than one EDA Application simultaneously interested in these different styles of data collection, a method for managing groups of related data as a unit can make it easier for EDA Applications to define, enable, and disable the transmission of data. An example of this organization is shown in Figure 6.

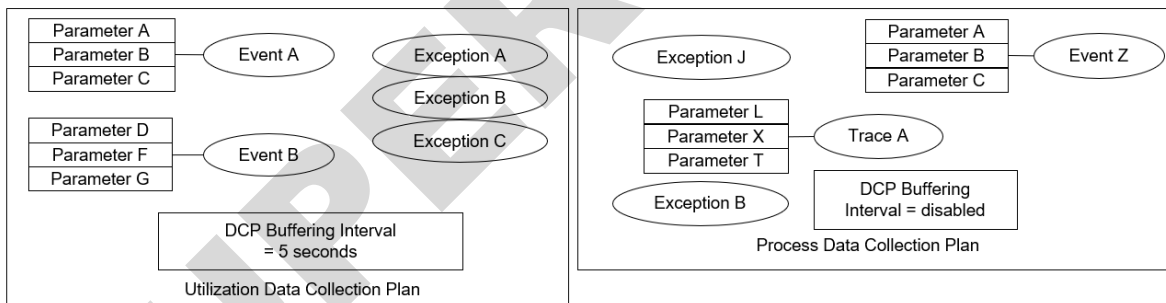


Figure 6
Data Collection Schemes with Different Objectives

8.1.4 It is possible for EDA Applications to submit data collection requests which, when executing together, significantly tax the equipment's computing resources. When the equipment detects that this is occurring, it should be able to provide a warning to Clients and/or Consumers that a performance threshold is being reached. EDA Applications can respond to these warnings by adjusting or disabling the active data collection requests. In the worst case, if the equipment determines that continuing to operate under degraded performance conditions may seriously jeopardize its ability to perform its intended function, the equipment can terminate all data collection activity.

8.1.5 This Specification describes the concepts and behaviors necessary to facilitate these capabilities on equipment.

9 Overview

9.1 Data Collection Concepts

9.1.1 This Specification allows Clients to describe a set of data in which they are interested, and to ask the equipment to send only that data off-tool, according to a buffering policy specified by the Clients. The types of data that are supported by this Specification are described in the following sections.

9.2 Trace Data

9.2.1 Trace data allows data acquisition recipients to collect data values that are continuously changing, or that change frequently (e.g., pressure, temperature, RF power, magnetic field strength, or individual actuator state changes). This Standard defines a mechanism for specifying when trace data collection begins and ends, what the collection frequency is, and what variables are to be collected in the trace (refer to ¶ 12.4).

9.2.2 This Specification defines a mechanism for specifying when to start tracing (refer to ¶ 12.4.2.2):

- On the occurrence of a specific event or exception from a set of possible events and exceptions
- As soon as enabled

9.2.3 This Specification defines a mechanism for specifying when to stop tracing (refer to ¶ 12.4.2.3):

- On the occurrence of a specific event or exception from a set of possible events and exceptions
- After a specified number of collection results have been accumulated
- As soon as disabled

9.2.4 This Specification defines a mechanism for identifying parameters to be included in trace data collection (refer to ¶ 12.8).

9.2.5 This Specification defines a mechanism to support on-tool buffering, allowing trace reports containing many collection results to be sent off-tool at a regular interval (refer to §§ 12.1.4.1 and 12.4.1.2).

9.2.6 Cached data is a special form of trace data collection, where the data samples are collected, cached on the equipment, and then reported in an efficient array format. This technique allows for data to be collected from the equipment and reported at much higher frequency than trace data collection. The higher frequency allows for EDA applications that require higher data sampling frequencies. Cached data is setup using requests and reports separate from trace requests and reports.

9.3 Events

9.3.1 Events allow data acquisition recipients to determine when a subsystem, process job, sensor/actuator, or other modeled system undergoes a state transition. Events may have data variables associated with the occurrence of the event. Such variables can include process data such as pressures, temperatures, or voltages, or context data such as job related data, recipe IDs, equipment configuration or offset values. Typically Clients need to be able to specify which variables are to be provided with the event when it is communicated.

9.3.2 This Specification defines a mechanism for identifying events and parameters to be included in data collection (refer to §§ 12.2 and 15.1.3).

9.3.3 This Specification defines a mechanism to support on-tool buffering of events, allowing a periodic report of any events that occurred during a specified interval to be sent off-tool (refer to § 12.1.4.1).

9.4 Exceptions

9.4.1 Exceptions allow data acquisition recipients to know when errors, warnings, or alarms occur on the equipment. These may indicate conditions which affect processing, safety, or the proper operation of the equipment. Exceptions may have data variables associated with the occurrence of the exception. The list of these variables will be fixed by the equipment implementation.

9.4.2 This Specification defines a mechanism for identifying exceptions to be included in data collection (refer to §§ 12.2.2.1 and 15.1.4).

9.4.3 This Specification defines a mechanism to support on-tool buffering of exceptions, allowing a periodic report of any exceptions that occurred during a specified interval to be sent off-tool (refer to § 12.1.4.1).

9.5 Data Collection Plans

9.5.1 A data collection plan assembles all of the related trace, event, and exception data that may be required for a specific EDA Application's purpose. For example, an EDA Application interested in tracking the utilization of the equipment may only be interested in specific events that mark actions the equipment takes that indicate how it is being exercised, certain exceptions that may interrupt the equipment, and no trace data. Such a data collection plan might be in effect at all times the equipment is running.

9.5.2 A fault detection EDA Application may be interested in trace, event, and exception data for a specific processing run, and may change the collected data depending on the recipe used. Such data collection plans may only be in effect while specific jobs are in process on the equipment. Coordination of data collection plan activation with job activity on the equipment is the EDA Application's responsibility, and is not specified in this Standard.

9.5.3 This Standard specifies a mechanism for Clients to define one or more data collection plans off-tool and submit those definitions to the equipment (refer to ¶ 10.3 and § 12). This Standard specifies a mechanism for determining what plans have been created on the equipment, how they are defined, and whether or not they are activated (refer to ¶¶ 10.3.16, 10.4.2, and 10.6). A mechanism for activating, deactivating, and deleting individual plans is also specified (refer to ¶¶ 10.5, 10.7, and 10.8). The required behavior of data collection plans is specified in § 13.

9.6 Equipment Performance Monitoring

9.6.1 It is possible for Clients to submit data collection requests which, when executing together, significantly tax the equipment's computing resources, in addition to any load placed on the equipment by jobs that are in progress. This Standard specifies a mechanism for notifying all DCP Consumers when such a condition has been detected, to identify what DCPs and jobs are in progress at the time, and to identify the specific equipment components that are being adversely affected (refer to ¶¶ 10.11, 13.4, 16.2.5, and 16.2.6).

9.7 Ad-Hoc Data Requests

9.7.1 Ad-hoc data requests are used in cases where the value of some data items is needed at a specific instance in time, and which may not be associated with a specific event or exception. As a simple example, consider an application that monitors equipment utilization. Such an application may have an initialization procedure that includes determining the current state of each monitored component on the equipment. Without a mechanism for requesting this information at initialization, the application would instead have to wait for state change events from each component before it could determine their state. This may not be acceptable if the application provides a visualization of the equipment to an end user who relies on the application to call attention to problem conditions on the equipment.

9.7.2 This Standard specifies a mechanism for Clients to issue ad-hoc requests for data in order to support these uses, through the GetObjTypeAttributes, GetObjTypeInstanceIds, and GetParameterValues, operations of the DataCollectionManager interface (refer to ¶¶ 10.9 and 10.10).

9.8 Generic Class Definitions

9.8.1 In some instances, this Standard uses generic class definitions rather than defining them explicitly as they are tied to a specific implementation technology. A Subordinate Standard shall identify how these generic classes are used with the implementation technology described in that Subordinate Standard.

Table 8 Generic Class Definitions

Generic Class	Details
Parameter Value Class	Used in operation GetParameterValues and classes EventReport, ExceptionReport, CollectedData, CachedDataReport and AttributeResult. The parameter value class shall support all of the parameter types defined in SEMI E125 and shall also support the reporting of a null value and 'no value'.
Operator Class	Used in comparing parameter values required by the Condition class. The operator class shall support the operations listed in Table 63 Operator Definitions and Applicability.

10 DataCollectionManager Interface

10.1 Figure 7 shows the interface used to manage data collection plans on the equipment, as well as making ad-hoc requests for specific data, and for determining the current performance conditions on the equipment.

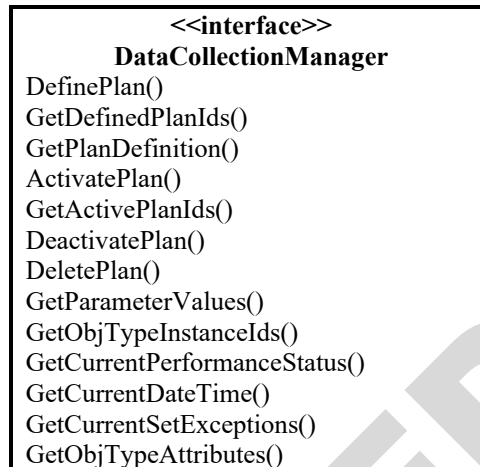


Figure 7
DataCollectionManager Interface

10.2 *DataCollectionManager* — The DataCollectionManager interface provides Clients with operations to manage data collection activities on the equipment. The interface provides operations for Clients to submit new DCP's, activate and deactivate them, look up existing plans, and delete defined plans. The DataCollectionManager interface also provides operations to request the current equipment performance status as well as requesting ad-hoc data on demand. The Equipment Server shall implement the DataCollectionManager interface. All of the operations defined in this interface can return error codes defined in SEMI E132.

10.2.1 DataCollectionManager Operations

Table 9 DataCollectionManager Operation Definition

<i>Operation</i>	<i>Description</i>	<i>Type</i>	<i>Requestor/ Sender</i>	<i>Responder/ Receiver</i>
DefinePlan	Submit a new plan definition to the equipment for validation.	RR	Client	Equipment Server
GetDefinedPlanIds	Retrieve a list of all valid DCP ids known to the equipment.	RR	Client	Equipment Server
GetPlanDefinition	Retrieve the definition of a specific DCP.	RR	Client	Equipment Server
ActivatePlan	Activate a defined DCP.	RR	Client	Equipment Server
GetActivePlanIds	Retrieve a list of all currently active DCPs.	RR	Client	Equipment Server
DeactivatePlan	Deactivate an active DCP.	RR	Client	Equipment Server
DeletePlan	Delete a DCP.	RR	Client	Equipment Server
GetParameterValues	Retrieve the most recent values of one or more Parameters.	RR	Client	Equipment Server

<i>Operation</i>	<i>Description</i>	<i>Type</i>	<i>Requestor/ Sender</i>	<i>Responder/ Receiver</i>
GetObjTypeInstanceIds	Retrieve a current list of the unique instance ids for one or more E39 ObjTypes.	RR	Client	Equipment Server
GetCurrentPerformanceStatus	Retrieve the most recent equipment performance status.	RR	Client	Equipment Server
GetCurrentDateTime	Retrieve the current equipment date and time.	RR	Client	Equipment Server
GetCurrentSetExceptions	Retrieve a list of current set exceptions in the equipment. (Refer to § 10.13 for further explanation.)	RR	Client	Equipment Server
GetObjTypeAttributes	Request the most recent values for one or more ObjType attributes.	RR	Client	Equipment Server

10.3 *DefinePlan* — Upon receiving this request, the equipment shall first verify that the Client has sufficient privilege (refer to § 11) to define a new DCP. If so, the equipment shall then validate the plan name for uniqueness and all referenced events, exceptions, parameters, and trace reports for existence and correctness. The equipment shall place no a-priori limitation on the number of plans that can be defined.

10.3.1 Any hard limitations imposed by constraints on available storage capacity shall be documented by the equipment supplier. If such limitations are encountered at runtime, the equipment shall return an equipment-defined error indicating that the DCP was not successfully defined, and a human-readable description of the reason. The mechanism for supporting such errors shall be fully specified in implementation mappings provided for this Specification.

10.3.2 If there are any issues with the submitted plan, the equipment shall return a description of all problems detected with the plan (refer to ¶ 10.3.5 for a description of possible DCP definition problems), and the plan shall not be defined on the equipment.

10.3.3 If there are no issues with the submitted plan, the equipment shall record the date and time at which the plan definition was submitted and the id of the Client that submitted the plan. The equipment shall store this information and the plan definition in nonvolatile storage. It shall thereafter be legal for DCP requests to refer to the submitted plan by its id field.

10.3.4 *DefinePlan* Operation Arguments

Table 10 DefinePlan Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
newPlan	The definition of a new DCP.	in	Structured data, of type DataCollectionPlan, described in ¶ 12.1.2.
planDefined	An acknowledgement that the new DCP definition was well-formed, correct, and supportable.	out	Structured data, of type DCPDefined, described in ¶ 10.3.5.
error	The Client submitting the DCP does not have sufficient privilege to define DCPs.	error	Structured data, of type UnauthorizedOperation, described in ¶ 10.3.6.
error	The DCP definition is invalid.	error	Structured data, of type InvalidPlan, described in ¶ 10.3.7.

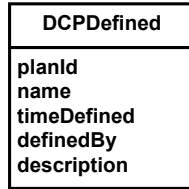


Figure 8
DCPDefined Class

10.3.5 *DCPDefined* — This class describes a DCP that has been successfully defined on the equipment.

10.3.5.1 *DCPDefined Attribute Definition Table*

Table 11 DCPDefined Attribute Definition

Attribute Name	Definition	Form
planId	Identifies a defined DCP.	Text, equal to the 'id' attribute of the DataCollectionPlan defined on the equipment.
name	Client-defined (or supplier-defined, in the case of built-in DCPs) human-readable name for the DCP.	Text, equal to the 'name' attribute of the DataCollectionPlan defined on the equipment.
timeDefined	The date and time at which the DCP was originally defined.	Text, formatted according to ¶ 15.2. For built-in plans (refer to ¶ 12.1.3), this field shall be equal to the date and time at which the plan was installed on the equipment.
definedBy	The identity of the Client that defined the DCP.	Text, equal to a valid identifier for the Client that took the action (refer to Limitations, ¶ 3.1.1). For built-in plans (refer to ¶ 12.1.3), this field shall be a URN of the form 'urn:semi-org:equipment'. The valid values to use for this attribute to identify Clients shall be specified by any adjunct Standard defining a technical binding for this Specification.
description	Client-defined (or supplier-defined, in the case of built-in DCPs) description of the purpose of the DCP.	Text, equal to the 'description' attribute of the DataCollectionPlan defined on the equipment.

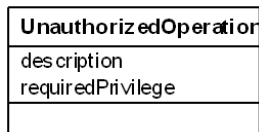


Figure 9
UnauthorizedOperation Error

10.3.6 *UnauthorizedOperation* — This class describes an authorization error associated with a specific operation.

10.3.6.1 *UnauthorizedOperation Attribute Definition Table*

Table 12 UnauthorizedOperation Attribute Definition

Attribute Name	Definition	Form
description	A description of the error.	Text, including the name of the operation and a description of any important context associated with the original request.

Attribute Name	Definition	Form
requiredPrivilege	The name of the privilege that is required for the operation.	Text, equal to a valid privilege name or identifier (refer to § 11) defined on the equipment that the Client shall possess in order to execute the operation. The valid values to use for this attribute shall be specified by any adjunct Standard defining a technical binding for this Specification.

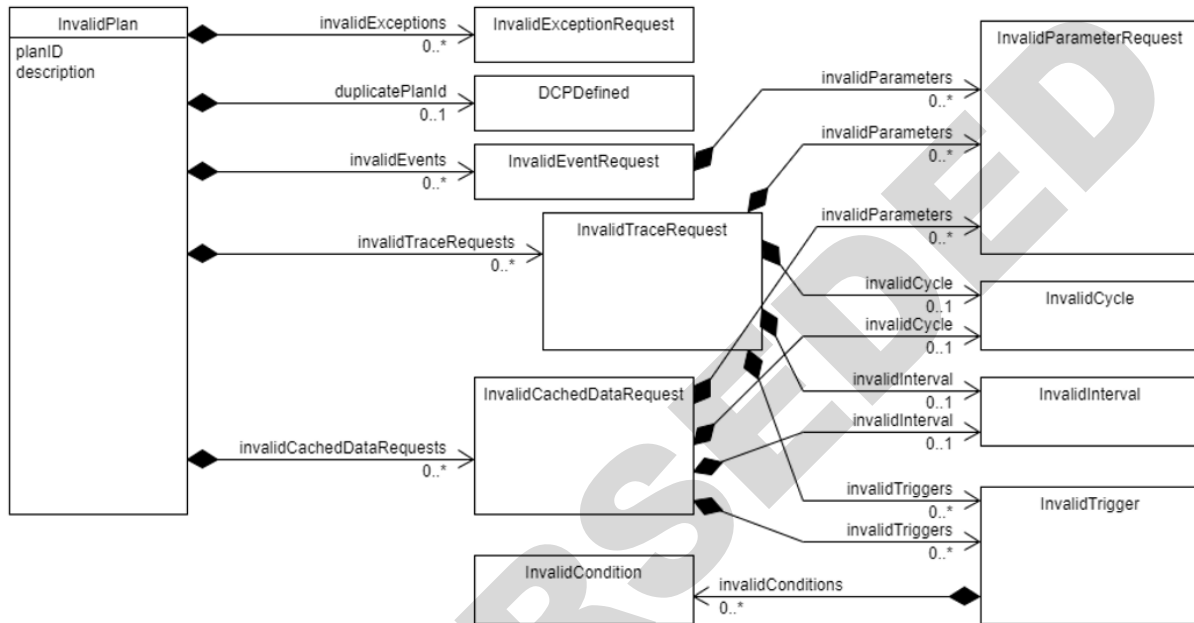


Figure 10
InvalidPlan Error

10.3.7 *InvalidPlan* — This class provides an overall description of all errors detected with a submitted DCP.

10.3.7.1 *InvalidPlan* Attribute Definition Table

Table 13 InvalidPlan Attribute Definition

Attribute Name	Definition	Form
planId	The identifier of the invalid DCP.	Text, equal to the 'id' attribute of the submitted DataCollectionPlan.
description	A human-readable description of why the DCP was rejected.	Text.

10.3.7.2 InvalidPlan Association Definition Table

Table 14 InvalidPlan Association Definition

Association Role Name	Definition	Comments
invalidEvents	List of event requests that have one or more errors.	Unordered list of elements of type InvalidEventRequest, described in ¶ 10.3.8.
invalidExceptions	List of exception requests that have one or more errors.	Unordered list of elements of type InvalidExceptionRequest, described in ¶ 10.3.9.
duplicatePlanId	Description of a pre-existing plan that has an identical 'id' attribute.	Structure, of type DCPDefined, described in ¶ 10.3.5.
invalidTraceRequests	List of trace requests that have one or more errors.	Unordered list of elements of type InvalidTraceRequest, described in ¶ 10.3.11.
invalidCachedData Requests	List of cached data requests that have one or more errors.	Unordered list of elements of type InvalidCachedDataRequest described in ¶ 10.3.17.

10.3.8 InvalidEventRequest — This class describes errors detected in requested events in a submitted plan definition. An event request can be invalid if (1) the equipment does not recognize the sourceId, (2) the equipment does not recognize the eventName, (3) the sourceId and eventName are both recognized by the equipment, but the specified source is not the source of the requested event, or (4) the same source/event combination appears in more than one event request within the DCP.

10.3.8.1 InvalidEventRequest Attribute Definition Table

Table 15 InvalidEventRequest Attribute Definition

Attribute Name	Definition	Form
sourceId	The sourceId from the invalid event request provided in the DCP.	Text.
eventName	The eventName from the invalid event request provided in the DCP.	Text.
invalidSourceId	Whether or not the sourceId identifies a valid source supported by the equipment.	Boolean. If true, the identified source is not recognized.
invalidEventName	Whether or not the eventName identifies a valid event supported by the equipment.	Boolean. If true, the event name is not recognized.
notProducedBySource	Whether or not the identified source is the source of the identified event.	Boolean. If true, both the source and event names are recognized, but the identified source does not generate the requested event.
isDuplicate	Whether or not the requested event appears more than once in the DCP.	Boolean.

10.3.8.2 InvalidEventRequest Association Definition Table

Table 16 InvalidEventRequest Association Definition

Association Role Name	Definition	Comments
invalidParameters	List of parameter requests for the event that have one or more errors.	Unordered list of zero or more elements of type InvalidParameterRequest, described in ¶ 10.3.10.

10.3.9 *InvalidExceptionRequest* — This class describes errors detected in requested exceptions in a submitted plan definition. An exception request can be invalid if (1) the equipment does not recognize the sourceId, (2) the equipment does not recognize the exceptionName, (3) the equipment recognizes both the sourceId and exceptionName, but the specified source is not the source of the requested exception, (4) the requested severity is not supported, or (5) the same exception criteria appears in more than one exception request within the DCP.

10.3.9.1 *InvalidExceptionRequest Attribute Definition Table*

Table 17 InvalidExceptionRequest Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
sourceId	The sourceId from the invalid exception request provided in the DCP.	Text.
exceptionName	The exceptionName from the invalid exception request provided in the DCP.	Text.
severity	The severity from the invalid exception request provided in the DCP.	Text. The value shall match one of the enumeration strings defined for the severity attribute of the Exception class defined in SEMI E125.
invalidSourceId	Whether or not the sourceId identifies a valid source supported by the equipment.	Boolean. If true, the identified source is not recognized.
invalidExceptionName	Whether or not the exceptionName identifies a valid exception supported by the equipment.	Boolean. If true, the exception name is not recognized.
invalidSeverity	Whether or not the specified severity is recognized by the equipment.	Boolean. If true, the severity is not recognized.
notProducedBySource	Whether or not the identified source is the source of the identified exception.	Boolean. If true, both the sourceId and exceptionName are recognized, but the identified source does not generate the requested exception.
isDuplicate	Whether or not the same exception criteria appears in more than one exception request in the DCP.	Boolean.

10.3.10 *InvalidParameterRequest* — This class describes errors detected in requested parameters in a submitted plan definition. Parameters are requested with either an event or as part of a trace or cached data request. A parameter request can be invalid if (1) the equipment does not recognize the sourceId, (2) the equipment does not recognize the parameterName, (3) the equipment recognizes both the sourceId and parameterName but the specified source is not the source of the requested parameter, or (4) the Parameter is produced by the specified source but (a) the equipment does not support reporting the parameter with the corresponding event request or (b) the equipment does not support including this parameter in a trace or cached data request.

10.3.10.1 InvalidParameterRequest Attribute Definition Table

Table 18 InvalidParameterRequest Attribute Definition

Attribute Name	Definition	Form
sourceId	The sourceId from the requested parameter.	Text.
parameterName	The name of the requested parameter.	Text.
invalidSourceId	Whether or not the sourceId identifies a valid source supported by the equipment.	Boolean. If true, the identified source is not recognized.
invalidParameterName	Whether or not the parameterName identifies a valid parameter supported by the equipment.	Boolean. If true, the identified parameter is not recognized.
notProducedBySource	Whether or not the identified source is the source of the requested parameter.	Boolean. If true, the source does not provide the requested parameter.
invalidContext	Whether or not the requested parameter is available in the requested context.	Boolean. If true, the equipment does not support reporting this parameter with the requested event, or including the parameter in a trace request.
notAvailableForCached Data	The parameter is not available for cached data collection.	Boolean. If true, the parameter is not available for cached data requests.

10.3.11 InvalidTraceRequest — This class describes errors detected in requested traces in a submitted plan definition. A trace request can be invalid if (1) the same trace id appears in more than one TraceRequest within the same DCP, (2) the TraceRequest includes an invalid ParameterRequest, (3) the TraceRequest specifies an invalid start/stop trigger, (4) the TraceRequest specifies an invalid trigger cycle, or (5) the TraceRequest specifies an interval that can't be supported by the equipment.

10.3.11.1 InvalidTraceRequest Attribute Definition Table

Table 19 InvalidTraceRequest Attribute Definition

Attribute Name	Definition	Form
traceRequestId	The traceRequestId from the invalid trace request provided in the DCP.	Positive Integer.
duplicateId	Whether or not the trace id is duplicated.	Boolean. If true, the trace id appears more than once in the same DCP.

10.3.11.2 InvalidTraceRequest Association Definition Table

Table 20 InvalidTraceRequest Association Definition

Association Role Name	Definition	Comments
invalidParameters	List of parameter requests for the trace request that have one or more errors.	Unordered list of zero or more elements of type InvalidParameterRequest, described in ¶ 10.3.10.
invalidTriggers	List of start/stop triggers for the trace request that have one or more errors.	Unordered list of zero or more elements of type InvalidTrigger, described in ¶ 10.3.12.
invalidInterval	Indicates an unsupportable trace interval.	Zero or one element of type InvalidInterval, described in ¶ 10.3.14.
invalidCycle	Indicates that the trace request contains an invalid cycle.	Zero or one element of type InvalidCycle, described in ¶ 10.3.15.
invalidGroupSize ForInterval	Indicates an unsupportable groupSize for the specified intervalInSeconds.	Zero or one element of type InvalidGroupSizeForInterval, described in ¶ 10.3.16.

10.3.12 InvalidTrigger — This class describes errors detected in requested start/stop triggers for a TraceRequest or CachedDataRequest in a submitted plan definition. A trigger can be invalid if (1) the equipment does not recognize the sourceId, (2) the equipment does not recognize the eventName or exceptionName, (3) the sourceId and eventName or exceptionName are both recognized by the equipment, but the specified source is not the source of the requested event or exception, or (4) the same start trigger appears more than once in the same trace request, or the same stop trigger appears more than once in the same trace or cached data request.

10.3.12.1 InvalidTrigger Attribute Definition Table

Table 21 InvalidTrigger Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
invalidStartTrigger	Whether or not the problem is with a start trigger. False indicates that the problem is with a stop trigger.	Boolean.
invalidEventTrigger	Whether or not the problem is with an event trigger. False indicates that the problem is with an exception trigger.	Boolean.
invalidExceptionState	The requested exception state is not defined for the exception.	Boolean.
sourceId	The sourceId from the invalid trigger provided in the DCP.	Text.
itemName	The event or exception name from the invalid trigger provided in the DCP.	Text.
invalidSourceId	Whether or not the sourceId identifies a valid source supported by the equipment.	Boolean. If true, the identified source is not recognized.
invalidItemName	Whether or not the eventName or exceptionName identifies a valid event or exception supported by the equipment.	Boolean. If true, the identified exception or event is not recognized.
notProducedBySource	Whether or not the identified source is the source of the identified event or exception.	Boolean. If true, the source does not generate the requested exception or event.
isDuplicate	Whether or not the same trigger appears more than once as a start trigger, or as a stop trigger.	Boolean.

10.3.12.2 InvalidTrigger Association Definition Table

Table 22 Trigger Association Definition

<i>Association Role Name</i>	<i>Definition</i>	<i>Comments</i>
invalidConditions	List of zero or more instances of type InvalidCondition.	Allowed only if the InvalidTrigger is for an EventTrigger.

10.3.13 InvalidCondition — This class describes errors detected in the Conditions defined for requested start/stop triggers for a TraceRequest or CachedDataRequest in a submitted plan definition.

10.3.13.1 InvalidCondition Attribute Definition Table

Table 23 InvalidCondition Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
unknownOperator	The operator is not one of the defined choices.	Boolean
badOperator	The operator specified is not compatible with the type of Parameter referenced by the Condition.	Boolean
unknownParameter	The referenced Parameter is not defined for this equipment.	Boolean
invalidParameter	The referenced Parameter is not valid for the trigger event or the format of the reference Parameter is not allowed for Conditions.	Boolean

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
badValue	The conditionValue specified is not compatible with the referenced Parameter or is a value that is not allowed.	Boolean

10.3.14 *InvalidInterval* — This class describes errors detected in requested trace or cached data intervals in a submitted plan definition. A trace or cached data interval is invalid if the equipment cannot collect the requested parameters at the requested frequency. The error returns the closest valid trace or cached data interval that can be supported by the equipment.

10.3.14.1 *InvalidInterval Attribute Definition Table*

Table 24 InvalidInterval Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
validInterval	A valid interval closest to the requested interval, expressed in seconds.	Floating point.

10.3.15 *InvalidCycle* — This class describes errors detected in a cyclical trace or cached data request in a submitted plan definition. A trace or cached data cycle is invalid if the TraceRequest or CachedDataRequest 'isCyclical' attribute (refer to ¶ 12.4) is true, but does not include both a start and stop trigger.

10.3.15.1 *InvalidCycle Attribute Definition Table*

Table 25 InvalidCycle Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
needsStartTrigger	Whether or not the TraceRequest or CachedDataRequest is missing a start trigger.	Boolean.
needsStopTrigger	Whether or not the TraceRequest or CachedDataRequest is missing a stop trigger.	Boolean.

10.3.16 *InvalidGroupSizeForInterval* — This class describes a problem detected in a submitted plan definition when the groupSize is too small for the specified intervalInSeconds. The resulting reporting interval (intervalInSeconds * groupSize) requires that reports be sent more frequently than the equipment can support. The error returns the smallest valid groupSize that can be supported by the equipment.

10.3.16.1 *InvalidReportInterval Attribute Definition Table*

Table 26 InvalidGroupSizeForInterval Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
validGroupSize	The smallest groupSize value allowed for the requested intervalInSeconds.	Positive Integer.

10.3.17 *InvalidCachedDataRequest* — This class describes errors detected in cached data request in a submitted plan definition. A cached data request can be invalid if (1) the same cached data id appears in more than one CachedDataRequest within the same DCP, (2) the CachedDataRequest includes an invalid ParameterRequest, (3) the CachedDataRequest specifies an invalid start/stop trigger, (4) the CachedDataRequest specifies an invalid collection interval, or (5) the CachedDataRequest specifies an invalid reporting interval.

10.3.17.1 InvalidCachedDataRequest Attribute Definition Table

Table 27 InvalidCachedDataRequest Attribute Definition

Attribute Name	Definition	Form
cachedDataRequestId	The id from the invalid cached data request provided in the DCP.	Positive Integer.
duplicateId	Whether or not the cached data id is duplicated.	Boolean. If true, the cached data id appears more than once in the same DCP.

10.3.17.2 InvalidCachedDataRequest Association Definition Table

Table 28 InvalidCachedDataRequest Association Definition

Association Role Name	Definition	Comments
invalidParameters	List of parameter requests for the cached data request that have one or more errors.	Unordered list of zero or more elements of type InvalidParameterRequest, described in ¶ 10.3.10.
invalidTriggers	List of start/stop triggers for the cached request that have one or more errors.	Unordered list of zero or more elements of type InvalidTrigger, described in ¶ 10.3.12.
invalidInterval	Indicates an unsupportable collection interval for the cached data request.	Zero or one element of type InvalidInterval, described in ¶ 10.3.14.
invalidCycle	Indicates that the cached data request contains an invalid cycle.	Zero or one element of type InvalidCycle, described in ¶ 10.3.15.
invalidReportingInterval	Indicates that the cached data request contains an unsupportable reporting interval.	Zero or one element of type InvalidReportingInterval, described in ¶ 10.3.18.

10.3.18 *InvalidReportingInterval* — This class provides the Client with a valid reporting interval when an invalid value was requested.

Table 29 InvalidReportingInterval Attribute Definition

Attribute Name	Definition	Form
validReportingInterval	A valid cache data reporting interval closest to the requested reporting interval, expressed in seconds. The value shall be zero if there is no reporting interval that is valid for all requested parameters.	Positive integer

10.4 *GetDefinedPlanIds* — Upon receiving this request, the equipment shall return a list of all currently defined data collection plan ids that are accessible by the Client (refer to § 11), the time at which they were defined, and the id of the Client that defined them. If the Client does not have sufficient privilege (refer to § 11), then the equipment shall return an error.

10.4.1 GetDefinedPlanIds Operation Arguments

Table 30 GetDefinedPlanIds Argument Definitions

Argument	Description	Kind	Form
definedPlans	List of defined plans that are visible to the Client.	out	Unordered list of structured data, of type DCPDefined, described in ¶ 10.3.5.

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
error	The Client does not have sufficient privilege to retrieve the list of defined plan ids.	error	Structure, of type UnauthorizedOperation, described in ¶ 10.3.6.

10.4.2 *GetPlanDefinition* — Upon receiving this request, the equipment shall return the requested plan definition to the Client as it was originally submitted to the equipment.

10.4.2.1 If the Client does not have sufficient privileges (refer to § 11) to access the requested DCP definition, the equipment shall return a UnauthorizedOperation error.

10.4.2.2 *GetPlanDefinition Operation Arguments*

Table 31 GetPlanDefinition Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
planId	Identifier for the requested plan.	in	Text, equal to a valid DCP id.
planDefinition	The requested plan definition.	out	Structure, of type DataCollectionPlan, described in ¶ 12.1.
error	The Client submitting the DCP does not have sufficient privilege to retrieve the requested DCP.	error	Structure, of type UnauthorizedOperation, described in ¶ 10.3.6.
error	The equipment does not recognize the requested planId.	error	Structure, of type NoSuchPlan, described in ¶ 10.4.3.

NoSuchPlan
planId

**Figure 11
NoSuchPlan Error**

10.4.3 *NoSuchPlan* — This class describes an error indicating an unrecognized DCP id.

10.4.3.1 *NoSuchPlan Attribute Definition Table*

Table 32 NoSuchPlan Attribute Definition

<i>Attribute Name</i>	<i>Definition</i>	<i>Form</i>
planId	The unrecognized planId.	Text.

10.5 *ActivatePlan* — Upon receiving this request, the equipment shall send a DCPActivation notification to the requesting Client and/or Consumer (if possible) as described in ¶ 16.2.11, and begin sending all DCP events, exceptions, and trace data collection reports to the operational message recipient according to any buffering policies specified in the DCP. The equipment shall place no a-priori limitation on the number of DCPs that a Client can activate. The equipment shall place no a-priori limitation on the number of Clients that can activate the same DCP (refer to § 13.1 Table 72, transition 4). This operation is the only means by which a DCP can be activated.

10.5.1 Any limitations on supportable DCP activity imposed by constraints on available equipment processing power or network interface bandwidth shall be documented by the equipment supplier. If such limitations are encountered at

runtime, the equipment shall use the equipment performance monitoring messages and behavior defined in § 13.4 to manage DCP activity with operational message recipients.

10.5.2 The equipment shall ensure that the requested DCP is valid (refer to ¶ 10.3.7) at the time it is activated. If, at activation time, the requested DCP is not valid, the equipment shall return an InvalidPlan error, and shall not activate the DCP. The equipment shall not prevent DCP activation for any reason (including internal communication failures) if the DCP is valid and the Client has sufficient privilege (refer to § 11). For required behavior in the event of internal communication failures, refer to ¶ 13.2.2.

10.5.3 If the Client does not have sufficient privilege (refer to § 11) to activate the requested DCP, the equipment shall return a UnauthorizedOperation error.

10.5.4 The equipment shall respond to duplicate requests from the same Client to activate the same DCP by returning an error including the DCPActivated information corresponding to that Client's original activation request, but shall take no other actions. The equipment shall not activate a DCP more than once for the same Client.

10.5.5 If the Client attempts to activate a DCP for a session that does send operational messages, the equipment shall return a 'Session does not send operational messages' error. (That is, when the session was established, SEMI E132's EstablishSession operation was called with operationalRecipients parameter is None.)

10.5.6 ActivatePlan Operation Arguments

Table 33 ActivatePlan Argument Definitions

Argument	Description	Kind	Form
planId	Identifier for the requested plan.	in	Text, equal to a valid DCP id.
activatedPlan	Confirmation that the requested plan has been activated.	out	Structured data, of type DCPActivated, described in ¶ 10.8.4.
error	The Client submitting the DCP does not have sufficient privilege to activate the requested DCP.	error	Structured data, of type UnauthorizedOperation, described in ¶ 10.3.6.
error	The equipment does not recognize the requested planId.	error	Structured data, of type NoSuchPlan, described in ¶ 10.4.3.
error	The requested DCP is invalid. This can result from a revision to the metadata.	error	Structured data, of type InvalidPlan, described in ¶ 10.3.7.
error	The Client making the request has previously activated the same DCP.	error	Structured data, of type DCPIsActive, described in ¶ 10.8.3.
error	Session does not send operational messages	error	Text, shall include the text 'Unrecognized equipment identifier' and the bad equipment identifier.

10.6 *GetActivePlanIds* — Upon receiving this request, the equipment shall return a list of the ids of all DCPs that have been activated by the requesting Client at the time of the request. If the Client has sufficient privilege (refer to § 11), the ids of all activation requests for all currently active DCPs shall be returned.

10.6.1 If the Client does not have sufficient privilege (refer to § 11) to retrieve the currently-active DCP ids, the equipment shall return a UnauthorizedOperation error.

10.6.2 GetActivePlanIds Operation Arguments

Table 34 GetActivePlanIds Argument Definitions

Argument	Description	Kind	Form
activePlans	List of active DCP ids.	out	Unordered list of elements of type DCPActivated, described in ¶ 10.8.4. Empty if no DCPs are active. Refer to ¶ 10.5.2 for further explanation.

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
error	The Client submitting the DCP does not have sufficient privilege to retrieve a list of active DCPs.	error	Structured data, of type UnauthorizedOperation, described in ¶ 10.3.6.

10.6.2.1 *activePlans* — The information returned in the response to this operation depends on the privilege assigned to the requesting Client. The equipment shall only return the DCPActivated information for activation requests made by the Client that is invoking the GetActivePlanIds operation (unless the Client has sufficient privilege—refer to ¶ 10.5.2). For example, if some DCPs have been activated by other Clients, but not by the requesting Client, the equipment shall not return the DCPActivated information for those other DCPs. Also, if the requesting Client has activated a DCP, and other Clients have also activated that same DCP, the equipment shall not return the DCPActivated information for those other Clients.

10.6.2.2 For Clients with sufficient privilege only (refer to § 11), the equipment shall return all available DCPActivation information for all Clients of all DCPs that are active at the time of the request.

10.7 *DeactivatePlan* — Upon receiving this request, the equipment shall deactivate the DCP, notify the Client and/or Consumer (if possible) as described in ¶ 16.2.7, and stop sending all event, exception, and trace reports associated with the identified DCP to the operational message recipients. If there are no other operational message recipients receiving data from the identified DCP, the equipment shall disable all collection/buffering activity related to this DCP (refer to Table 72, transition 8). In this case, the equipment shall discard any buffered data at the time of the request.

10.7.1 If the identified DCP has not been previously activated by the requesting Client, and the request is not a termination request (refer to ¶ 10.7.3.2), the equipment shall return the DCPNotActive error. If the request is a termination request, but the requested plan has not been previously activated by any Client, the equipment shall return the DCPNotActive error.

10.7.2 If the Client does not have sufficient privilege (refer to § 10) to deactivate the requested DCP, the equipment shall return a UnauthorizedOperation error.

10.7.3 *DeactivatePlan Operation Arguments*

Table 35 DeactivatePlan Argument Definitions

<i>Argument</i>	<i>Description</i>	<i>Kind</i>	<i>Form</i>
planId	Identifier for the requested plan.	in	Text, equal to a valid DCP id.
terminate	Whether or not the DCP should be deactivated for all Clients.	in	Boolean. Refer to ¶ 10.7.3.1 for more information.
deactivatedPlan	Confirmation that the requested plan has been deactivated	out	Structure, of type DCPDeactivated, described in ¶ 10.7.4.
error	The Client submitting the DCP does not have sufficient privilege to deactivate the DCP	error	Structured data, of type UnauthorizedOperation, described in ¶ 10.3.6.
error	The equipment does not recognize the requested planId	error	Structured data, of type NoSuchPlan, described in ¶ 10.4.3.
error	The requested plan is not active.	error	Structured data, of type DCPNotActive, described in ¶ 10.7.5.

10.7.3.1 *planId* — This operation defines the following reserved planId to name groups of DCPs for deactivation. This identifier is not valid for use as a DCP id in any operation other than DeactivatePlan.

10.7.3.1.1 '*urn:semi-org:dcm:allDCPs*' — This identifier can be used to request deactivation of all plans that are currently active for the requesting Client. The data collection for other Clients is not affected. When used together with a request to terminate (refer to ¶ 10.7.3.2), the equipment shall terminate all DCPs currently active on the