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 *Consumer* — This specification uses the term “consumer” to refer to any software entity that uses the DataCollectionManager interface to manage data collection on the equipment.

6.1.3 *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 consumer, but can not be changed or deleted by any consumer. See Section 11.1.2.1 for details.

6.1.4 *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 a consumer. 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 a consumer. The term “sending” is used to refer to the act of sending a report to a consumer.

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 consumer 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 its consumers, 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 document is abstract in nature, and does not specify or imply any such mechanism. Any adjunct standard that provides an implementation of this specification must include a description of the mechanism used for representing the UML associations shown in this document.

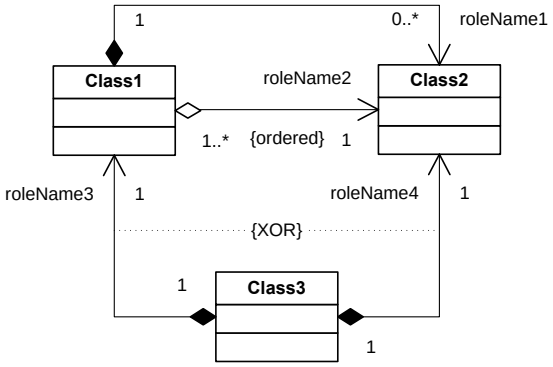

Figure 1
Association Notation

6.2.3 *Association Notation* — This document uses the UML aggregation diamond adornment, role names, end multiplicities, and navigability in all class diagrams specifying associations. Unadorned associations are not used, for economy of notation. All role names are public by default, therefore the visibility symbols are not used. See the UML 1.4 specification, Section 3.43 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. See the UML 1.4 specification, Sections 3.43.2.5 and 3.48 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. See the UML 1.4 specification, Section 3.43.2.2 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. See the UML 1.4 specification, Section 3.42.5.1 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

Association Role Name	*Definition*	*Comments*

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

Attribute Name	*Definition*	*Form*
		See 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 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

Operation	*Description*	*Type*	*Requestor/Sender*	*Responder/Receiver*
		See 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 'consumer' or 'equipment'.

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 'consumer' or 'equipment'.

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

Argument	*Description*	*Kind*	*Form*
		See list below	**See 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 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 Background

7.1 *Equipment Data Collection*

7.1.1 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 consumers to dynamically define the data that is of interest to them (see Figure 2).

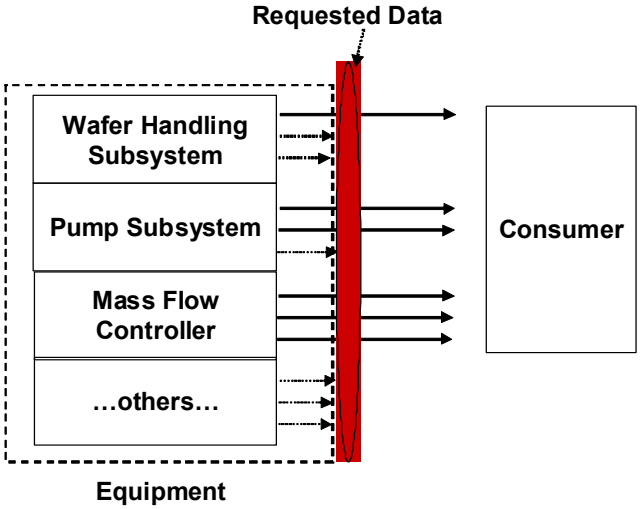

Figure 2
Consumer Specifies What Data Should Be Sent Off-Tool

7.1.2 In addition to declaring interest in specific data that a consumer is interested in, different consumers have different requirements for the acceptable latency with which data should be reported off-tool. For example, a consumer 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 (see Figure 3). A consumer 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 (see Figure 4).

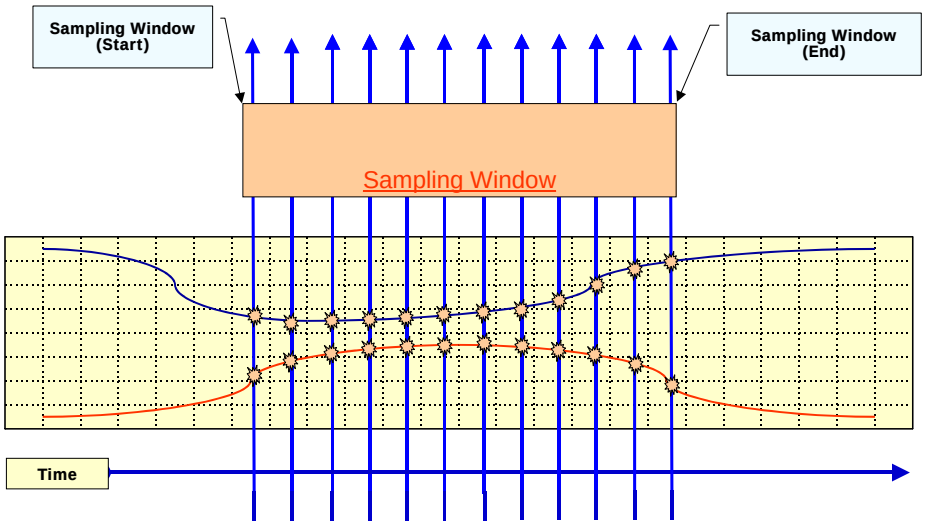

Figure 3
Un-buffered Data Transmission

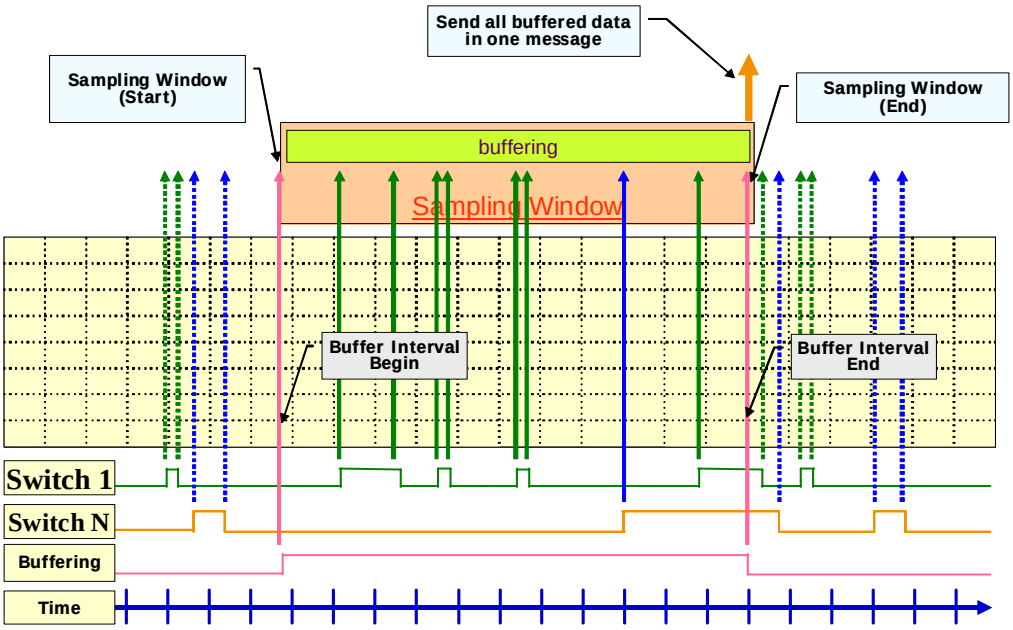

Figure 4
Buffered Data Transmission

7.1.3 Because a single tool is likely to have more than one consumer 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 consumers to define, enable, and disable the transmission of data. An example of this organization is shown in Figure 5.

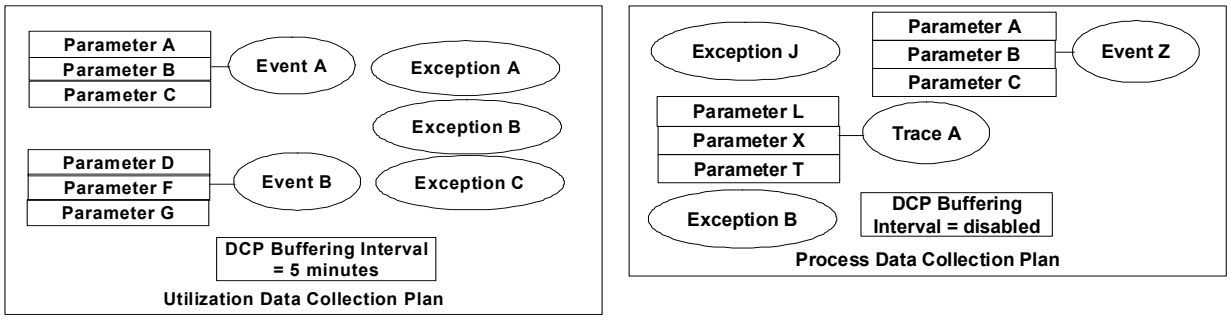

Figure 5
Data Collection Schemes with Different Objectives

7.1.4 It is possible for consumers 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 consumers that a performance threshold is being reached. Consumers 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.

7.1.5 This specification describes the concepts and behaviors necessary to facilitate these capabilities on equipment.

8 Overview

8.1 *Data Collection Concepts*

8.1.1 This specification allows consumers 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 consumer. The types of data that are supported by this specification are described in the following sections.

8.2 *Trace Data*

8.2.1 Trace data allows data acquisition consumers to request the collection of data values that are continuously changing, or that change frequently such as pressure, temperature, RF power, magnetic field strength, or individual actuator state changes, etc. This document 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 (see Section 11.1.5).

8.2.2 This specification defines a mechanism for specifying when to start tracing (see Section 11.1.5.6):

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

8.2.3 This specification defines a mechanism for specifying when to stop tracing (see Section 11.1.5.7):

- 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

8.2.4 This specification defines a mechanism for identifying parameters to be included in trace data collection (see Section 11.1.9). Consumers can use any naming convention supported by the equipment, provided that such conventions can support the concept of a parameter source and a parameter name represented as text fields.

8.2.5 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 (see Sections 11.1.2.3 and 11.1.5.3).

8.3 *Events*

8.3.1 Events allow data acquisition consumers 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 ID's, equipment configuration or offset values. Typical consumers need to be able to specify which variables are to be provided with the event when it is communicated.

8.3.2 This specification defines a mechanism for identifying events and parameters to be included in data collection. Consumers can use any naming convention that is supported by the equipment, provided that such convention can support the concept of event sources and event id's, and parameter sources and parameter names, each represented as text fields (see Sections 11.1.3 and 14.1.3).

8.3.3 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 (see Section 11.1.2.3).

8.4 *Exceptions*

8.4.1 Exceptions allow data acquisition consumers 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.

8.4.2 This specification defines a mechanism for identifying exceptions to be included in data collection. Consumers can use any naming convention that is supported by the equipment, provided that such convention can support the concept of exception sources and exception id's, each represented as text fields (see Sections 11.1.4 and 14.1.4).

8.4.3 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 (see Section 11.1.2.3).

8.5 *Data Collection Plans*

8.5.1 A data collection plan assembles all of the related trace, event, and exception data that may be required for a specific consumer's purpose. For example, a consumer 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.

8.5.2 A fault detection consumer 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 consumers' responsibility, and is not specified in this document.

8.5.3 This document specifies a mechanism for consumers to define one or more data collection plans off-tool and submit those definitions to the equipment (see Sections 9.1.2.2 and 11). This document specifies a mechanism for determining what plans have been created on the equipment, how they are defined, and whether or not they are activated (see Sections 9.1.2.3, 9.1.2.4, and 9.1.2.6). A mechanism for activating, deactivating, and deleting individual plans is also specified (see Sections 9.1.2.5, 9.1.2.7, and 9.1.2.8). The required behavior of data collection plans is specified in Section 12.

8.6 *Equipment Operational Performance Monitoring*

8.6.1 It is possible for consumers 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 document 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 (see Sections 9.1.2.11, 13, 15.1.5, and 15.1.6).

8.7 *Ad-hoc Data Requests*

8.7.1 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.

8.7.2 This document specifies a mechanism for consumers to issue ad-hoc requests for data in order to support these uses, through the GetObjectInstanceIds and GetParameterValue operations of the DataCollectionManager interface (see Sections 9.1.2.9 and 9.1.2.10).

9 Data Collection Plan Management

9.1 *DataCollectionManager Interface*

9.1.1 Figure 6 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.

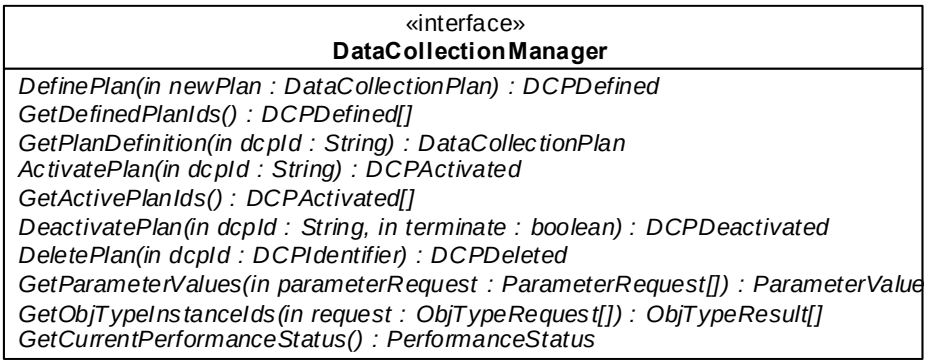

Figure 6
The Data CollectionManager Interface

9.1.2 *DataCollectionManager* — The DataCollectionManager interface provides consumers with operations to manage data collection activities on the equipment. The interface provides operations for consumers 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.

9.1.2.1 *DataCollectionManager Operations*

Table 5 DataCollectionManager Operation Definition

Operation	*Description*	*Type*	*Requestor/Sender*	*Responder/Receiver*
DefinePlan	Submit a new plan definition to the equipment for validation.	RR	Consumer	Equipment
GetDefinedPlanIds	Retrieve a list of all valid DCP id's known to the equipment.	RR	Consumer	Equipment
GetPlanDefinition	Retrieve the definition of a specific DCP.	RR	Consumer	Equipment
ActivatePlan	Activate a defined DCP.	RR	Consumer	Equipment
GetActivePlanIds	Retrieve a list of all currently active DCPs.	RR	Consumer	Equipment
DeactivatePlan	Deactivate an active DCP.	RR	Consumer	Equipment
DeletePlan	Delete a DCP.	RR	Consumer	Equipment
GetParameterValues	Retrieve the most recent values of one or more Parameters.	RR	Consumer	Equipment
GetObjTypeInstanceIds	Retrieve a current list of the unique instance id's for one or more E39 ObjTypes.	RR	Consumer	Equipment
GetCurrentPerformanceStatus	Retrieve the most recent equipment performance status.	RR	Consumer	Equipment

9.1.2.2 *DefinePlan* — Upon receiving this request, the equipment shall first verify that the consumer has sufficient privilege (see Section 10) 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.

9.1.2.2.1 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.

9.1.2.2.2 If there are any issues with the submitted plan, the equipment shall return a description of all problems detected with the plan (see Section 9.1.2.2.6 for a description of possible DCP definition problems), and the plan shall not be defined on the equipment.

9.1.2.2.3 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 consumer that submitted the plan. The equipment shall store this information and the plan definition in non-volatile storage. It shall thereafter be legal for DCP requests to refer to the submitted plan by its id field.

9.1.2.2.4 *DefinePlan Operation Arguments*

Table 6 DefinePlan Argument Definitions

Argument	*Description*	*Kind*	*Form*
newPlan	The definition of a new DCP.	in	Structured data, of type DataCollectionPlan, described in Section 11.1.2 .
planDefined	An acknowledgement that the new DCP definition was well-formed, correct, and supportable.	out	Structured data, of type DCPDefined, described in Section 9.1.2.2.4 .
error	The consumer submitting the DCP does not have sufficient privilege to define DCPs.	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5 .
error	The DCP definition is invalid.	error	Structured data, of type InvalidPlan, described in Section 9.1.2.2.6 .

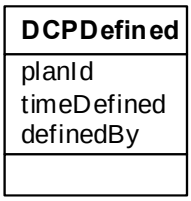

Figure 7
DCP Defined Class

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

9.1.2.2.5.1 *DCPDefined Attribute Definition Table*

Table 7 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.
timeDefined	The date and time at which the DCP was originally defined.	Text, formatted according to Section 14.5 For built-in plans (see Section 11.1.2.1), this field shall be equal to

		the date and time at which the plan was installed on the equipment.
definedBy	The identity of the consumer that defined the DCP.	Text, equal to a valid identifier for the consumer that took the action (see Limitations, Section 3.1.2). For built-in plans (see Section 11.1.2.1), this field shall be a URN of the form "urn:semi-org:equipment". The valid values to use for this attribute to identify consumers shall be specified by any adjunct document defining a technical binding for this specification.

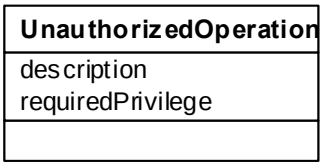

Figure 8
UnauthorizedOperation Error

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

9.1.2.2.6.1 *UnauthorizedOperation Attribute Definition Table*

Table 8 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.
requiredPrivilege	The name of the privilege that is required for the operation	Text, equal to a valid privilege name or identifier (see Section 10) defined on the equipment that the consumer must possess in order to execute the operation. The valid values to use for this attribute shall be specified by any adjunct document defining a technical binding for this specification.

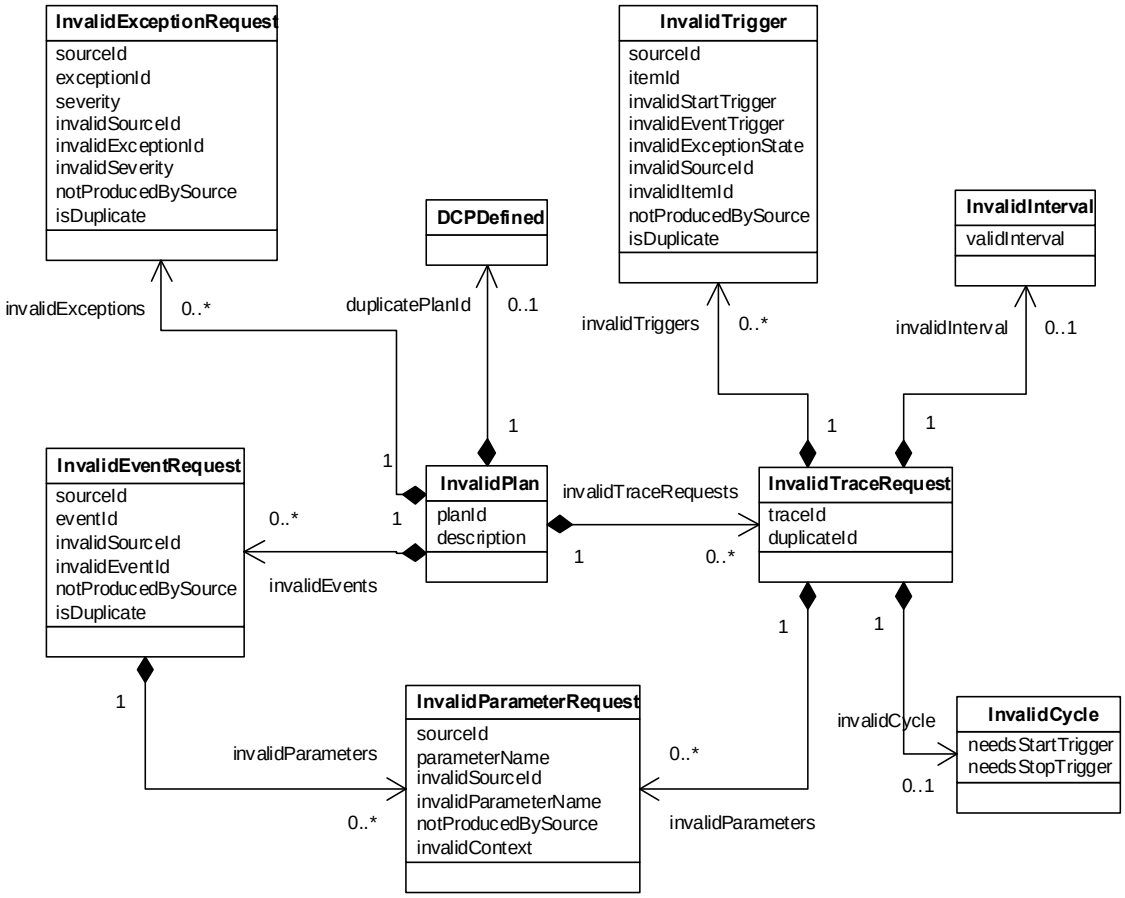

Figure 9
InvalidPlan Error

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

9.1.2.2.7.1 *InvalidPlan Attribute Definition Table*

Table 9 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.

9.1.2.2.7.2 *InvalidPlan Association Definition Table*

Table 10 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 Section 9.1.2.2.7.
invalidExceptions	List of exception requests that have one or more errors.	Unordered list of elements of type InvalidExceptionRequest, described in Section 9.1.2.2.8.
duplicatePlanId	Description of a pre-existing plan that has an identical 'id' attribute.	Structure, of type DCPDefined, described in Section 9.1.2.2.4.
invalidTraceRequests	List of trace requests that have one or more errors.	Unordered list of elements of type InvalidTraceRequest, described in Section 9.1.2.2.10.

9.1.2.2.8 *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 eventId, 3) the sourceId and eventId 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.

9.1.2.2.8.1 *InvalidEventRequest Attribute Definition Table*

Table 11 InvalidEventRequest Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The sourceId from the invalid event request provided in the DCP.	Text
eventId	The eventId 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.
invalidEventId	Whether or not the eventId identifies a valid event supported by the equipment.	Boolean. If true, the event id 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 id's 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.

9.1.2.2.8.2 *InvalidEventRequest Association Definition Table*

Table 12 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 Section 9.1.2.2.9.

9.1.2.2.9 *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 exceptionId, 3) the equipment recognizes both the sourceId and exceptionId, 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 event request within the DCP.

9.1.2.2.9.1 *InvalidExceptionRequest Attribute Definition Table*

Table 13 InvalidExceptionRequest Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The sourceId from the invalid exception request provided in the DCP.	Text.
exceptionId	The eventId from the invalid exception request provided in the DCP.	Text.
severity	The severity from the invalid exception 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.
invalidExceptionId	Whether or not the exceptionId identifies a valid exception supported by the equipment.	Boolean. If true, the exception id 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 source and exception id's 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.

9.1.2.2.10 *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 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 request.

9.1.2.2.10.1 *InvalidParameterRequest Attribute Definition Table*

Table 14 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.

9.1.2.2.11 *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.

9.1.2.2.11.1 *InvalidTraceRequest Attribute Definition Table*

Table 15 InvalidTraceRequest Attribute Definition

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

9.1.2.2.11.2 *InvalidTraceRequest Association Definition Table*

Table 16 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 Section 9.1.2.2.9.
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 Section 9.1.2.2.11.
invalidInterval	Indicates an unsupportable trace interval.	Zero or one element of type InvalidInterval, described in Section 9.1.2.2.12.
invalidCycle	Indicates that the trace request contains an invalid cycle.	Zero or one element of type InvalidCycle, described in Section 9.1.2.2.13.

9.1.2.2.12 *InvalidTrigger* — This class describes errors detected in requested start/stop triggers for traces 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 eventId or exceptionId, 3) the sourceId and eventId or exceptionId 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 request.

9.1.2.2.12.1 *InvalidTrigger Attribute Definition Table*

Table 17 InvalidTrigger Attribute Definition

Attribute Name	*Definition*	*Form*
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.
itemId	The event or exception id 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.

Attribute Name	*Definition*	*Form*
invalidItemId	Whether or not the eventId or exceptionId 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.

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

9.1.2.2.13.1 *InvalidInterval Attribute Definition Table*

Table 18 InvalidInterval Attribute Definition

Attribute Name	*Definition*	*Form*
validInterval	A valid trace interval closest to the requested interval, expressed in seconds.	Floating point.

9.1.2.2.14 *InvalidCycle* — this class describes errors detected in a cyclical trace request in a submitted plan definition. A trace cycle is invalid if the TraceRequest 'isCyclical' attribute (see Section 11.1.5) is true, but does not include both a start and stop trigger.

9.1.2.2.14.1 *InvalidCycle Attribute Definition Table*

Table 19 InvalidCycle Attribute Definition

Attribute Name	*Definition*	*Form*
needsStartTrigger	Whether or not the TraceRequest is missing a start trigger.	Boolean.
needsStopTrigger	Whether or not the TraceRequest is missing a stop trigger.	Boolean.

9.1.2.3 *GetDefinedPlanIds* — upon receiving this request, the equipment shall return a list of all currently defined data collection plan id's that are accessible by the consumer (see Section 10), the time at which they were defined, and the id of the consumer that defined them. If the consumer does not have sufficient privilege (see Section 10), then the equipment shall return an error.

9.1.2.3.1 *GetDefinedPlanIds Operation Arguments*

Table 20 GetDefinedPlanIds Argument Definitions

Argument	*Description*	*Kind*	*Form*
definedPlans	List of defined plans that are visible to the consumer.	out	Unordered list of structured data, of type DCPDefined, described in Section 9.1.2.2.4.
error	The consumer does not have sufficient privilege to retrieve the list of defined plan id's.	error	Structure, of type UnauthorizedOperation, described in Section 9.1.2.2.5.

9.1.2.4 *GetPlanDefinition* — upon receiving this request, the equipment shall return the requested plan definition to the consumer as it was originally submitted to the equipment.

9.1.2.4.1 If the consumer does not have sufficient privileges (see Section 10) to access the requested DCP definition, the equipment shall return an UnauthorizedOperation error.

9.1.2.4.2 *GetPlanDefinition Operation Arguments*

Table 21 GetPlanDefinition Argument Definitions

Argument	*Description*	*Kind*	*Form*
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 Section 11.1.
error	The consumer submitting the DCP does not have sufficient privilege to retrieve the requested DCP.	error	Structure, of type UnauthorizedOperation, described in Section 9.1.2.2.5.
error	The equipment does not recognize the requested planId.	error	Structure, of type NoSuchPlan, described in Section 9.1.2.4.3.

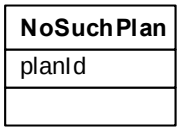

Figure 10
NoSuchPlan Error

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

9.1.2.4.3.1 *NoSuchPlan Attribute Definition Table*

Table 22 NoSuchPlan Attribute Definition

Attribute Name	*Definition*	*Form*
planId	The unrecognized planId.	Text.

9.1.2.5 *ActivatePlan* — Upon receiving this request, the equipment shall begin sending all DCP events, exceptions, and trace data collection reports to the consumer according to any buffering policies specified in the DCP. The equipment shall place no a-priori limitation on the number of DCPs that a consumer can activate. The equipment shall place no a-priori limitation on the number of consumers that can activate the same DCP (see Section 12.1 Table 45, transition 4). This operation is the only means by which a DCP can be activated.

9.1.2.5.1 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 operational performance monitoring messages and behavior defined in Section 13 to manage DCP activity with consumers.

9.1.2.5.2 The equipment shall ensure that the requested DCP is valid (see Section 9.1.2.2.6) 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 consumer has sufficient privilege (see Section 10). For required behavior in the event of internal communication failures, see Section 12.2.2.

9.1.2.5.3 If the consumer does not have sufficient privilege (see Section 10) to activate the requested DCP, the equipment shall return an UnauthorizedOperation error.

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

9.1.2.5.5 *ActivatePlan Operation Arguments*

Table 23 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 Section 9.1.2.8.4.
error	The consumer submitting the DCP does not have sufficient privilege to activate the requested DCP.	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5.
error	The equipment does not recognize the requested planId.	error	Structured data, of type NoSuchPlan, described in Section 9.1.2.4.3.
error	The requested DCP is invalid.	error	Structured data, of type InvalidPlan, described in Section 9.1.2.2.6.
error	The consumer making the request has previously activated the same DCP.	error	Structured data, of type DCPIsActive, described in Section 9.1.2.8.3.

9.1.2.6 *GetActivePlanIds* — Upon receiving this request, the equipment shall return a list of the id's of all DCPs that have been activated by the requesting consumer at the time of the request. If the consumer has sufficient privilege (see Section 10), the id's of all activation requests for all currently active DCPs shall be returned.

9.1.2.6.1 If the consumer does not have sufficient privilege (see Section 10) to retrieve the currently-active DCP id's, the equipment shall return an UnauthorizedOperation error.

9.1.2.6.2 *GetActivePlanIds Operation Arguments*

Table 24 GetActivePlanIds Argument Definitions

Argument	*Description*	*Kind*	*Form*
activePlans	List of active DCP id's.	out	Unordered list of elements of type DCPActivated, described in Section 9.1.2.8.4. Empty if no DCPs are active. See Section 9.1.2.6.3 for further explanation.
error	The consumer submitting the DCP does not have sufficient privilege to retrieve a list of active DCPs.	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5.

9.1.2.6.3 *activePlans* — The information returned in the response to this operation depends on the privilege assigned to the requesting consumer. The equipment shall only return the DCPActivated information for activation requests made by the consumer that is invoking the GetActivePlanIds operation (unless the consumer has sufficient privilege – see Section 9.1.2.6.3.1). For example, if some DCPs have been activated by other consumers, but not by the requesting consumer, the equipment shall not return the DCPActivated information for those other DCPs. Also, if the requesting consumer has activated a DCP, and other consumers have also activated that same DCP, the equipment shall not return the DCPActivated information for those other consumers.

9.1.2.6.3.1 For consumers with sufficient privilege only (see Section 10), the equipment shall return all available DCPActivation information for all consumers of all DCPs that are active at the time of the request.

9.1.2.7 *DeactivatePlan* — Upon receiving this request, the equipment shall stop sending of all event, exception, and trace requests associated with the identified DCP to the requesting consumer. If there are no other consumers receiving data from the identified DCP, the equipment shall disable all collection/buffering activity related to this DCP (see Table 45, transition 8). In this case, the equipment shall discard any buffered data at the time of the request.

9.1.2.7.1 If the identified DCP has not been previously activated by the requesting consumer, and the request is not a termination request (see Section 9.1.2.7.5), 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 consumer, the equipment shall return the DCPNotActive error.

9.1.2.7.2 If the consumer does not have sufficient privilege (see Section 10) to deactivate the requested DCP, the equipment shall return an UnauthorizedOperation error.

9.1.2.7.3 *DeactivatePlan Operation Arguments*

Table 25 DeactivatePlan Argument Definitions

Argument	*Description*	*Kind*	*Form*
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 consumers.	in	Boolean. See Section 9.1.2.7.4 for more information.
deactivatedPlan	Confirmation that the requested plan has been deactivated	out	Structure, of type DCPDeactivated, described in Section 9.1.2.7.6.
error	The consumer submitting the DCP does not have sufficient privilege to deactivate the DCP	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5.
error	The equipment does not recognize the requested planId	error	Structured data, of type NoSuchPlan, described in Section 9.1.2.4.3.
error	The requested plan is not active.	error	Structured data, of type DCPNotActive, described in Section 9.1.2.7.7.

9.1.2.7.4 *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.

9.1.2.7.4.1 *"urn:semi-org:dcm:allDCPs"* — This identifier can be used to request deactivation of all plans that are currently active for the requesting consumer. If other consumers have activated any of these same plans, those consumers shall continue to receive DataCollectionReports from those plans. When used together with a request to terminate (see Section 9.1.2.7.5), the equipment shall terminate all DCPs currently active on the equipment for all consumers, including those that have been activated by the requesting consumer.

9.1.2.7.5 *terminate* — If set to "true", the equipment shall stop buffering/sending all data included in the identified DCP to all consumers that have previously activated the DCP. The equipment shall discard any buffered data at the time of the request. The equipment shall notify each such consumer of the reason for the termination using the DCPDeactivated notification (see Section 15.1.6.2). The consumer must have sufficient privilege (see Section 10) to request termination of a DCP.

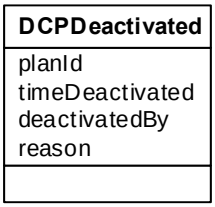

Figure 11
DCPDeactivated Class

9.1.2.7.6 *DCPDeactivated* — This class describes a DCP that has been successfully deactivated on the equipment.

9.1.2.7.6.1 *DCPDeactivated Attribute Definition Table*

Table 26 DCPDeactivated Attribute Definition

Attribute Name	*Definition*	*Form*
planId	Identifies a plan that has been deactivated.	Text, equal to the 'id' attribute of the DataCollectionPlan that was deactivated.
timeDeactivated	The date and time at which the DCP was deactivated.	Text, formatted according to Section 14.5.
deactivatedBy	The identity of the consumer that deactivated the DCP.	Text, equal to a valid identifier for the consumer that took the action (see Limitations, Section 3.1.2). If the equipment is deactivating the DCP, this field shall be a URN of the form "urn:semi-org:equipment". The valid values to use for this attribute to identify consumers shall be specified by any adjunct document defining a technical binding for this specification.
reason	A human-readable description of the reason that the DCP was deactivated.	Text.

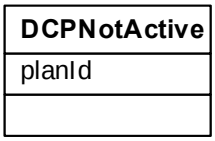

Figure 12
DCPNotActive Class

9.1.2.7.7 *DCPNotActive* — This class describes a DCP that has not been activated, either by a specific consumer, or by any consumer (see Section 9.1.2.7.1).

9.1.2.7.7.1 *DCPNotActive Attribute Definition Table*

Table 27 DCPNotActive Attribute Definition

Attribute Name	*Definition*	*Form*
planId	Identifies the plan that is not active.	Text, equal to the 'id' attribute of the DataCollectionPlan that is not active.

9.1.2.8 *DeletePlan* — Upon receiving this request, the equipment shall delete the requested plan definition from non-volatile storage. The equipment shall reject this request if the requested plan is currently active for one or more consumers. For a plan to be deleted, it must be deactivated (no consumer is currently receiving data produced by the plan). Note that no consumer can delete built-in DCPs provided with the equipment (see Section 11.1.2.1).

9.1.2.8.1 If the consumer does not have sufficient privilege (see Section 10) to delete the requested DCP, the equipment shall return an UnauthorizedOperation error.

9.1.2.8.2 *DeletePlan Operation Arguments*

Table 28 DeletePlan Argument Definitions

Argument	*Description*	*Kind*	*Form*
planId	Identifier for the requested plan.	in	Text, equal to a valid DCP id.
deletedPlan	Confirmation that the requested plan has been deleted.	out	Structured data, of type DCPDeleted, described in Section 9.1.2.8.5.

Argument	*Description*	*Kind*	*Form*
error	The consumer submitting the DCP does not have sufficient privilege to delete the requested DCP.	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5 . If the consumer was attempting to delete a built-in DCP (see Section 11.1.2.1), the requiredPrivilege attribute of the UnauthorizedOperation class shall be set to the value “no such privilege”.
error	The equipment does not recognize the requested planId.	error	Structured data, of type NoSuchPlan, described in Section 9.1.2.4.3.
error	The requested planId is currently active	error	Structured data, of type DCPIsActive (see Section 9.1.2.8.3).

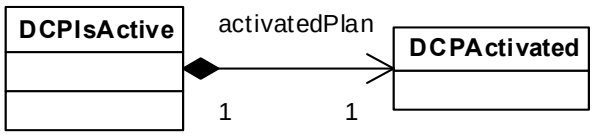

Figure 13
DCPIsActive Error

9.1.2.8.3 *DCPIsActive* — This class describes an error in activating or deleting a DCP that is currently active. It provides information describing the original activation of the DCP.

9.1.2.8.3.1 *DCPIsActive Association Definition Table*

Table 29 DCPIsActive Association Definition

Association Role Name	*Definition*	*Comments*
activatedPlan	Provides a description of the original activation of the DCP.	Structured data, of type DCPActivated, described in Section 9.1.2.8.4.

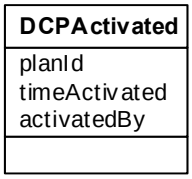

Figure 14
DCPActivated Class

9.1.2.8.4 *DCPActivated* — This class describes a DCP that has been successfully activated on the equipment.

9.1.2.8.4.1 *DCPActivated Attribute Definition Table*

Table 30 DCPActivated Attribute Definition

Attribute Name	*Definition*	*Form*
planId	Identifies an active DCP.	Text, equal to the ‘id’ attribute of the DataCollectionPlan that has been activated.
timeActivated	The date and time at which the DCP was activated.	Text, formatted according to Section 14.5.
activatedBy	The identity of the consumer that activated the DCP.	Text, equal to a valid identifier for the consumer that took the action (see Limitations, Section 3.1.2). The valid values to use for this attribute to identify

		consumers shall be specified by any adjunct document defining a technical binding for this specification.

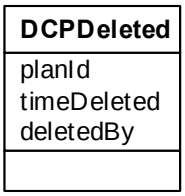

Figure 15
DCPDeleted Class

9.1.2.8.5 *DCPDeleted* — This class describes a DCP that has been deleted from the equipment. It is used only in the response to the DeletePlan operation. The equipment is not required to retain a history of deleted DCPs.

9.1.2.8.5.1 *DCPDeleted Attribute Definition Table*

Table 31 DCPDeleted Attribute Definition

Attribute Name	*Definition*	*Form*
planId	Identifies the DCP that was deleted.	Text, equal to the 'id' attribute of the DataCollectionPlan that was deleted.
timeDeleted	The date and time at which the DCP was deleted.	Text, formatted according to Section 14.5.
deletedBy	The identity of the consumer that deleted the DCP.	Text, equal to a valid identifier for the consumer that deleted the DCP (see Limitations, Section 3.1.2). The valid values to use for this attribute to identify consumers shall be specified by any adjunct document defining a technical binding for this specification.

9.1.2.9 *GetParameterValues* — This operation can be used by consumers to determine the current state of the equipment. It is not intended to be used as a substitute for data collection through DCPs. Upon receiving this request, the equipment shall return the most current values for the requested Parameters known to the equipment at the time of the request. The equipment is not required to actively retrieve new or update internally stored values at the time of the request. Consumers should be aware that the time at which the equipment last obtained or updated the requested Parameter values may be earlier than the time at which the request is made. If some values cannot be obtained at the time of the request (perhaps due to an internal equipment failure), or if the request refers to Parameters that are not defined, the equipment shall indicate so by returning NoValue (see Section 14.2.1.4) for each such Parameter.

9.1.2.9.1 If the consumer does not have sufficient privilege (see Section 10) to request Parameter values, the equipment shall return an UnauthorizedOperation error.

9.1.2.9.2 *GetParameterValues Operation Arguments*

Table 32 GetParameterValues Argument Definitions

Argument	*Description*	*Kind*	*Form*
parameterRequest	The Parameters whose values are being requested.	in	List of elements of type ParameterRequest, described in Section 11.1.9.
parameterValues	The values of the requested parameters.	out	List of elements of type ParameterValue, described in Section 14.2.
error	The consumer does not have sufficient privilege to request Parameter values.	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5.

9.1.2.10 *GetObjTypeInstanceIds* — This operation provides consumers with information required to request the values of attributes of specific E39-compliant objects that have been instantiated on the equipment. A consumer

interested in the attributes of specific E39 objects can only make requests from a specific object instance if the unique instance identification information for such objects is known. This operation allows consumers to learn from the equipment how many E39-compliant objects of a given ObjType are instantiated on the equipment at the time of the request, and what their unique instance identification information is. The consumer may then use that instance information to request E39-compliant object attribute values using either the GetParameterValues operation (see Section 9.1.2.9), or by including this information in a DCP. E39-compliant object attribute values are returned as ParameterValues (see Section 14.2).

9.1.2.10.1 This specification does not define the format of an E39-compliant object instance identifier; this format shall be fully specified by any implementation mapping of this specification.

9.1.2.10.2 Upon receiving this request, the equipment shall return a list of the instance identifiers of all SEMI E39-compliant objects that exist at the time of the request, and are of the requested ObjType, and that are provided by the requested source. If the request includes unrecognized ObjTypes or sources, the equipment shall indicate so by returning ObjTypeRequestError (see Section 9.1.2.10.11) for each such request. Consumers can request the list of all instances, all instances of a specific type, all instances from a specific source, or all instances of a specific type from a specific source.

9.1.2.10.3 If the consumer does not have sufficient privilege (see Section 10) to request ObjType instance id's, the equipment shall return an UnauthorizedOperation error.

9.1.2.10.4 *GetObjTypeInstanceIds Operation Arguments*

Table 33 GetObjTypeInstanceIds Argument Definitions

Argument	*Description*	*Kind*	*Form*
request	The ObjTypes whose instance id's are being requested.	in	List of elements of type ObjTypeRequest, described in Section 9.1.2.10.6.
objTypeInstances	The instance id's of the requested ObjTypes.	out	List of elements of type ObjTypeResult, described in Section 9.1.2.10.9 . See Section 9.1.2.10.5 for further description.
error	The consumer does not have sufficient privilege to request ObjType instance id's.	error	Structured data, of type UnauthorizedOperation, described in Section 9.1.2.2.5.

9.1.2.10.5 *objTypeInstances* — The equipment shall return one and only one element of type ObjTypeResult for each unique ObjType-source combination resulting from the original request. For example, if the consumer requested all "Carrier" ObjType instance id's, regardless of the source, and the equipment has only one source of that ObjType, then only one ObjTypeResult will be returned. That result will contain all instance id's for that ObjType provided by that source. If there were two or more sources of that ObjType, the equipment would return as many ObjTypeResult elements as there were sources, each element containing all instance id's of that ObjType provided by that source.

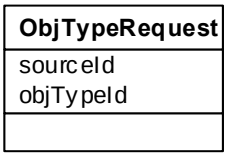

Figure 16
ObjTypeRequest Class

9.1.2.10.6 *ObjTypeRequest* — This class represents a request for the instance id's of one or more ObjTypes.

9.1.2.10.6.1 *ObjTypeRequest Attribute Definition Table*

Table 34 ObjTypeRequestError Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	Identifies the requested source of the requested ObjType.	Text, formatted according to the convention used to identify the source of an ObjType. See limitations, Section 3.1.1. See Sections 9.1.2.10.7 and 9.1.2.10.8 for further description.
objTypeId	Identifies the ObjType whose instance id's are being requested.	Text. See Sections 9.1.2.10.7 and 9.1.2.10.8 for further description.

9.1.2.10.7 If the 'sourceId' attribute is non-empty, the equipment shall return all instance id's of the ObjType specified in the 'objTypeId' attribute that are available from the specified source. If the 'objTypeId' attribute is empty, the equipment shall return the id's of all E39-compliant ObjType instances available from the specified source, regardless of their ObjType.

9.1.2.10.8 If the 'sourceId' attribute is empty, the equipment shall return all instance id's of the ObjType specified in the objTypeId attribute, regardless of their source. If the objTypeId attribute is also empty, the equipment shall return all instances of all E39-compliant ObjTypes, regardless of their source.

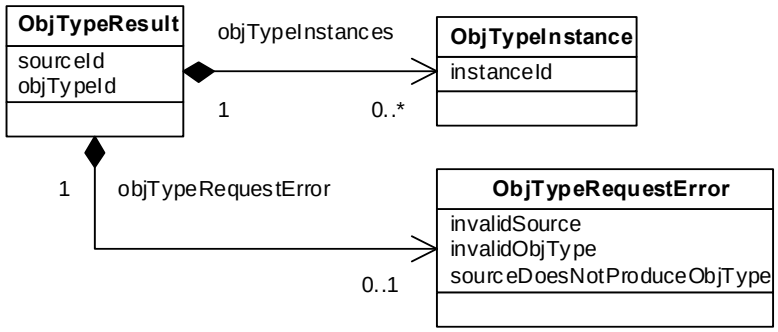

Figure 17
ObjTypeResult and Related Classes

9.1.2.10.9 *ObjTypeResult* — This class provides a list of requested ObjType instance id's. If the original request included unrecognized sources or ObjTypes, or ObjTypes that are not provided by the requested sources, the equipment shall return an ObjTypeRequestError (see Section 9.1.2.10.11) for each such request.

9.1.2.10.9.1 *ObjTypeResult Attribute Definition Table*

Table 35 ObjTypeResult Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	Identifies the source of the ObjType instance.	Text, formatted according to the convention used to identify the source of an E39-compliant ObjType. See limitations, Section 3.1.1. If the original request contained an invalid source or ObjType, this field must be equal to the sourceId provided with the corresponding original ObjTypeRequest.
objTypeId	The ObjType whose instances are provided with this result.	Text, equal to the ObjType name. If the original request contained an invalid source or ObjType, this field must be equal to the objTypeId provided with the original corresponding ObjTypeRequest.

9.1.2.10.9.2 *ObjTypeResult Association Definition Table*

Table 36 ObjTypeResult Association Definition

Association Role Name	*Definition*	*Comments*
objTypeInstances	The list of ObjType instance id's for the corresponding ObjTypeResult.	List of zero or more elements of type ObjTypeInstance, described in Section 9.1.2.10.10.
objTypeError	A description of a problem detected in the original ObjTypeRequest.	Zero or one element of type ObjTypeRequestError, described in Section 9.1.2.10.11.

9.1.2.10.10 *ObjTypeInstance* — This class, shown in Figure 23, provides the instance id of a requested ObjType from its requested source.

9.1.2.10.10.1 *ObjTypeInstance Attribute Definition Table*

Table 37 ObjTypeInstance Attribute Definition

Attribute Name	*Definition*	*Form*
instanceId	The instance id of an E39-compliant object that has the source and objTypeId of the ObjTypeResult to which this ObjTypeInstance belongs.	Text, format to be specified in implementation mappings.

9.1.2.10.11 *ObjTypeRequestError* — This class, shown in Figure 23, describes an error detected in a specific ObjTypeRequest provided as input to the GetObjTypeInstanceIds operation.

9.1.2.10.11.1 *ObjTypeRequestError Attribute Definition Table*

Table 38 ObjTypeRequest Attribute Definition

Attribute Name	*Definition*	*Form*
invalidSource	Whether or not the corresponding ObjTypeRequest specified an unrecognized source.	Boolean
invalidObjType	Whether or not the corresponding ObjTypeRequest specified an unrecognized ObjType.	Boolean
sourceDoesNotProduceObjType	Whether or not the corresponding ObjTypeRequest specified a source that does not provide the requested ObjType.	Boolean

9.1.2.11 *GetCurrentPerformanceStatus* — Upon receiving this request, the equipment shall return the most recently evaluated equipment performance status (see Section 15.1.6.1.1).

9.1.2.11.1 If the consumer does not have sufficient privilege (see Section 10) to request the current performance status, the equipment shall return an UnauthorizedOperation error.

9.1.2.11.2 *GetCurrentPerformanceStatus Operation Arguments*

Table 39 GetCurrentPerformanceStatus Argument Definitions

Argument	*Description*	*Kind*	*Form*
status	The most recent operational performance status.	out	One and only one element of type PerformanceStatus, described in Section 15.1.6.1.1.

10 DCP Privilege Model

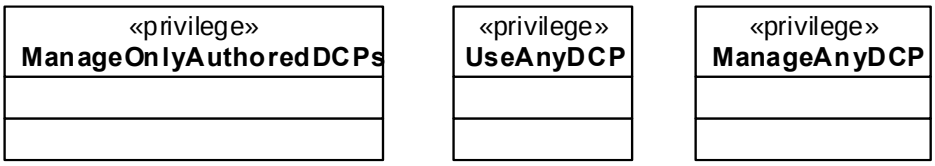

Figure 18
Data Collection Privileges

10.1 *Privileges*

10.1.1 Figure 18 shows the privileges that shall be enforced when processing requests from DCP consumers. Note that suppliers may provide more fine-grained privileges than shown here, if desired. If a consumer attempts to execute an operation on the DataCollectionManager interface that requires any of these privileges, and such privilege has not been assigned to the consumer, the equipment shall not execute the request, and shall instead return the UnauthorizedOperation error described in Section 9.1.2.2.5. The meaning of each privilege is provided in Table 40.

10.1.2 Definition of the mechanism for assigning and enforcing privileges is beyond the scope of this specification (see limitations, Section 3.1.2). Such mechanisms shall be specified completely in any implementation mapping for this specification.

10.1.3 *Data Collection Management Privilege Definition Table*

Table 40 Data Collection Management Privilege Definition

Operation	*No Privilege*	*ManageOnly-AuthoredDCPs*	*UseAnyDCP*	*ManageAny-DCP*
DefinePlan	Equipment shall reject request with Unauthorized-Operation.	No restriction.	No restriction.	No restriction.
GetDefinedPlanIds	Equipment shall reject request with Unauthorized-Operation.	Equipment shall return only the id's of plans defined by the requesting consumer and built-in plans.	Equipment shall return the id's of all plans defined on the equipment, including built-in plans and plans defined by other consumers.	Equipment shall return the id's of all plans defined on the equipment, including built-in plans and plans defined by other consumers.
GetPlanDefinition	Equipment shall reject request with Unauthorized-Operation.	Equipment shall accept only requests for plans defined by the requesting consumer, and built-in plans. Requests for all other plans shall be rejected with Unauthorized-Operation.	Equipment shall accept requests for any plan defined by any consumer, including built-in plans and plans defined by other consumers.	Equipment shall accept requests for any plan defined by any consumer, including built-in plans and plans defined by other consumers.

Operation	*No Privilege*	*ManageOnly-AuthoredDCPs*	*UseAnyDCP*	*ManageAny-DCP*
ActivatePlan	Equipment shall reject request with Unauthorized-Operation.	Equipment shall accept only requests for plans defined by the requesting consumer, and built-in plans. Requests for all other plans shall be rejected with Unauthorized-Operation.	Equipment shall accept requests for any plan defined on the equipment, including built-in plans and plans defined by other consumers.	Equipment shall accept requests for any plan defined on the equipment, including built-in plans and plans defined by other consumers.
GetActivePlanIds	Equipment shall reject request with Unauthorized-Operation.	Equipment shall return only the DCPActivated information for plans activated by the requesting consumer (see Section 9.1.2.8.4). If a plan activated by the requesting consumer is also active for other consumers, the equipment shall not return the DCPActivated information for those other consumers.	Equipment shall return only the DCPActivated information for plans activated by the requesting consumer (see Section 9.1.2.8.4). If a plan activated by the requesting consumer is also active for other consumers, the equipment shall not return the DCPActivated information for those other consumers.	Equipment shall return all DCPActivated information for all plans activated by all consumers (see Section 9.1.2.8.4).
DeactivatePlan	Equipment shall reject request with Unauthorized-Operation.	Equipment shall accept only requests to deactivate plans activated by the requesting consumer. Requests for all other plans shall be rejected with Unauthorized-Operation.	Equipment shall accept only requests to deactivate plans activated by the requesting consumer. Requests for all other plans shall be rejected with Unauthorized-Operation.	Equipment shall accept requests to deactivate any plan(s) activated by any consumer(s).
DeletePlan	Equipment shall reject request with Unauthorized-Operation.	Equipment shall accept only requests to delete plans defined by the requesting consumer. Requests for all other plans, including built-in plans, shall be rejected with Unauthorized-Operation.	Equipment shall accept only requests to delete plans defined by the requesting consumer. Requests for all other plans, including built-in plans, shall be rejected with Unauthorized-Operation.	Equipment shall accept requests to delete any plan defined by any consumer. Equipment shall reject requests to delete built-in plans with Unauthorized-Operation.
GetParameterValues	Equipment shall reject request with Unauthorized-Operation.	No restriction.	No restriction.	No restriction.
GetObjTypeInstanceIds	Equipment shall reject request with Unauthorized-Operation.	No restriction.	No restriction.	No restriction.

Operation	*No Privilege*	*ManageOnly-AuthoredDCPs*	*UseAnyDCP*	*ManageAny-DCP*
GetCurrentPerformanceStatus	Equipment shall reject request with Unauthorized-Operation.	No restriction.	No restriction.	No restriction.

11 Data Collection Plan Definition

11.1 *DataCollectionPlan*

11.1.1 Figure 19 illustrates how a data acquisition consumer can specify to the equipment what data should be reported for a specific plan.

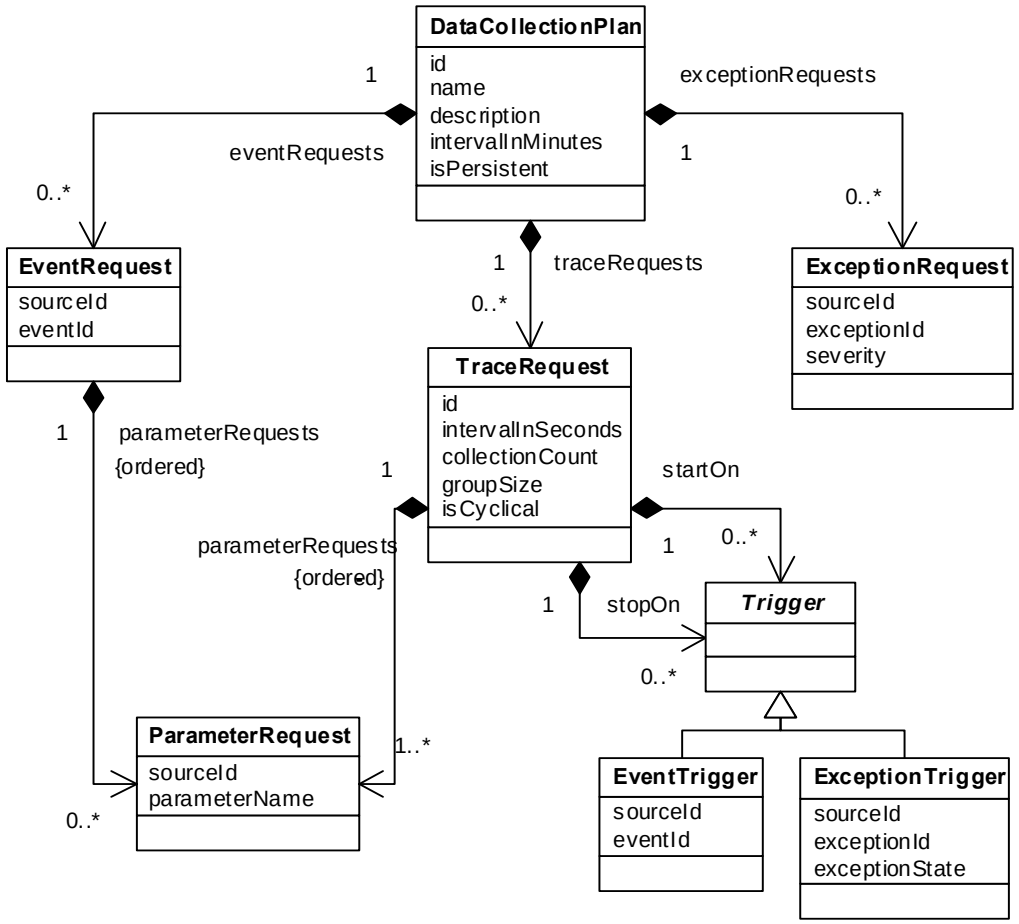

Figure 19
Data Collection Plan

11.1.2 *DataCollectionPlan* — A data collection plan has a consumer-defined (or supplier-defined, in the case of built-in DCPs) name and id, and acts as a container for classes that describe what trace data, exceptions, and events are to be reported to the consumer, as well as any buffering policy for the plan.

11.1.2.1 *Built-in DCPs* — Equipment suppliers may provide pre-defined DCPs that are included with the equipment, and don't require definition by a consumer. Such DCPs shall be defined according to the format specified in this section, and are referred to as "built-in" DCPs. Built-in DCPs can be activated and viewed by any

consumer with sufficient privilege (see Section 10) using the DataCollectionManager's ActivatePlan and GetPlanDefinition operations, respectively. Built-in DCPs cannot be deleted by any consumer.

11.1.2.2 *DataCollectionPlan Attribute Definition Table*

Table 41 DataCollectionPlan Attribute Definition

Attribute Name	*Definition*	*Form*
name	Consumer-defined (or supplier-defined, in the case of built-in DCPs) human-readable name for the DCP.	Text.
id	Consumer-defined (or supplier-defined, in the case of built-in DCPs) unique identifier for the DCP.	Text, unique across all DCP definitions on the equipment, formatted according to the ISO 11578:1996 definition of UUID.
description	Consumer-defined (or supplier-defined, in the case of built-in DCPs) description of the purpose of the DCP.	Text.
intervalInMinutes	The preferred number of minutes for which the equipment shall buffer all data described by the plan.	Positive integer ≥ 0. See Section 11.1.2.3 for further explanation.
isPersistent	Whether or not the DCP, once activated, shall remain activated across equipment shutdowns.	Boolean. See behavior defined in Section 12.1.5.

11.1.2.3 *intervalInMinutes*

11.1.2.3.1 If the value of this field is zero, events and exceptions contained in the plan shall be reported to the consumer when they occur, and trace reports contained in the plan shall be reported according to their individual buffer settings (see the description of the TraceRequest type in Section 11.1.5).

11.1.2.3.2 If the value of this field is greater than zero, upon activation of the DCP, any exceptions, events, or traces defined in the plan that occur during this time span shall be buffered on the equipment in the order that they occur. Buffered data shall be reported at the end of the interval, or when the equipment's buffering capacity for the currently active DCPs is reached, whichever is sooner. After each interval expires, the equipment shall begin buffering new data for the next interval, again sending the results when the interval expires (or buffering capacity is reached). Each result sent shall contain only the data collected during that interval. The equipment shall continue this buffering/sending cycle as long as the DCP is active.

11.1.2.3.3 All data sent is formatted using the DataCollectionReport format described in Section 14.1.1.1.

11.1.2.4 *DataCollectionPlan Association Definition Table*

Table 42 DataCollectionPlan Association Definition

Association Role Name	*Definition*	*Comments*
eventRequests	A set of events that the consumer is requesting to be reported for this plan.	List of structured data, of type EventRequest, described in Section 11.1.3.
exceptionRequests	A set of exceptions that the consumer is requesting to be reported for this plan.	List of structured data of type ExceptionRequest, described in Section 11.1.4.
traceRequests	A set of trace data collection activities that the consumer is requesting to be reported for this plan.	List of structured data of type TraceRequest, described in Section 11.1.5.

11.1.3 *EventRequest* — Identifies an event generated by a specific source that the consumer is requesting to be reported when the DCP is activated. The equipment shall report only those events matching both the source and id attributes specified in the EventRequst. The format for sending an event report is described in Section 14.1. The equipment shall support multiple DCPs requesting the same event, with different sets of requested parameters. It shall be an error to request the same event more than once in the same DCP (see Section 9.1.2.2.7).

11.1.3.1 *EventRequest Attribute Definition Table*

Table 43 EventRequest Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The source of the event.	Text, formatted according to the convention used to identify an event source. See limitations, Section 3.1.1.
eventId	The identifier for the event.	Text, formatted according to the convention used to identify a specific event. See limitations, Section 3.1.1.

11.1.3.2 *EventRequest Association Definition Table*

Table 44 EventRequest Association Definition

Association Role Name	*Definition*	*Comments*
parameterRequests	A set of parameters that the consumer is requesting to be reported when the requested event is detected	Ordered list of structured data, of type ParameterRequest, described in Section 11.1.5.6 . The order of the ParameterRequests appearing in an EventRequest determines the order in which those Parameter values will be reported in the corresponding EventReport (see Section 14.1.3).

11.1.4 *ExceptionRequest* — Identifies an exception or class of exceptions that the consumer is requesting to be reported when the DCP is activated. The format for sending an exception report is described in Section 14.1. The equipment shall support multiple DCPs requesting the same exception criteria. It shall be an error to request the same exception criteria more than once in the same DCP (see Section 9.1.2.2.8).

11.1.4.1 *ExceptionRequest Attribute Definition Table*

Table 45 ExceptionRequest Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The source of the exception.	Text, formatted according to the convention used to identify an exception source. See limitations, Section 3.1.1.
exceptionId	The identifier of the exception.	Text, formatted according to the convention used to uniquely identify an exception. See limitations, Section 3.1.1.
severity	The severity of the exception.	Text, formatted according to the convention used to identify the severity of an exception. Acceptable values, if any, are supplier-defined.

11.1.4.2 *sourceId* — If this is the only non-empty attribute, then all exceptions from this source will be reported, regardless of id or severity.

11.1.4.3 *exceptionId* — If this is the only non-empty attribute, then all exceptions with this id will be reported, regardless of source or severity.

11.1.4.4 *severity* — If this is the only non-empty attribute, then all exceptions with this severity will be reported, regardless of the source or id of the exception. Note that equipment suppliers are not required to define severity classifications for exceptions. If severity classifications are not supported by the equipment supplier, any ExceptionRequest containing a non-empty severity attribute shall be rejected as invalid (see Section 9.1.2.2.8). If the equipment supplier does define severity classifications for exceptions, the ability to request exceptions based on severity in a DCP shall be supported.

11.1.4.5 If more than one of the above attributes is non-empty, then whether or not an exception will be reported will be determined by applying Boolean AND logic to the non-empty attributes. At least one attribute must be non-empty.

11.1.4.6 It is not possible to customize the parameters to be sent when exceptions occur. Equipment suppliers define what parameters, if any, are sent with exceptions.

11.1.4.7 ExceptionRequests may refer to exceptions whose state the equipment tracks (for example, the "alarm set" and "alarm clear" states for E30 alarms). If an ExceptionRequest in a DCP refers to such an exception, the equipment shall send an ExceptionReport (see Section 14.1.4) each time there is a change in the exception's state. Different data may be reported for each different exception state. If an Exception does not have any associated state, it is not necessary to report any description of the state of the exception (see Section 14.1.4.1.2).

11.1.5 *TraceRequest* — This class identifies a set of parameters whose values are to be periodically collected and reported to the consumer using the format defined in Section 14.1 . The class is used to describe at what rate the set of requested parameters should be collected, if/how collected trace data should be buffered on the tool and when the results should be sent to the consumer, and when the trace should start and stop.

11.1.5.1 *TraceRequest Attribute Definition Table*

Table 46 TraceRequest Attribute Definition

Attribute Name	*Definition*	*Form*
id	Unique identifier for this trace request.	Integer, unique across all trace requests belonging to this DCP.
intervalInSeconds	The period of time in seconds between each collection.	Real number. Numbers less than 1.0 specify sub-second intervals. See Section 9.1.2.2.12 for possible errors.
collectionCount	The total number of collection results that should be obtained before terminating the trace.	Positive integer ≥ 0. See Section 11.1.5.2 for further explanation.
groupSize	The preferred number of collection results to obtain before sending to the consumer.	Integer ≥ 0. See Section 11.1.5.3 for further explanation.
isCyclical	Whether or not the trace should re-trigger after its stop trigger has occurred.	Boolean. True indicates that the trace is cyclical. See Section 11.1.5.4 for further explanation.

11.1.5.2 *collectionCount*

11.1.5.2.1 A value of 0 for this attribute indicates that the trace shall continue indefinitely, until a stop trigger is detected, or until the DCP is deactivated. If > 0, the trace shall continue until a stop trigger (see Section 11.1.5.7) is detected or until the total number of collection results obtained is equal to collectionCount, whichever occurs first. For example, the consumer may wish to provide both a stop trigger and a non-zero collectionCount in order to limit the total number of collection results to report regardless of whether or not the stop trigger occurs, or in case an error condition on the equipment prevents the stop trigger from firing.

11.1.5.3 *groupSize*

11.1.5.3.1 Depending on the buffering capacity of the equipment, the DCP may send a TraceReport before 'groupSize' results have been obtained. A TraceReport must not include more than 'groupSize' collection results (excluding groupSize = 0). See Section 12.3.13 for a description of the effect of this setting on DCP behavior.

11.1.5.4 *isCyclical*

11.1.5.4.1 At least one start and one stop trigger must be supplied for a trace request to be cyclical. If this attribute has the value 'true', the trace shall be restarted if any one of the start triggers is detected after termination. If this attribute has the value 'false', the trace will not restart unless the DCP is deactivated and then re-activated. See Section 9.1.2.2.13 for possible errors.

11.1.5.5 *TraceRequest Association Definition Table*

Table 47 TraceRequest Association Definition

Association Role Name	*Definition*	*Comments*
parameterRequests	A set of parameters whose values that the consumer is requesting to be collected at the interval specified in the TraceRequest.	Ordered list of structured data, of type ParameterRequest, described in Section 11.1.9 . The order of the ParameterRequests appearing in a TraceRequest determines the order in which those Parameter values will be reported in the corresponding TraceReport (see Section 14.1.5).
startOn	A list of zero or more events and/or exceptions that can initiate the trace.	List of structured data of any type subclassed from Trigger. See Section 11.1.5.6 for further explanation.
stopOn	A list of zero or more events and/or exceptions that can stop the trace.	List of structured data of any type subclassed from Trigger. See Section 11.1.5.7 for further explanation.

11.1.5.6 *startOn*

11.1.5.6.1 If no triggers are provided, the trace initiates when the DCP is activated.

11.1.5.6.2 If one or more triggers are provided, the trace is started when any one of the start triggers is detected after the activation of the DCP.

11.1.5.6.3 Subsequent occurrences of any of the start triggers after the trace has started are ignored, and do not affect the execution of the trace.

11.1.5.6.4 The same start trigger may not appear more than once in the same TraceRequest (see Section 9.1.2.2.11).

11.1.5.7 *stopOn*

11.1.5.7.1 If no triggers are provided, the trace terminates when the total number of collection results is equal to 'collectionCount', or when the DCP is deactivated, whichever occurs first.

11.1.5.7.2 If one or more triggers are provided, the trace is stopped when any one of the stop triggers is detected, or when the total number of collection results is equal to 'collectionCount', whichever occurs first. If the trace request is buffered (groupSize > 1), any collected results stored at this time are sent in a single DataCollectionReport.

11.1.5.7.3 If a trigger provided by this association is identical to one specified in the 'startOn' association, trace data collection shall start on the first occurrence of that trigger after activation of the plan, and stop on the next occurrence of the same trigger (and so on, if cyclical).

11.1.5.7.4 The same stop trigger may not appear more than once in the same TraceRequest (see Section 9.1.2.2.11).

11.1.6 *Trigger* — An abstract class representing an occurrence that can trigger the starting or stopping of a trace. This class has no attributes or associations.

11.1.7 *EventTrigger* — Identifies an event that can be generated by a specific source, and whose occurrence can trigger the starting or stopping of a trace. See Section 9.1.2.2.11 for possible errors.

11.1.7.1 *EventTrigger Attribute Definition Table*

Table 48 EventTrigger Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The source of the event.	Text, formatted according to the convention used to identify an event source. See limitations, Section 3.1.1.
eventId	The identifier for the event.	Text, formatted according to the convention used to identify a specific event. See limitations, Section 3.1.1.

11.1.8 *ExceptionTrigger* — Identifies an exception that can be generated by a specific source, and whose occurrence can trigger the starting or stopping of a trace. For stateful exceptions, the trigger can be activated by a transition to any of the defined states for that exception. See Section 9.1.2.2.11 for possible errors.

11.1.8.1 *ExceptionTrigger Attribute Definition Table*

Table 49 ExceptionTrigger Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The source of the exception.	Text, formatted according to the convention used to identify an exception source. See limitations, Section 3.1.1.
exceptionId	The identifier for the exception.	Text, formatted according to the convention used to identify a specific exception. See limitations, Section 3.1.1.
exceptionState	For stateful exceptions, the exception state which activates the trigger.	Text, equal to a valid state for the exception. For exceptions corresponding to E30 alarms, valid values are: "urn:semi-org:E30:alarmSet" "urn:semi-org:E30:alarmClear" If no state is provided, the trigger shall activate when the equipment detects an occurrence of the exception.

11.1.9 *ParameterRequest* — This class provides the source and name of a specific parameter that the equipment is to report to the consumer as part of the DCP's event or trace reports. Multiple trace reports or event reports across multiple DCPs may refer to the same parameter.

11.1.9.1 *ParameterRequest Attribute Definition Table*

Table 50 ParameterRequest Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	Unique identifier for the source of the parameter.	Text, must be a valid parameter source, formatted according to the convention used for identifying a source. See limitations, Section 3.1.1 .
parameterName	Name of the parameter.	Text, must be a valid parameter name for the specified source, formatted according to the convention used for identifying a parameter. See limitations, Section 3.1.1 .

12 Data Collection Plan State Models

12.1 *DataCollectionPlan State Model*

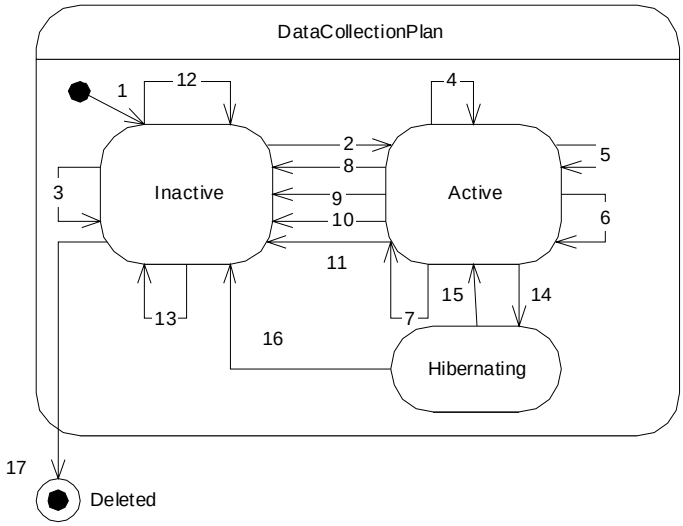

Figure 20
DataCollectionPlan State Model

12.1.1 The purpose of the DataCollectionPlan state model is to define the consumer's view of the behavior of a DCP, which includes the consumer's interactions with the equipment necessary to define, activate, de-activate, and delete DCPs. Each DCP on the equipment shall execute independently, as described in these state models. The equipment supplier is not required to provide a direct, literal implementation of these state models. They are documented here to provide a precise definition of the required behavior for DCPs on the equipment. The transitions described in these state machines are not to be made available for event data collection.

12.1.1.1 Implementation mappings of this specification may identify additional conditions other than those defined in this document under which DCP activity should be terminated (that is, cause a transition from the "Active" to the "Inactive" state). All such conditions, and the details of required equipment behavior under those conditions, shall be fully specified in any implementation mapping for which this is a consideration.

12.1.2 *DataCollectionPlan* — The superstate for all DCP states. The equipment has validated the DCP and stored it in non-volatile memory. The DCP may now be activated and de-activated.

12.1.3 *Inactive* — The equipment is not sending or buffering any event, exception, or trace reports requested in the DCP to any consumer. A transition to *Active* is required for the data defined in this DCP to be reported to a consumer.

12.1.4 *Active* — The equipment is sending and/or buffering all event, exception, and trace reports requested in the DCP to all consumers that have successfully activated the DCP. This state contains substates that are described in Section.12.2 .

12.1.5 *Hibernating* — The equipment has paused the sending and buffering of all event, exception, and trace reports requested in the DCP, because the equipment is entering a state in which it will not be possible to communicate data to any consumers. The DCP has its 'isPersistent' attribute equal to 'true'.

12.1.6 *Deleted* — The DCP is no longer stored in non-volatile memory, and can no longer be legally referenced by any consumer.

12.1.7 *DataCollectionPlan Transition Table*

Table 51 DataCollectionPlan Transition Definition

Num	*Previous State*	*Trigger*	*New State*	*Actions*
1	(no state)	A consumer invokes the DefinePlan operation of the DataCollection-Manager interface. The equipment has validated its contents.	Inactive	The equipment shall store the DCP in non-volatile memory. The DCP can now be activated.
2	Inactive	A consumer invokes the ActivatePlan operation of the DataCollection-Manager interface using the identifier of this DCP. The consumer has sufficient privilege for this operation.	Active	The equipment shall store the id of the requesting consumer and the time of the request. If the DCP is persistent, the equipment shall store this information in non-volatile memory. The equipment shall begin sending and/or buffering all exception, event, and trace reports according to the DCP definition, to the requesting consumer.
3	Inactive	A consumer invokes the ActivatePlan operation of the DataCollection-Manager interface using the identifier of this DCP. The equipment has determined that the DCP is invalid.	Inactive	The equipment shall not activate the DCP. The equipment shall identify all errors in the DCP, and return this information using the InvalidPlan error, defined in Section 9.1.2.2.6.
4	Active	A consumer invokes the ActivatePlan operation of the DataCollection-Manager interface using the identifier of this DCP. The consumer has sufficient privilege. The DCP was previously activated by at least one other consumer.	Active	If the requesting consumer had already previously requested activation of this DCP, the equipment shall not activate the DCP again for that consumer (see Section 9.1.2.5.3). Otherwise, the equipment shall store the id of the requesting consumer and the time of the request. If the DCP is persistent, the equipment shall store this information in non-volatile memory. The equipment shall begin sending data from this DCP to the requesting consumer beginning with the next available DataCollectionReport (see Section 9.1.2.5).
5	Active	A consumer invokes the DeactivatePlan operation of the DataCollectionManager interface using the identifier of this DCP. The consumer has sufficient privilege. More than one consumer is currently receiving data from this DCP.	Active	The equipment shall stop sending data from this DCP to the requesting consumer. The equipment shall delete the activation information stored (either in volatile, or non-volatile storage) when the requesting consumer originally activated the DCP. The equipment shall continue to buffer/send data from this DCP to the remaining consumers.
6	Active	A consumer with insufficient privilege invokes the DeactivatePlan operation of the DataCollectionManager interface using the identifier of this DCP.	Active	The equipment shall respond to the consumer with an error stating that the DCP will not be deactivated because the consumer does not have sufficient privilege. See Section 9.1.2.7.
7	Active	A consumer invokes the DeletePlan operation of the DataCollection-Manager interface using the identifier of this DCP. At least one consumer is receiving data from this DCP.	Active	The equipment shall respond to the consumer with an error stating that the DCP will not be deleted because the DCP is active (see Section 9.1.2.8).

Num	*Previous State*	*Trigger*	*New State*	*Actions*
8	Active	A consumer invokes the DeactivatePlan operation of the DataCollection-Manager interface using the identifier of this DCP. The consumer has sufficient privilege, and there are no other consumers.receiving data from this DCP.	Inactive	The equipment shall delete the activation information stored (either in volatile, or non-volatile storage) when the requesting consumer originally activated the DCP. The equipment stops sending and buffering the exception, event, and trace reports requested in the DCP to the requesting consumer, discarding any data buffered at the time of the deactivation.
9	Active	A consumer invokes the DeactivatePlan operation of the DataCollectionManager interface using the identifier of this DCP. The consumer has sufficient privilege and is requesting that the DCP be terminated (see Section 9.1.2.7.5).	Inactive	The equipment shall stop sending and buffering all data for this DCP to all consumers that are currently receiving its data. The equipment shall notify each consumer of the reason for the de-activation using the DCPDeactivated class defined in Section 9.1.2.7.6. The equipment shall delete the activation information stored (either in volatile, or non-volatile storage) when each consumer originally activated the DCP.
10	Active	The equipment is going to enter a state in which it will not be possible to communicate data to any consumers (e.g., equipment shutdown or emergency power off). The DCP is not persistent.	Inactive	The equipment shall deactivate the DCP and notify each consumer receiving data from this DCP (if possible), as described in Section 15.1.6.2. The equipment stops sending and buffering the exception, event, and trace reports requested in the DCP to the consumer. The equipment shall discard all data that has been buffered up to this point. If possible, the equipment shall delete the activation information stored when each consumer originally activated the DCP.
11	Active	The equipment determines that performance conditions require that data collection activity be terminated (see Section 13.1.4).	Inactive	The equipment terminates all active DCPs, and notifies each consumer using the DCPDeactivation notification defined in Section 15.1.6.2. The equipment shall delete the activation information stored (either in volatile, or non-volatile storage) when each consumer originally activated the DCP
12	Inactive	A consumer with insufficient privilege invokes the ActivatePlan operation of the DataCollectionManager interface using the identifier of this DCP.	Inactive	The equipment responds to the consumer with an error stating that the DCP will not be activated because the consumer doesn't have sufficient privilege (see Section 9.1.2.7).
13	Inactive	A consumer with insufficient privilege invokes the DeletePlan operation of the DataCollectionManager interface using the identifier of this DCP.	Inactive	The equipment responds to the consumer with an error stating that the DCP will not be deleted because the consumer doesn't have sufficient privilege (see Section 9.1.2.8).
14	Active	The equipment is going to enter a state in which it will not be possible to communicate data to any consumers (for example, as part of an orderly shutdown or emergency power off). The DCP is persistent.	Hibernating	The equipment shall notify the consumers (if possible) that the DCP is going into a Hibernating state, as described in Section 15.1.6.3. The equipment stops reporting and buffering the exception, event, and trace reports defined in the DCP to the consumers. The equipment shall discard all data that has been buffered up to this point.

Num	*Previous State*	*Trigger*	*New State*	*Actions*
15	Hibernating	The equipment has entered a state in which it is possible to begin communicating data to consumers (for example, as part of startup or power on).	Active	The equipment shall resume sending and/or buffering all exception, event, and trace reports, according to the DCP definition, and sending these reports to the consumers that had previously activated the DCP. If the DCP contains TraceRequests, all such traces shall be re-started in the Tracing-Disabled state (see Section 12.3).
16	Hibernating	The equipment, prior to re-activating a DCP in the Hibernating state, has determined that the DCP is invalid.	Inactive	The equipment shall send the DCPDeactivation notification (see Section 15.1.6.2) to all consumers that had previously activated the DCP, and terminate the DCP.
17	Inactive	A consumer invokes the DeletePlan operation of the DataCollection-Manager interface using the identifier of this DCP.	Deleted	The equipment deletes the DCP from non-volatile storage.

12.2 *DataCollectionPlan-Active State Model*

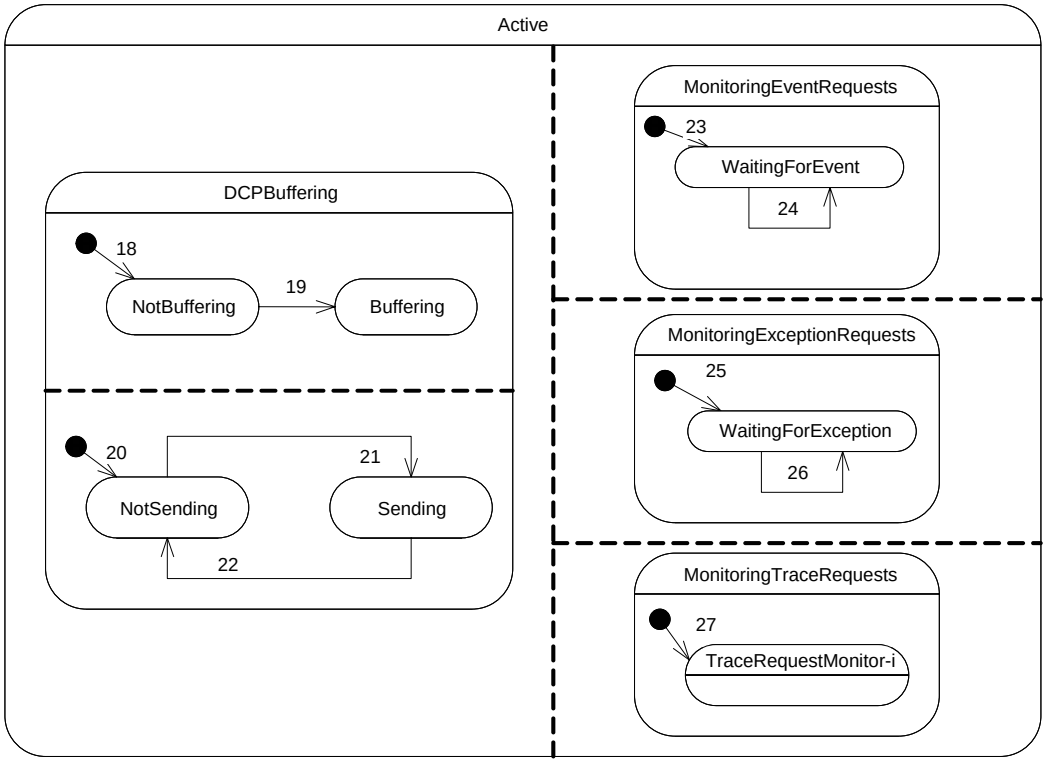

Figure 21
DataCollectionPlan-Active State Model Diagram

12.2.1 The purpose of the DataCollectionPlan-Active State Model is to define a consumer's view of DCP behavior in the *Active* state of the DataCollectionPlan State Model. Each DCP's Active state on the equipment shall execute independently, as described in this state model. The equipment supplier is not required to provide a direct, literal implementation of this state model. It is documented here to provide a precise definition of the required behavior

for DCPs in the Active state on the equipment. The transitions described in this state model are not to be made available for event data collection.

12.2.2 While in the Active state, if the equipment experiences internal communication or operational failures that prevent the collection of event, exception, or trace data requested by the DCP, the equipment shall not de-activate the DCP. If the values of any requested parameters can't be obtained or refreshed as a result of such communication failures, the equipment shall use the "NoValue" class (see Section 14.2.1.4) as the reported ParameterValue for all such affected parameters. The equipment is not required to take any other actions to notify consumers of such failures.

12.2.2.1 Consumers should be aware that such failures may temporarily prevent the equipment from reporting requested events or exceptions altogether, and/or result in reporting "NoValue" (see Section 14.2.1.4) for requested parameters affected by the failure. In the event that the equipment recovers from such failures and any affected DCPs are still active, the equipment shall resume reporting all event, exception, and trace data whose reporting had been affected by the failure.

12.2.3 *DCPBuffering* — This is the top level state for describing the event, exception, and trace buffering behavior of the DCP

12.2.4 *NotBuffering* — The DCP is immediately sending DataCollectionReports for each requested event and exception as they are detected. TraceReports may be individually buffered, according to the 'groupSize' setting of the DCP's TraceRequests (see Section 11.1.5), but are also sent in the DataCollectionReport format (see Section 14.1). This state is only active if the DataCollectionPlan attribute 'intervalInMinutes' attribute is set to 0.

12.2.5 *Buffering* — The DCP is storing all events, exceptions, and trace data in an internal buffer in the order in which they occur. Buffered data is sent in a single DataCollectionReport (see Section 14.1) when 'intervalInMinutes' expires, or when the equipment's DCP buffering capacity has been reached, whichever occurs first. The equipment continues buffering new data for the next interval, sending a single DataCollectionReport when 'intervalInMinutes' expires. Each DataCollectionReport sent at the end of each interval contains only the data collected during that interval. The equipment continues this buffering/sending cycle as long as the DCP is active. This state is only active if the DataCollectionPlan attribute 'intervalInMinutes' attribute is > 0.

12.2.6 *NotSending* — The DCP is waiting for a DataCollectionReport to become ready for sending to the consumer who activated the plan.

12.2.7 *Sending* — The DCP is sending a DataCollectionReport to the consumer that activated this DCP (see Section 14.1).

12.2.8 *MonitoringEventRequests* — The DCP is actively tracking and reporting any event requests provided with the DCP. This is the top level state for the *WaitingForEvent* state.

12.2.9 *WaitingForEvent* — The DCP is waiting for the occurrence of any of the requested events defined in the DCP.

12.2.10 *MonitoringExceptionRequests* — The DCP is actively tracking and reporting any exception requests provided with the DCP. This is the top level state for the *WaitingForException* state.

12.2.11 *WaitingForException* — The DCP is waiting for the occurrence of any of the requested exceptions specified in the DCP.

12.2.12 *MonitoringTraceRequests* — The DCP is actively tracking and reporting the results from any trace requests provided with the DCP. This is the top level state for the *TraceRequestMonitor-i* states.

12.2.13 *TraceRequestMonitor-i* — For each trace request provided with the DCP, there is an independently-executing *TraceRequestMonitor-i* (i = 0 up to the number of trace requests included in the DCP) that defines the DCP reporting behavior for trace requests. This state contains substates that are described in Section 12.3.

12.2.14 *DataCollectionPlan-Active Transition Table*

Table 52 DataCollectionPlan-Active Transition Definition

Num	*Previous State*	*Trigger*	*New State*	*Actions*
18	(no state)	DCP is activated	NotBuffering	The DCP's 'intervalInMinutes attribute is inspected to determine whether or not the DCP should be buffering event, exception, and trace data results. If set to 0, the DCP shall immediately send event, exception, and trace results as they occur. See Section 12.2.4 for further explanation.
19	NotBuffering	DCP 'intervalInMinutes' attribute is > 0.	Buffering	The DCP begins buffering all event, exception, and trace results in the order that they occur. See Section 12.2.5 for further explanation.
20	(no state)	DCP is activated.	NotSending	The DCP begins waiting for a DataCollectionReport to be made available for sending to the consumer.
21	NotSending	A new DataCollectionReport is ready to be sent to the consumer.	Sending	The DCP sends the DataCollectionReport to the consumer.
22	Sending	A DataCollectionReport has been sent to the consumer.	NotSending	The DCP waits for a DataCollectionReport to be made available for sending to the consumer.
23	(no state)	DCP is activated.	WaitingForEvent	The DCP begins waiting for the occurrence of any requested events.
24	WaitingForEvent	A requested event has been detected.	WaitingForEvent	The DCP creates an EventReport containing the parameters requested for that event (see Section 14.1.3). If the DCP is not buffered, a new DataCollectionReport (see Section 14.1) is created containing the Event Report and made available for sending. If the DCP is buffered, the EventReport is added to the DCP buffer in the order in which it was detected. The equipment continues waiting for the occurrence of any requested events.
25	(no state)	DCP is activated.	WaitingForException	The DCP begins waiting for the occurrence of any requested exceptions. Any exceptions in the DCP whose state the equipment tracks and that have been detected at the time the DCP is activated are immediately buffered or sent to the consumer in a single DataCollectionReport, formatted according to Section 14.1. The current state of each exception shall be provided in their individual ExceptionReports (see Section 14.1.4).

Num	*Previous State*	*Trigger*	*New State*	*Actions*
26	WaitingForException	An exception matching the criteria specified in an exception request has occurred.	WaitingForException	The DCP creates an ExceptionReport for the exception (see Section 14.1.4). If the DCP is not buffered, a new DataCollectionReport (see Section 14.1) is created containing the ExceptionReport and made available for sending. If the DCP is buffered, the ExceptionReport is added to the DCP buffer in the order in which the exception was detected.
27	(no state)	DCP is activated.	TraceRequestMonitor-i	The DCP begins monitoring any trace requests. See Section 12.3 for further explanation.

12.3 *Trace Request State Model*

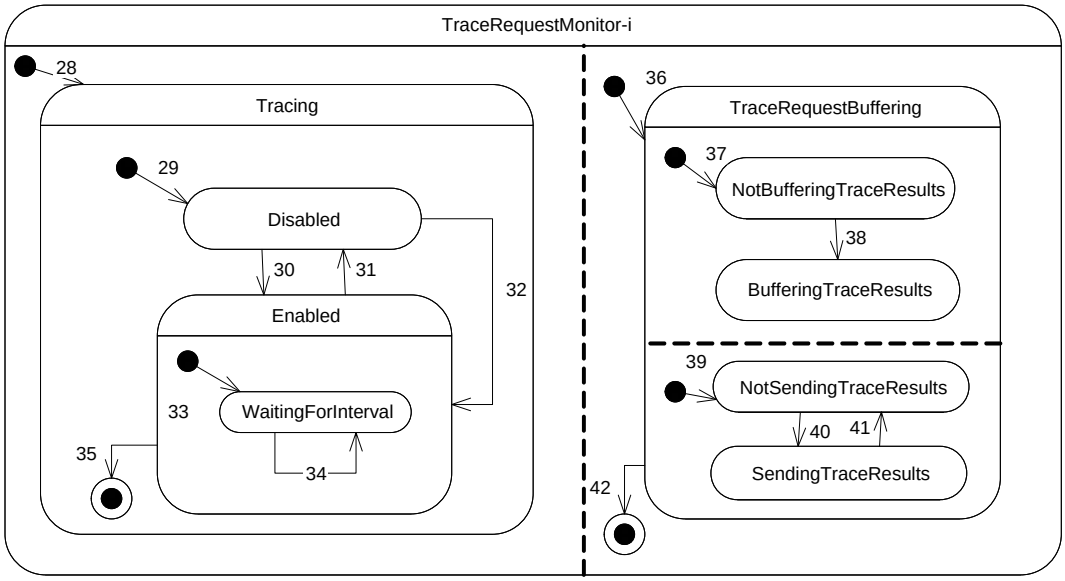

Figure 22
TraceRequest State Model

12.3.1 The purpose of the trace request state model is to define a consumer's view of DCP trace behavior while in the *Active* state of the DataCollectionPlan state model. Each trace request within each active DCP shall execute independently, as described in this state model. The equipment supplier is not required to provide a direct, literal implementation of this state model. It is documented here to provide a precise definition of the required behavior for DCP trace activity on the equipment. The transitions described in this state model are not to be made available for event data collection.

12.3.2 *TraceRequestMonitor-i* — This is the top level state for describing the behavior of a single trace request within a single DCP in the Active state. Each trace request within each active DCP executes this state model independently.

12.3.3 *Tracing* — This is the top level state for trace activity for a single trace request.

12.3.4 *Disabled* — The DCP is not performing trace data collection. The DCP is waiting for the occurrence of a start trigger defined for this trace request.

12.3.5 *Enabled* — The DCP is performing trace data collection (see substate descriptions) as requested in a specific trace request provided with the DCP. Either the occurrence of a start trigger defined for the trace request was detected, or there was no defined start trigger and the DCP began tracing immediately after activation. If the DCP entered this state due to the detection of a start trigger, it shall record the specific trigger that was detected and the time at which it was detected for inclusion in a trace report (see Section 14.1.5).

12.3.6 *Waiting forInterval* — The DCP is waiting for a time period equal to the TraceRequest attribute 'intervalInSeconds' (see Section 11.1.5) to expire before collecting the data associated with the trace request.

12.3.7 *TraceRequestBuffering* — This is the top level state for trace request buffering activity. Buffered trace requests store the results of each trace interval as a CollectedData entry within a single TraceReport (see Section 14.1.5). Each CollectedData entry appears within the TraceReport in the order the collection results were obtained.

12.3.7.1 When a stop trigger is detected, or the total number of results collected since the trace was enabled is equal to 'collectionCount', or the number of results collected since the last send is equal to 'groupSize', or the DCP's trace buffering capacity has been reached, the DCP creates a single DataCollectionReport and makes it available for sending. The DataCollectionReport contains a single TraceReport that contains two or more CollectedData entries, in the order in which the results were obtained (see Section 14.1.5).

12.3.8 *NotBufferingTraceResults* — The trace request is not buffering collected results into a single TraceReport, because 'groupSize' is ≤ 1. Each collected result is added to a new TraceReport and either made available for sending with a DataCollectionReport, or added to the DCP buffer in the order the TraceReport was completed.

12.3.9 *BufferingTraceResults* — The trace request is buffering collected results into a single TraceReport because 'groupSize' is > 1. Each new result is appended to the TraceReport as CollectedData (see Section 14.1.5.2).

12.3.10 *NotSendingTraceResults* — The DCP is waiting for this trace request to produce a DataCollectionReport containing a buffered TraceReport for sending.

12.3.11 *SendingTraceResults* — A DataCollectionReport containing a buffered TraceReport has been made available for sending. The DCP sends the DataCollectionReport to the DCP consumer.

12.3.12 *TraceRequest Transition Table*

Table 53 TraceRequest Transition Definition

Num	*Previous State*	*Trigger*	*New State*	*Actions*
28	(no state)	DCP is activated.	Tracing	The DCP begins monitoring trace activity for this trace request
29	(no state)	DCP is activated.	Disabled	If there are any start triggers defined for this trace request, the DCP stays in the Disabled state until the occurrence of one of the start triggers before enabling the trace. See Section 12.3.4 for further explanation.
30	Disabled	DCP is activated, or is returning from the Hibernating to the Active state. There are no start triggers defined for the trace request.	Enabled	The DCP initiates trace activity for this trace request.
31	Enabled	The stop trigger for this request is detected. The trace request is cyclical.	Disabled	The equipment shall record the specific trigger that was detected and the time at which it was detected for inclusion in the corresponding trace report (see Section 14.1.5). The DCP sends any buffered trace results to the consumer in a single DataCollectionReport message, and begins waiting for the occurrence of a requested start trigger.

Num	*Previous State*	*Trigger*	*New State*	*Actions*
32	Disabled	A start trigger was detected.	Enabled	The DCP initiates trace activity for this trace request. The DCP shall record the specific trigger that was detected and the time at which it was detected for inclusion in the first TraceReport generated (see Section 14.1.5).
33	(no state)	The trace request is enabled	WaitingForInterval	The DCP collects and records the requested trace data and buffers or sends a new TraceReport. See Section 12.3.13 for further explanation. The DCP continues waiting for the next 'intervalInSeconds' period to expire.
34	WaitingForInterval	The period specified by 'intervalInSeconds' for this trace request expires.	WaitingForInterval	The DCP collects and records the requested trace data in a corresponding TraceReport. See Section 12.3.13 for further explanation. The DCP continues waiting for the next 'intervalInSeconds' period to expire.
35	Enabled	The trace request is not cyclical ('isCyclical' is false), and the stop trigger for this request was detected, or the total number of requested collected results was obtained.	(no state)	The trace terminates, and cannot be re-enabled without deactivating and re-activating the DCP.
36	(no state)	The DCP is activated.	TraceRequest-Buffering	The DCP begins performing any trace buffering required for this trace request.
37	(no state)	The DCP is activated.	NotBufferingTrace-Results	The DCP determines whether this trace request should be buffered. If this trace request is not buffered, the trace request remains in this state until the trace terminates. See Section 12.3.8 for further explanation.
38	NotBufferingTrace-Results	The trace request's 'groupSize' is > 1.	BufferingTrace-Results	The DCP begins buffering collected trace results. See Section 12.3.9 for further explanation.
39	(no state)	The DCP is activated.	NotSendingTrace-Results	The DCP waits for a buffered trace report to become available for sending.
40	NotSendingTrace-Results	A buffered trace report is available for sending.	SendingTrace-Results	The DCP sends the corresponding DataCollectionReport to the consumer.
41	SendingTrace-Results	A buffered trace report has been sent to the consumer.	NotSending-TraceResults	The DCP waits for a buffered trace report to become available for sending.
42	TraceRequest-Buffering	The trace request is not cyclical ('isCyclical' is false), and the stop trigger for this request was detected, or the total number of requested collected results was obtained.	(no state)	The DCP performs no more activities related to this trace request.

12.3.13 *Reporting Trace Results* — Trace results are reported as CollectedData within a TraceReport (see Section 14.1.5). In addition to the collected data values, the DCP shall record the time at which the data collection was initiated (the time at which the TraceRequest's 'intervalInSeconds' period expired).

12.3.13.1 If the DCP is buffered ('intervalInMinutes' attribute is > 0), and the TraceRequest is not buffered (groupSize is ≤ 1), the DCP shall create a new TraceReport containing a single CollectedData entry with the

collected data values for that trace interval. This TraceReport shall be added to the DCP buffer according to the time at which the report was completed. This process is repeated at each trace interval until the trace terminates.

12.3.13.2 If the DCP is buffered ('intervalInMinutes' attribute is > 0), and the TraceRequest is buffered (groupSize is > 1), the DCP shall add the results as a CollectedData entry to the trace buffer in the order it was obtained. This process continues until a number of results equal to 'groupSize' has been collected. The DCP shall then create a new DataCollectionReport containing a single TraceReport with a number of CollectedData entries equal to 'groupSize'. This DataCollectionReport is then added to the DCP buffer according to the time at which the report was completed. This process is repeated at each trace interval until the trace terminates.

12.3.13.3 If the DCP is not buffered ('intervalInMinutes' attribute is 0), and the TraceRequest is not buffered (groupSize is ≤ 1), the DCP shall create a new DataCollectionReport containing a single TraceReport with a single CollectedData entry for that interval and make the DataCollectionReport available for sending. This process is repeated at each trace interval until the trace terminates.

12.3.13.4 If the DCP is not buffered, but the TraceRequest is buffered (groupSize > 1), the DCP shall add the results as a CollectedData entry to the trace buffer in the order it was obtained. This process continues until a number of results equal to 'groupSize' has been collected. The DCP shall then create a new DataCollectionReport containing a single TraceReport with a number of CollectedData entries equal to 'groupSize'. This DataCollectionReport is then made available for sending. This process is repeated at each trace interval until the trace terminates.

13 Operational Performance Monitoring

13.1 *Operational Performance Conditions State Model*

13.1.1 The purpose of the Operational Performance Conditions State Model is to define a consumer's view of equipment behavior when its ability to perform its intended function (including the production of data requested by active DCPs) has become impaired due to resource overloading.

13.1.2 The equipment shall continuously monitor (at equipment-defined intervals) its operational performance status according to its established criteria. Each time the performance is evaluated, the equipment shall store in non-volatile memory the job and DCP identifiers that were active at the time the evaluation was initiated (see Section 15.1.6.1.1). The equipment shall use this information as the basis for replies to the GetCurrentPerformanceStatus operation of the DataCollectionManager interface, and for the PerformanceWarning and PerformanceRestored notifications sent via the DCPConsumer interface (see Section 15.1.5).

13.1.3 The equipment shall include the following minimum set of conditions in determining the performance conditions to monitor:

- Impairment of the equipment's ability to properly control its components for their intended purpose.
- Impairment of the equipment's ability to process/measure/test material at its intended throughput rates.
- Impairment of the equipment's ability to deliver data from active DCPs to consumers.

13.1.4 If at any time the equipment determines that continuing to operate under degraded performance conditions will endanger people, the equipment (including its ability to sustain the transmission of data, or the processing of data collection management operations), or the material being processed, measured, or tested, the equipment shall terminate all active DCPs, and notify each client accordingly, using the DCPDeactivation message (see Section 15.1.6.2).

13.1.5 Note that this specification does not define a mechanism for establishing operational performance thresholds, or any technique for detecting when an equipment system's operational performance is degrading. It is assumed that equipment performance can be quantified, that it is possible to detect when such performance degrades below some supplier-defined threshold, and that it is possible to identify which components of the equipment are affected by the issue.

13.1.6 *Normal* — The equipment is operating within supplier-specified performance parameters.

13.1.7 *BelowThreshold* — One or more components of the equipment system is operating below supplier-specified performance parameters.

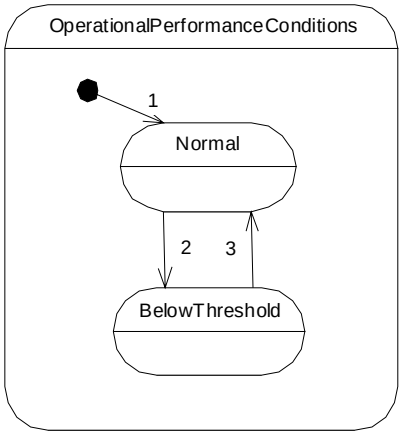

Figure 23
Operational Performance State Diagram

13.1.8 *OperationalPerformanceConditions Transition Table*

Table 54 Operational Performance Conditions Transition Definition

Num	*Previous State*	*Trigger*	*New State*	*Actions*
1	(no state)	The equipment initializes.	Normal	
2	Normal	The equipment detects a reduction in performance below supplier-specified parameters.	BelowThreshold	The equipment notifies all consumers that have the ManageOnlyAuthoredDCPs privilege level or higher,that the equipment is experiencing operational performance problems, providing the information described in Section 15.1.5.
3	BelowThreshold		Normal	The equipment notifies all consumers that have the ManageOnlyAuthoredDCPs privilege level or higher of the current performance status, providing the information described in Section 15.1.5.

14 Data Collection Report Formats

14.1 *Data Collection Reports*

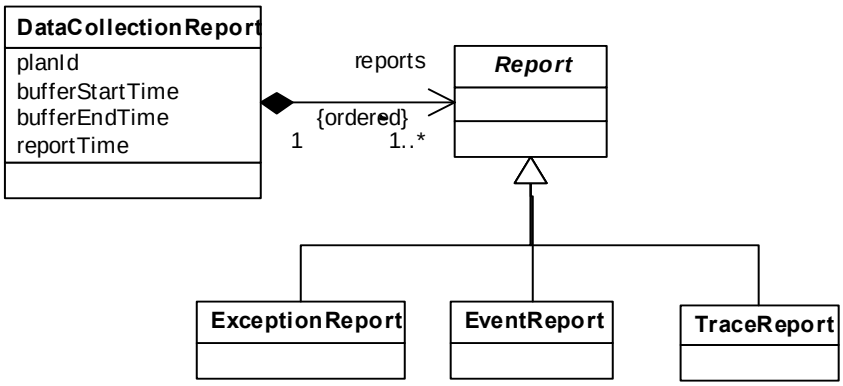

Figure 24
Data Collection Reports

14.1.1 Figure 24 shows how to represent a DataCollectionReport containing the data associated with a specific DCP. These formats are used to send data to consumers using the interface defined in Section 15. The equipment is not required to retain these reports for re-transmission once they have been sent to a consumer.

14.1.1.1 *DataCollectionReport* — this class represents one or more collected data items. A data collection report contains one or more Reports, which can be an ExceptionReport, EventReport, or TraceReport. If the DataCollectionReport contains more than one of those items, they are ordered according to the time of their occurrence ('reportTime', in the case of TraceReports). A DataCollectionReport for a given DCP shall not contain more than one Report if the 'intervalInMinutes' field of the corresponding DCP was set to 0.

14.1.1.2 *DataCollectionReport Attribute Definition Table*

Table 55 DataCollectionReport Attribute Definition

Attribute Name	*Definition*	*Form*
planId	The unique id of this trace report, as specified in the DCP.	Text
bufferStartTime	The time at which the buffer period being reported was started.	Text, formatted according to Section 14.5 .
bufferEndTime	The time at which the buffer period being reported ended.	Text, formatted according to Section 14.5 .
reportTime	The time at which the DataCollectionReport was created.	Text, formatted according to Section 14.5 .

14.1.1.3 *DataCollectionReport Association Definition Table*

Table 56 DataCollectionReport Association Definition

Association Role Name	*Definition*	*Comments*
reports	One or more reports providing data requested by an active DCP.	Ordered list of structured data, of any type derived from Report.

14.1.2 *Report* — This abstract base class represents any report item that can be included in a DataCollectionReport. This class has no attributes or associations.

14.1.3 *Event Reports*

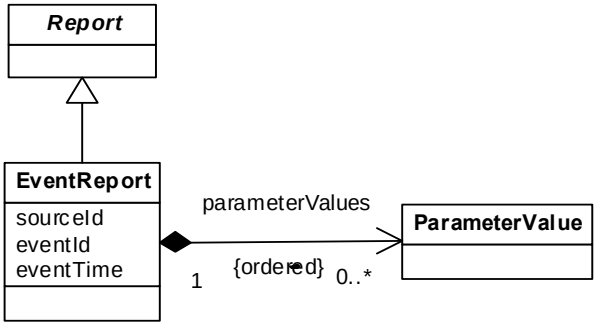

Figure 25
Event Reports

14.1.3.1 Figure 25 shows how to represent the occurrence of an event included in a DCP.

14.1.3.1.1 *EventReport* — Represents the occurrence of an event. Each event has a source or origin of the event, a unique event id, and the time at which the event was detected.

14.1.3.1.2 *EventReport Attribute Definition Table*

Table 57 EventReport Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The identifier of the source of the event.	Text, formatted according to the convention used for identifying an event source (see limitations, Section 3.1.1).
eventId	The identifier of the event.	Text, formatted according to the convention used for identifying an event (see limitations, Section 3.1.1).
eventTime	The time at which the event was detected.	Text, formatted according to Section 14.5.

14.1.3.2 EventReport Association Definition Table

Table 58 EventReport Association Definition

Association Role Name	*Definition*	*Comments*
parameterValues	The parameters requested for this event.	List of zero or more elements of type ParameterValue, described in Section 14.2. ParameterValues appear in the order specified in the original EventRequest in the corresponding DCP.

14.1.4 *Exception Reports*

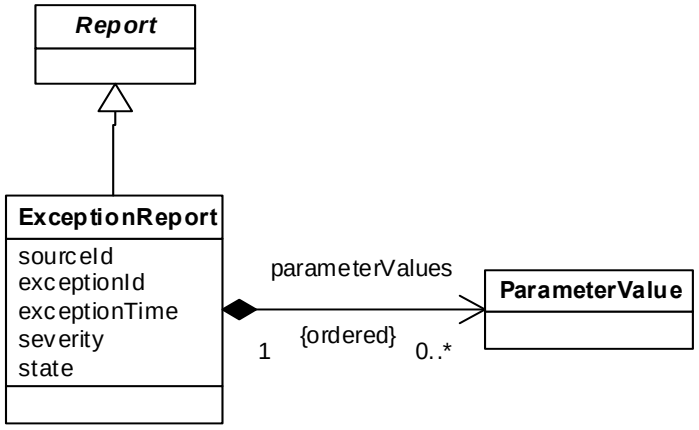

Figure 26
Exception Reports

14.1.4.1 Figure 26 shows how to report the occurrence of an exception included in a DCP.

14.1.4.1.1 *ExceptionReport* — Represents the occurrence of an exception. Each exception has a source or origin of the exception, a unique exception id, the time at which the exception was detected, and the severity of the exception.

14.1.4.1.2 *ExceptionReport Attribute Definition Table*

Table 59 ExceptionReport Attribute Definition

Attribute Name	*Definition*	*Form*
sourceId	The identifier of the source of the exception.	Text, formatted according to the convention used for identifying an exception source (see Limitations, Section 3.1.1).

Attribute Name	*Definition*	*Form*
exceptionId	The identifier of the exception.	Text, formatted according to the convention used for identifying an exception (see Limitations, Section 3.1.1).
exceptionTime	The time at which the exception was detected.	Text, formatted according to Section 14.5.
severity	The severity of the exception.	Text. Allowable values are supplier-defined.
state	The state of the exception, if applicable.	Text. For exceptions corresponding to E30 alarms, valid values are: "urn:semi-org:E30:alarmSet" "urn:semi-org:E30:alarmClear" If the exception has no associated state, this field shall be empty.

14.1.4.2 *ExceptionReport Association Definition Table*

Table 60 ExceptionReport Association Definition

Association Role Name	*Definition*	*Comments*
parameterValues	The parameters requested for this exception.	Ordered list of zero or more parameter values (see Section 14.2). Parameters are positioned in an order specified by the supplier.

14.1.5 *Trace Reports*

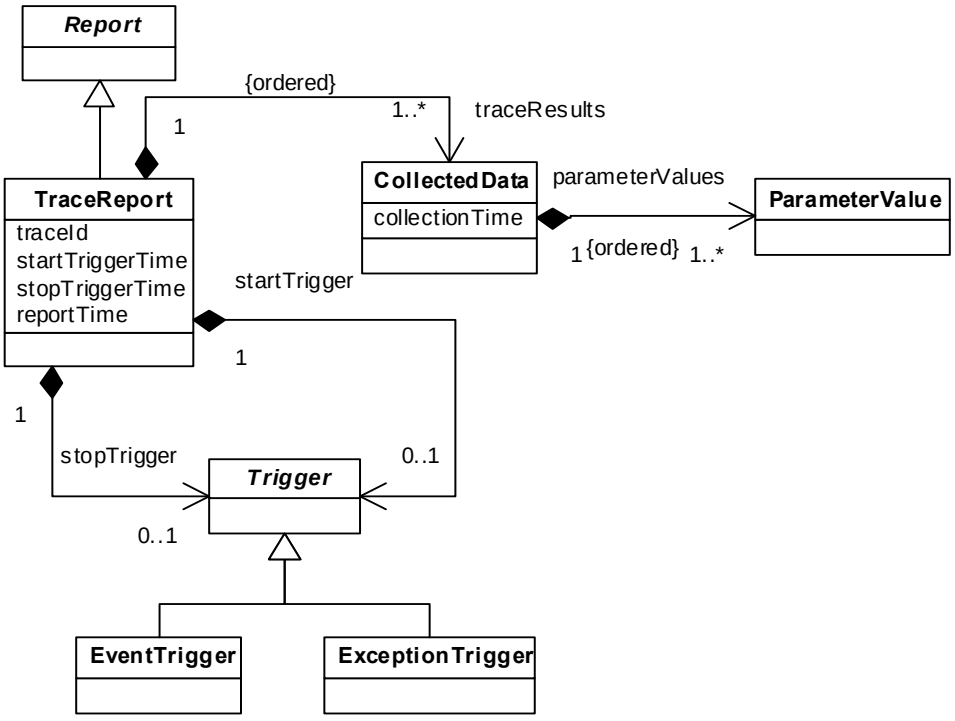

Figure 27
Trace Reports

14.1.5.1 Figure 27 shows how to represent a trace report. A single trace report can include one or more collected results, and each result includes the values of a set of parameters, appearing in the order specified in the DCP's TraceRequest.

14.1.5.1.1 More than one result is included in a trace report if and only if the DCP has specified that the equipment should buffer collected data for the trace before reporting (that is, the 'groupSize' attribute of the TraceRequest class is > 1).

14.1.5.1.2 If the trace request included start- and/or stop-triggers, the TraceReport includes the trigger that started/stopped the trace, and the time at which the trigger was detected. These are only required to be provided in the first TraceReport immediately following the detection of a start or stop trigger.

14.1.5.1.3 *TraceReport* — Represents an individual trace report that has obtained one or more collection results. A trace report has a unique identifier within the scope of the DCP, and may have been started and stopped by specific triggers, if requested in the DCP. Each of these identifiers, triggers, and associated timestamps is included in the trace report.

14.1.5.1.4 *TraceReport Attribute Definition Table*

Table 61 TraceReport Attribute Definition

Attribute Name	*Definition*	*Form*
traceId	The unique id of this trace report, as specified in the DCP.	Integer.
startTriggerTime	The time at which the trace start trigger was detected.	Text, formatted according to Section 14.5.
stopTriggerTime	The time at which the trace stop trigger was detected.	Text, formatted according to Section 14.5.
reportTime	The time at which the report creation was completed.	Text, formatted according to Section 14.5.

14.1.5.2 *TraceReport Association Definition Table*

Table 62 TraceReport Association Definition

Association Role Name	*Definition*	*Comments*
traceResults	A list of data collected for this trace report.	Ordered list of one or more elements of type CollectedData, appearing in the order of each result's start time (see Section 14.1.5.2.1). Must contain no more than 'groupSize' entries if 'groupSize' is > 1.
startTrigger	The trigger that initiated the trace	Zero or one element of any type derived from Trigger (see Section 11.1.6). Only required to be provided with the first TraceReport to be sent after the trigger was detected.
stopTrigger	The trigger that stopped the trace	Zero or one element of any type derived from Trigger (see Section 11.1.6). Only required to be provided with the first TraceReport to be sent after the trigger was detected.

14.1.5.2.1 *CollectedData* — Represents the results of the collection of the values of one or more parameters. A data collection result has a collection time corresponding to when the equipment began collecting the associated data values.

14.1.5.2.2 *CollectedData Attribute Definition Table*

Table 63 CollectedData Attribute Definition

Attribute Name	*Definition*	*Form*
collectionTime	The time at which the DCP began collecting the values of the parameters requested for this trace.	Text, formatted according to Section 14.5.

14.1.5.3 *CollectedData Association Definition Table*

Table 64 CollectedData Association Definition

Association Role Name	*Definition*	*Comments*
parameterValues	The parameters requested for this trace.	Ordered list of elements of type ParameterValue, described in Section 14.2. ParameterValues appear in the order specified in the original TraceRequest in the corresponding DCP.

14.2 *Parameter Values*

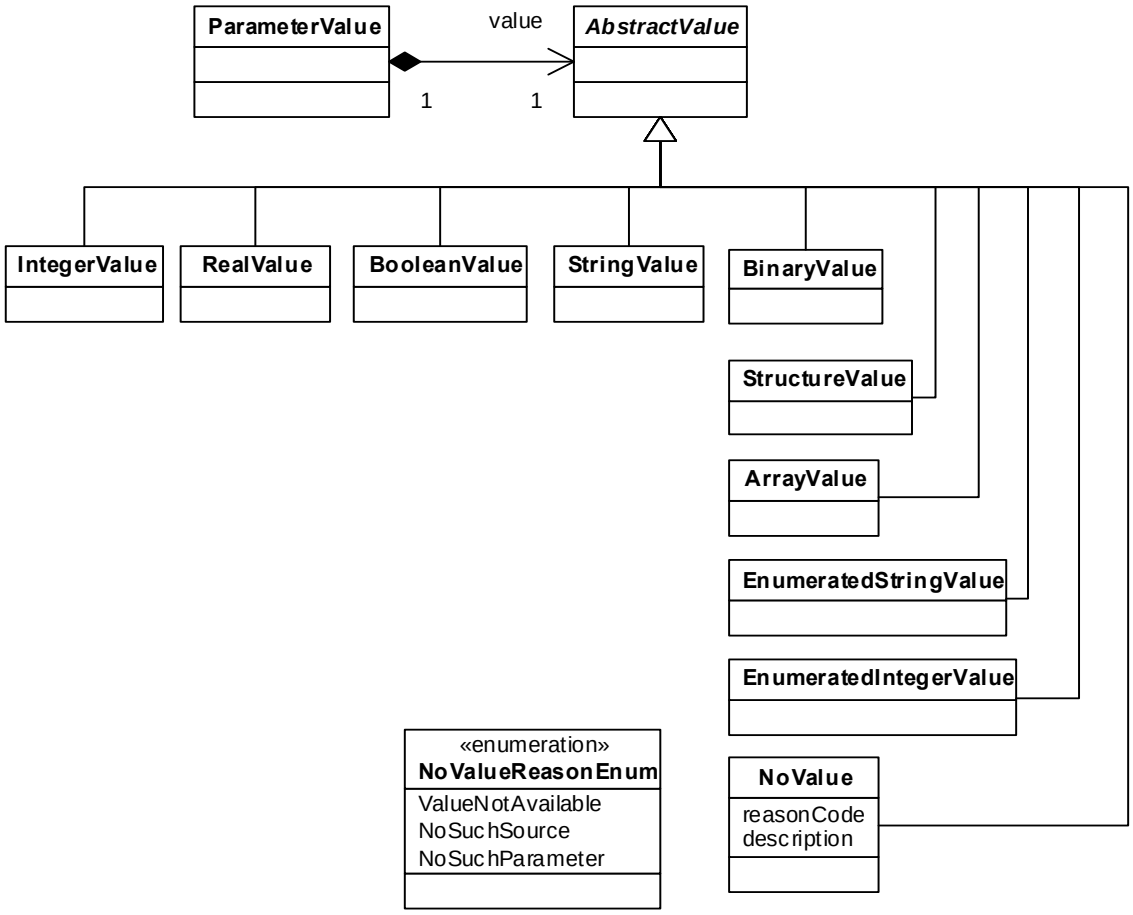

Figure 28
Parameter Values

14.2.1 Figure 28 shows how to represent a given data value corresponding to the specific parameters requested for an event or trace report in a DCP, or as a response to the GetParameterValues operation described in Section 9.1.2.9.

14.2.1.1 *ParameterValue* — Represents a data value originating from a specific source. For example, the temperature from a specific processing chamber, a measurement or test result variable, the value of an E39-based object attribute, etc.

14.2.1.2 *ParameterValue Association Definition Table*

Table 65 ParameterValue Association Definition

Association Role Name	*Definition*	*Comments*
value	The value of this parameter.	Any type derived from the abstract class AbstractValue (see Sections 14.3 and 14.4).

14.2.1.3 *AbstractValue* — An abstract class representing any derived primitive, structured, or enumerated value type. This class has no attributes or associations.

14.2.1.4 *NoValue* — This class is used to communicate problems in obtaining the values of requested Parameters. There may be situations where unanticipated conditions arise that prevent the equipment from accessing a data value due to internal communications failures or other such problems. Less commonly, a DCP that was valid at the time it was activated may refer to Parameters that are no longer defined due to some on-the-fly reconfiguration of the equipment.

14.2.1.5 *NoValue Attribute Definition Table*

Table 66 NoValue Attribute Definition

Attribute Name	*Definition*	*Form*
reasonCode	The high-level reason that the value of the Parameter can't be provided.	Enumerated, of type NoValueReasonEnum (see Section 14.2.1.6).
description	Human-readable description of the cause of the problem, if known.	Text.

14.2.1.6 *NoValueReasonEnum* — This enumeration classifies the major sources of an error in providing a Parameter's value.

14.2.1.7 *NoValueReasonEnum Attribute Definition Table*

Table 67 NoValueReasonEnum Attribute Definition

Attribute Name	*Definition*	*Form*
ValueNotAvailable	The requested Parameter exists, but equipment was unable to obtain its value.	n/a
NoSuchSource	The source of the requested Parameter does not exist.	n/a
NoSuchParameter	The source of the requested Parameter exists, but no such Parameter is provided by that source.	n/a

14.3 *Primitive Parameter Values*

14.3.1 To make the data requested in a DCP available to a consumer, there must be a mechanism for representing the values of primitive, or single-valued, types such as integers and strings in the type system being used. The specific simple or primitive types that should be represented are typically determined by the implementation technology that will be used to communicate values of the declared type (e.g., SECS-II, XML, OMG IDL, etc.). The primitive types described here should be considered a minimum set that could be refined and extended to work best with the type system in use. Any specification that translates these types into a specific type system shall define how it has mapped these primitive types into the technology-specific type system in addition to specifying any extended primitive types beyond those described here.

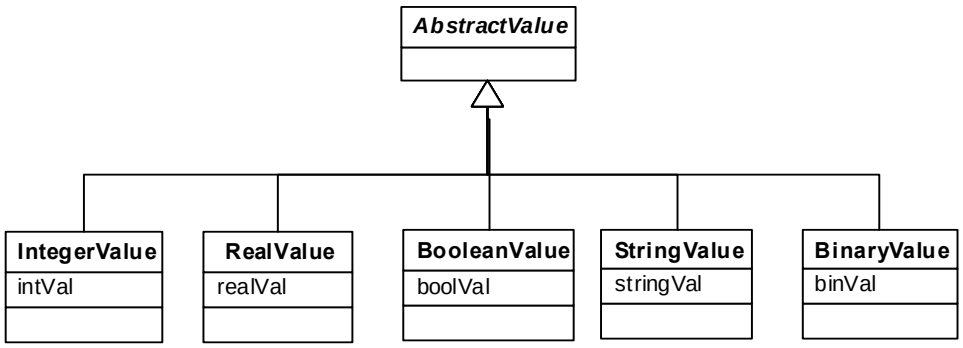

Figure 29
Primitive Value Types

14.3.1.1 Figure 29 shows a basic set of classes that can be used to represent primitive type values.

14.3.1.2 *IntegerValue* — Represents an integer value.

14.3.1.3 *IntegerValue Attribute Definition Table*

Table 68 IntegerValue Attribute Definition

Attribute Name	*Definition*	*Form*
intVal	The value of the integer item.	Integer

14.3.1.4 *RealValue* — Represents a real (floating point) number value.

14.3.1.5 *RealValue Attribute Definition Table*

Table 69 RealValue Attribute Definition

Attribute Name	*Definition*	*Form*
realVal	The value of the real number item.	Real

14.3.1.6 *BooleanValue* — Represents a boolean value.

14.3.1.7 *BooleanValue Attribute Definition Table*

Table 70 BooleanValue Attribute Definition

Attribute Name	*Definition*	*Form*
boolVal	The value of the Boolean item.	Boolean

14.3.1.8 *StringValue* — Represents a string value.

14.3.1.9 *StringValue Attribute Definition Table*

Table 71 StringValue Attribute Definition

Attribute Name	*Definition*	*Form*
stringVal	The value of the string item.	Text

14.3.1.10 *BinaryValue* — Represents a binary value.

14.3.1.11 *BinaryValue Attribute Definition Table*

Table 72 BinaryValue Attribute Definition

Attribute Name	*Definition*	*Form*
binVal	The value of the binary item.	Binary

14.4 *Structured and Enumerated Parameter Values*

14.4.1 To make the data requested in a DCP available to a consumer, there must be a mechanism for representing the values of structured types such as arrays and structures in the type system being used. The specific structured types that should be represented are typically determined by the implementation technology that will be used to communicate values of the declared type (e.g., SECS-II, XML, OMG IDL, etc.). The primitive types described here should be considered a minimum set that could be refined and extended to work best with the type system in use. Any specification that translates these types into a specific type system shall define how it has mapped these primitive types into the technology-specific type system in addition to specifying any extended primitive types beyond those described here.

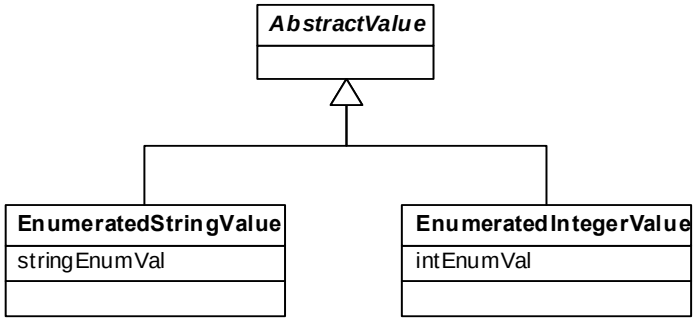

Figure 30
Enumerated Values

14.4.2 Figure 30 shows how to represent values of enumerated types, which have a discrete set of possible values, only one of which is provided by an instance of these classes (definition of the complete set of possible values is out of scope for this specification).

14.4.2.1 *EnumeratedStringValue* — Represents a single value from a string-based enumeration.

14.4.2.2 *EnumeratedStringValue Attribute Definition Table*

Table 73 EnumeratedStringValue Attribute Definition

Attribute Name	*Definition*	*Form*
stringEnumVal	The value of the enumerated item.	Text, restricted to one of the legal values of the enumeration.

14.4.2.3 *EnumeratedIntegerValue* — Represents a single value from an integer-based enumeration.

14.4.2.4 *EnumeratedIntegerValue Attribute Definition Table*

Table 74 EnumeratedIntegerValue Attribute Definition

Attribute Name	*Definition*	*Form*
intEnumVal	The value of the enumerated item.	Integer, restricted to one of the legal values of the enumeration.