Table 3 Objects Embedded in SAC Object and Identifiers

Object Name	*Object Identifier (tag)*	*Support Required in a Device*	*Comment*
Assembly	AsmIn*	No	Zero or more instances per device allowed.**
Local Link	LnkIn*	No	Zero or more instances per device allowed.**

* "In" is used to indicate the instance number of the object; "n" is a non-negative integer.

** The specification of the number of Assembly or Local Link object instances allowed per device may be further constrained by the appropriate sensor/actuator network specification.

7.2.1 *SAC Object Attributes* — All required and "common optional" SAC object attributes are listed in Table 4. Note that many of the attributes are in text "human-readable" forms. Information provided in this table includes, for each attribute: (1) an attribute identifier tag; (2) an indication of user access (via the network) to the attribute, (i.e., an indication of whether the attribute is network-settable); and (3) an attribute type. Required attributes must be supported in all SAC object instantiations. Optional attributes may or may not be supported; a further specification of the requirement of support for these optional attributes can be found in Section 7.2.3 and in an appropriate sensor/actuator network specific device model specification.

Table 4 SAC Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Last Calibration Date	SacA1	RW	No	date
Next Calibration Date	SacA2	RW	No	date
Expiration Timer	SacA3	RW	No	signed integer
Expiration Warning Enable	SacA4	RW	No*	Boolean
Run Hours	SacA5	R	No	unsigned integer
Reserved	SacA7-SacA64	—	—	Reserved for future expansion.

* The Expiration Warning Enable attribute is required if the Expiration Timer attribute is supported.

7.2.1.1 *Last Calibration Date* (Optional) — An attribute which identifies the date the device was last calibrated. The attribute is formatted as a date defined in Section 5.2.

7.2.1.2 *Next Calibration Date* (Optional) — An attribute which identifies the date the device is scheduled for the next calibration. The attribute is formatted as a date defined in Section 5.2.

7.2.1.3 *Expiration Timer* (Optional) — An attribute which identifies the number of run hours remaining until the next recommended calibration. The attribute is an unsigned integer with a resolution of 1 hour.

7.2.1.4 *Expiration Warning Enable* (Optional) — An attribute which specifies whether the calibration expiration timer will set a specific warning status bit in the exception status attribute of the DM object instance (specifically the "calibration expiration" exception bit — see Section 7.3.1.14).

The expiration warning enable attribute is Boolean and can take on one of the following values:

0 = Disable

1 = Enable

7.2.1.5 *Run Hours* (Optional) — An attribute which identifies the number of hours that the device has been powered ON. The attribute is an unsigned integer with a resolution of 1 hour.

7.2.1.6 *Initial and Default Values* — The initial and default values for relevant SAC attributes are given in Table 5:

Table 5 SAC Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Last Calibration Date	LVV	0,0,0*	
Next Calibration Date	LVV	0,0,0	
Expiration Timer	LVV	0	
Expiration Warning Enable	LVV	0	
Run Hours	LVV	0	

* Corresponding to year, month, and day; see Section 5.2.

7.2.2 *SAC Object Services* — All SAC object instances must provide the services listed in Table 6 (the SAC object is the "service provider" of these services). SAC services are identified with a service identifier of the form "SacSn", where "n" is a positive decimal number. SAC service identifiers SacS1 through SacS3 are used by this common device model standard. SAC service identifiers SacS4 through SacS32 are reserved.

Table 6 SAC Services

Service	*Service Identifier*	*Required*	*Type*	*Description*
Reset	SacS1	Yes	R	Used to place object in INITIALIZING state.
Abort	SacS2	Yes	R	Used to place object in ABORT state.
Recover	SacS3	Yes	R	Used to move object from ABORT state to RECOVERING state.
Get Attribute	SacS4	Yes	R	Used to read object attribute.
Set Attribute	SacS5	Yes	R	Used to modify object attribute.
Operate	SacS6	Yes	R	Used to move object to the OPERATING state from INITIALIZING or RECOVERING state.
Restore Default	SacS7	No	R	Used to restore object attributes to their default values.
Publish Attribute	SacS8	No	N	Used to proactively report an object attribute value.
	<SacS32	–	–	Reserved for future extensions of the CDM.

7.2.2.1 *Reset (Service Identifier: SacS1)* — Used to place the SAC object in its INITIALIZED state. The description of this state is device-specific. There are no parameters specified for this service.

7.2.2.2 *Abort (Service Identifier: SacS2)* — Used to place the SAC object in its ABORT state. The description of this state is device-specific. There are no parameters specified for this service.

7.2.2.3 *Recover (Service Identifier: SacS3)* — Used to move the SAC object from an ABORT state to its RECOVERED state. The description of this state is device-specific. There are no parameters specified for this service.

Additional services and/or service parameters may be provided by a SAC object that are device-specific and/or implementation-specific. These services may be Notification or Request type services (see Section 6.4.1). A complete definition of additional services and service parameters that are device-specific may be found in an appropriate sensor/actuator network specific device model specification.

7.2.2.4 *Get Attribute* — This service is used to read the value of an object instance attribute over the network. Table 7 describes the parameters specified for this service.

Table 7 SAC Object Get Attribute Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Attribute ID	M	C	Network-Specific	Attribute Identifier of the attribute whose value is being requested.
Attribute Value	–	M	Context-Specific	Value of the attribute requested.

7.2.2.5 *Set Attribute* — This service is used to modify the value of an object instance attribute over the network which has been identified as modifiable. Table 8 describes the parameters specified for this service.

Table 8 SAC Object Set Attribute Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Attribute ID	M	C	Network-Specific	Attribute Identifier of the attribute whose value is being modified.
Attribute Value	M	C	Context-Specific	Value of the attribute to modify.

7.2.2.6 *Operate* — This service is used to move the object instance to the OPERATING state from the INITIALIZING or RECOVERING state There are no parameters specified for this service. Operate requests are issued to all S, A, and C objects.

7.2.2.7 *Restore Default* (Optional) — This service is used to restore attributes of this object instance to their default values. Table 9 describes the parameters specified for this service. For Restore Default Conditions = 2, a Restore Default Condition = 1 request is issued to all S, A, and C objects.

Table 9 SAC Object Restore Default Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Restore Conditions	M	C	byte, enumerated	0 = Restore specific attribute of this object 1 = Restore this object 2 = Restore all objects of this device
Attribute ID	C	C	Network-Specific	Attribute ID of the attribute whose value is being restored. This parameter is only used for Restore Conditions = 0.

7.2.2.8 *Publish Attribute* (Optional) — Used to proactively communicate a SAC object attribute. Table 10 describes parameters specified for this service.

Table 10 SAC Publish Attribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf**	*Description*
Attribute ID	M	–	Attribute Identifier of the attribute whose value is being reported.
Attribute Value	M	–	Value of attribute being reported.

*The utilization of a service response message and its format is outside the scope of this document.

7.2.3 *SAC Instance Behavior* — SAC object instance behavior that is common to all devices is illustrated in Figure 5. Associated states and state transitions are described in Tables 11 and 12. The following also applies to SAC object behavior:

A SAC object service may be requested internally by the SAC object.

Upon SAC object instantiation or upon receipt of a reset request, the SAC responds by entering or staying in its INITIALIZED state.

Upon receipt of an abort request, the SAC object responds by entering or staying in its ABORT state.

Upon receipt of a recover request, the SAC object responds by transitioning to its RECOVERED state if and only if it is currently in its ABORT state. Otherwise, it responds with an error indication.

Additional behavior may be exhibited by a SAC object that is device-specific. A complete definition of this additional SAC object behavior may be found in an appropriate sensor/actuator network specific device model specification.

All SAC services have the same behavior associated with invalid service requests unless otherwise indicated. Common service error responses are listed in Appendix 2.

If a Set Attribute request is received over the network for an attribute that is not network-settable, or if the value in the SetAttribute request is beyond the specified range allowed for the value of the specified attribute, the attribute value is not changed and an error service response is generated.

When the SAC object instance is INITIALIZED, all attributes are set to their appropriate initial values (as indicated in their associated object instance attribute descriptions). Initial values could correspond to those retrieved from the device's non-volatile memory and reflect, in large part, the values set in a previous NORMAL OPERATING state.

Additionally, whenever the SAC object instance enters the INITIALIZED state, it must issue a Reset service request to all Sensor, Actuator, and Controller object instances. Each object instance must then respond with a Pass or Fail response indication upon completion of its respective INITIALIZING application process. The SAC object instance must coordinate these responses in order to determine whether the device is qualified to enter the OPERATING state. If qualified, an Execute service request is issued to the DM object instance. If not qualified, an exception condition is reported to the DM object instance (see Section 7.3).

When the SAC object instance is recovering from an ABORT state, all attributes are set to their appropriate values (as determined by the associated object instance recovering application process defined by the manufacturer). Additionally, the SAC object instance must issue a Recover service request to all Sensor, Actuator, and Controller object instances. Each object instance must then respond with a Pass or Fail response indication upon completion of its respective recovering application process. The SAC object instance must coordinate these responses in order to determine whether the device is qualified to enter the OPERATING state. If qualified, an Execute service request is issued to the DM object instance. If not qualified, an exception condition is reported to the DM object instance.

The SAC object optionally includes an Expiration Timer attribute. The value of this attribute is decremented at a rate described by the "ExpirationTimer" attribute format definition. Whenever the "ExpirationTimer" attribute is less than or equal to zero, a warning exception condition exists and is reported to the DM object instance (utilizing the PublishAttribute service) provided the "ExpirationWarningEnable" attribute is set to Enable. When the "ExpirationWarningEnable" attribute is set to Disable, the "ExpirationTimer" attribute continues to decrement; however, a warning exception condition will not be reported to the DM object instance.

The "ExpirationTimer" attribute will continue to be decremented to values less than zero. Therefore, a negative number indicates an elapsed time past expiration. When this attribute reaches its most negative value, it maintains this value and no longer decrements.

The "ExpirationTimer" is also maintained by the object while in the ABORT state. In the RECOVERING state, the current status of the device is evaluated and any exception conditions that exist are reported to the DM object instance.

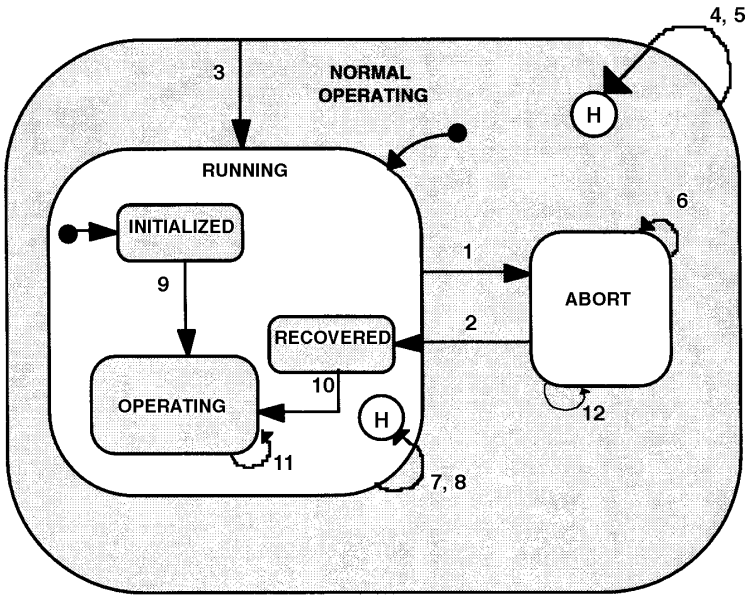

Figure 5
SAC Object Instance Behavior*

NOTE: Only generic behavior is indicated in this chart. A complete state chart necessarily requires the inclusion of device-specific behavior.

* Transitions 4 through 8, 11, and 12 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

Table 11 SAC Object Instance Behavior State Description

State	*Description*
NORMAL OPERATING	Object instance exists. Object services can be processed.
RUNNING	This is the entry sub-state to NORMAL OPERATING. Object instance is not in its ABORT state.
INITIALIZED	This is the entry sub-state to NORMAL OPERATING and RUNNING. Object instance exists and has been initialized. All attributes set to appropriate initial values (as indicated in an appropriate sensor/actuator network specific device model specification).
RECOVERED	Object instance is in an abort recovered state.
OPERATING	This represents the entire collection of sub-states within RUNNING except INITIALIZED and RECOVERED. The further delineation of this sub-state and transition into this sub-state is outside the scope of this document.
ABORT	Object instance is in an aborted state; a detailed description of this state is outside the scope of this document.

Table 12 SAC Object Instance Behavior State Transition Matrix*, **

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
1	RUNNING	Abort request.	ABORT	Abort. Abort response.	ABORT state is device-specific.
2	ABORT	Recover request.	RECOVERED	Recover. Recover response.	RECOVERED state is device-specific.
3	NORMAL OPERATING	Reset request.	RUNNING	Reset. Reset response.	Valid for all sub-states of NORMAL OPERATING. Move to RUNNING/INITIALIZED.
4	NORMAL OPERATING	Invalid service request.	NORMAL OPERATING	Error response.	Valid for all substates of NORMAL OPERATING.

5	NORMAL OPERATING	GetAttribute, SetAttribute, or RestoreDefault request.	NORMAL OPERATING	Get or Set appropriate attribute or restore defaults.	Valid for all substrates of NORMAL OPERATING.
6	ABORT	Abort request, Operate request.	ABORT	Error response.	Invalid request in this state.
7	RUNNING	Recover request.	RUNNING	Error response.	Recover service can only be invoked while in ABORT state.
8	RUNNING	PublishAttribute.	RUNNING	Attribute identified by attribute ID in PublishAttribute service is published.	Notification service.
9	INITIALIZED	Operate request or Device-specific.	OPERATING	Device-specific.	Behavior associated with this transition may be further specified in an appropriate sensor/actuator network specific device model specification.
10	RECOVERED	Operate request or Device-specific.	OPERATING	Device-specific.	Behavior associated with this transition may be further specified in an appropriate sensor/actuator network specific device model specification.
11	OPERATING	Operate request.	OPERATING	Error response.	Invalid request in OPERATING state.
12	ABORT	PublishAttribute.	ABORT	None.	Attributes cannot be published while in ABORT state.

* Transitions 4 through 8, 11, and 12 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

** Note: Only generic behavior is indicated in this table.

7.2.4 *Objects Embedded in SAC Object* — The object types that are embedded in the SAC object are listed in Table 3. The structure and behavior of instances of these object types are detailed in the remainder of this section.

7.2.4.1 *Assembly Object* — Assembly object instances may be used to provide a mechanism for grouping more than one attribute from one or more object instances in a device into a single data structure for communication over the network.

7.2.4.1.1 *Assembly Object Attributes* — All required and "common optional" Assembly object attributes are listed in Table 13.

Table 13 Assembly Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Data	AsmA1	Context-Specific	Yes	Context-Specific

7.2.4.1.1.1 *Data* — An attribute which contains the assembled data. The structure of this data is context-specific.

7.2.4.1.2 *Assembly Object Services* — The Assembly object instance provides two services: GetAttribute and SetAttribute.

7.2.4.1.2.1 *Get Attribute* — This service is used to read the value of an object instance attribute over the network. Table 14 describes the parameters specified for this service.

Table 14 Assembly Object Get Attribute Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Attribute ID	M	C	Network-Specific	Attribute Identifier of the attribute whose value is being requested.
Attribute Value	—	M	Context-Specific	Value of the attribute requested.

7.2.4.1.2.2 *Set Attribute* — This service is used to modify the value of an object instance attribute over the network which has been identified as modifiable. Table 15 describes the parameters specified for this service.

Table 15 Assembly Object Set Attribute Parameter Definition

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Attribute ID	M	C, =	Network-Specific	Attribute Identifier of attribute whose value is to be modified. Inclusion of Attribute in Rsp/Conf. is optional.
Attribute Value	M	C	Context-Specific	Value to which the attribute is to be modified. Inclusion of Attribute in Rsp/Conf. is optional. Conditions under which the value may differ from that supplied in the associated Req/Ind. message (e.g., negative response) are not defined in this document.

Additional services and/or service parameters may be provided by an Assembly object instance that are device-specific and/or implementation-specific. These services may be Notification or Request type services (see Section 6.4.1). A complete definition of additional services and service parameters that are device-specific may be found in an appropriate sensor/actuator network specific device model specification.

7.2.4.1.3 *Assembly Object Behavior* — The behavior exhibited by the Assembly object instance is in support of the GetAttribute and SetAttribute services. That is, when a GetAttribute service request is received for the attribute "Data", the response contains the "Data" attribute value. Similarly, when a SetAttribute service request is received for the attribute "Data", the value of "Data" is modified, if indeed the attribute is modifiable, and a successful service response is returned; otherwise, a service not successful is returned.

7.2.4.2 *Local Link Object* — Local Link object instances may be used to "link" an attribute of one object instance to an attribute of another object instance.

7.2.4.2.1 *Local Link Object Attributes* — All required and "common optional" Local Link object attributes are listed in Table 16.

Table 16 Local Link Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Source Object Class	LnkA1	Context-Specific	N	Context-Specific
Source Object Instance	LnkA2	Context-Specific	N	Context-Specific
Source Object Attribute	LnkA3	Context-Specific	Y	Context-Specific
Destination Object Class	LnkA4	Context-Specific	N	Context-Specific
Destination Object Instance	LnkA5	Context-Specific	N	Context-Specific
Destination Object Attribute	LnkA6	Context-Specific	Y	Context-Specific
Commit	LnkA7	Context-Specific	N	Boolean

7.2.4.2.1.1 *Source Object Class* — Where applicable, the class of the object instance from which the source attribute is to be linked.

7.2.4.2.1.2 *Source Object Instance* — The instance number of the object from which the source attribute is to be linked.

7.2.4.2.1.3 *Source Object Attribute* — The attribute ID of the source attribute to be linked.

7.2.4.2.1.4 *Destination Object Class* — Where applicable, the class of the object instance to which the destination attribute is to be linked.

7.2.4.2.1.5 *Destination Object Instance* — The instance number of the object to which the destination attribute is to be linked.

7.2.4.2.1.6 *Destination Object Attribute* — The attribute ID of the destination attribute to be linked.

7.2.4.2.1.7 *Commit* — An attribute that indicates whether the Link object will perform the link function (the link function is described in Section 7.2.4.2.3). When this attribute is FALSE, the object cannot perform the link function.

7.2.4.2.2 *Local Link Object Services* — There are no common Local Link object services specified. Services and service parameters may be provided by a Local Link object instance that are device-specific and/or implementation-specific. These services may be Notification or Request type services (see Section 6.4.1). A complete definition of additional services and service parameters that are device-specific may be found in an appropriate sensor/actuator network specific device model specification.

7.2.4.2.3 *Local Link Object Behavior* — Local Link object instance behavior common to all instances is as follows. When instantiated, this object performs the link function. That is, it first uses the GetAttribute service capability of the source object instance identified by the Source Object Instance and (possibly) Source Object Class attributes to retrieve the attribute value of the attribute identified by the Source Object Attribute. If the Commit attribute value is TRUE, the object then uses the SetAttribute service capability of the destination object instance identified by the Destination Object Instance and (possibly) Destination Object Class attributes to set the attribute value of the attribute identified by Destination Object Attribute to the (retrieved) value of Source Object Attribute. The object shall also perform the link function any time the Commit attribute value transitions to TRUE. The object shall not write to the destination object instance whenever the value of the Commit attribute is set to FALSE. Issues such as frequency of this link update behavior (while the Commit attribute is set to TRUE, or when the Commit attribute does not exist), type matching of Source and Destination attributes, etc., are beyond the scope of this document.

7.3 *Device Manager (DM) Object* — The DM object is the device component responsible for managing and consolidating the device operation. The DM object houses information about the device (e.g., serial number, device type) so that the device may be uniquely identified remotely via a sensor/actuator network. The DM object is also responsible for providing details of the device status. As such, the responsibility of the DM object includes managing the reporting of exceptions to normal processing as alarms or warnings. Note that it is outside the scope of this document to specify the detailed conditions which initiate alarms and warnings. Exception handling that is common to all devices on the network is described here. This behavior and structure may be extended to handle exceptions which are specific to certain devices or are proprietary to individual manufacturers. Extensions for specific devices would be described in a sensor/actuator network specific device model for those devices.

A device shall contain exactly one instantiation of a DM object.

7.3.1 *DM Object Attributes* — All required and "common optional" DM object attributes are listed in Table 17. Note that many of the attributes are in text "human-readable" forms. Information provided in this table includes, for each attribute: (1) an attribute identifier tag; (2) an indication of user "write" access (via the network) to the attribute, (i.e., an indication of whether the attribute is network-settable); and (3) an attribute type and maximum length. (Note that in many cases it may be desirable to limit the length of these attributes.) Required attributes must be supported in all DM object instantiations. Optional attributes may or may not be supported; a further specification of the requirement of support for these optional attributes can be found in Section 7.3.3 and in an appropriate sensor/actuator network specific device model specification. DM attributes are identified with an attribute identifier of the form "DmAn" where "n" is a positive decimal number. DM attribute identifiers DmA1 through DmA14 are used in this common device model standard. DM attribute identifier numbers DmA15 through DmA64 are reserved.

Table 17 Device Manager Instance Attributes*

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	DmA1	RO	Y	Text, 8 characters maximum
Standard Revision Level	DmA2	RO	Y	Text, 10 characters maximum
Device Manufacturer Identifier	DmA3	RO	Y	Text, 20 characters maximum
Manufacturer Model Number	DmA4	RO	Y	Text, 20 characters maximum
Software or Firmware Revision Level	DmA5	RO	Y	Text, 8 characters maximum
Hardware Revision Level	DmA6	RO	Y	Text, 8 characters maximum
Serial Number (optional)	DmA7	RO	N	Text, 30 characters maximum
Device Configuration (optional)	DmA8	RO	N	Text, 50 characters maximum
Device Status	DmA9	RO	Y	unsigned integer, enumerated, see text
Reporting Mode	DmA10	RW	Y	Formatted byte, see text
Exception Status Report Interval (optional)	DmA11	RW	N	unsigned integer, < 65,536

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Exception Status	DmA12	RO	Y	Formatted byte, see text
Exception Detail Alarm (optional)	DmA13	—	N	Structure of
Common Exception Detail	—	—	—	Structure of
Size	—	RO	—	unsigned integer, < 256
Detail	—	RO	—	Ordered list of bytes, see text
Device Exception Detail	—	—	—	Structure of
Size	—	RO	—	unsigned integer, < 256
Detail	—	RO	—	Ordered list of bytes, see text
Manufacturer Exception Detail	—	—	—	Structure of
Size	—	RO	—	unsigned integer, < 256
Detail	—	RO	—	Ordered list of bytes, see text
Exception Detail Warning (Optional)	DmA14	***	N	Structure of***
Visual Indicator	DmA15	RW	N	Byte, enumerated
Alarm Enable	DmA16	RW	N	Boolean
Warning Enable	DmA17	RW	N	Boolean
Exception Detail Type	DmA18	RW	N	Byte, enumerated
Exception Detail Alarm Queue	DmA19	R	N	Ordered list of type Exception Detail Alarm.
Exception Detail Warning Queue	DmA20	R	N	Ordered list of type Exception Detail Warning.
Date and Time	DmA21	RW	N	See Section 7.3.1.23
Date and Time Type	DmA22	RW	N	Enumerated USINT
Reserved for CDM	< DmA32	—	—	Reserved for future CDM DM attribute definitions.

* A “Definition” column is not included. For attribute definitions, see text.

** A value of 00 indicates that this timer is disabled.

*** Same as Exception Detail Alarm.

7.3.1.1 *Device Type* — An attribute which identifies the type of the device on the sensor/actuator network. It is formatted as a text string with a maximum length of 8 characters. The exact content may be defined in an appropriate sensor/actuator network specific device model specification. For example, “MFC” = mass flow controller, “CDG” = capacitance diaphragm gauge.

7.3.1.2 *Standard Revision Level* — An attribute which identifies the most recent version of the sensor/actuator network specific device model to which the device adheres. It is formatted as a text string with a maximum length of 9 characters as “ENNNNNNYY,” where “E” is a constant defined by SEMI, “NNNNNN” is the number of the standard, and “YY” is the last two digits of the year in which the standard/revision was published. If there is no relevant standard or the device is not compliant to the standard, then “NNNNNN” is zero length and “YY” is set to “00” (thus, the attribute is set to “E00”).

7.3.1.3 *Device Manufacturer Identifier* — An attribute which designates the manufacturer of the device. Each manufacturer is responsible for defining this variable for their products in a way which will be unique to that manufacturer and which must be the same for all devices they provide. Formatted as a text string with a maximum length of 20 characters, it will usually be the company name, as in “Acme Instrument Co.”

7.3.1.4 *Manufacturer Model Number* — An attribute which designates the basic model identification number, defined by the manufacturer. It is formatted as a text string with a maximum length of 20 characters.

7.3.1.5 *Device Software or Firmware Revision Level* — An attribute which designates the version of microprocessor code which is contained in the device, defined by the manufacturer. It is formatted as a text string with a maximum length of 5 characters.

7.3.1.6 *Hardware Revision Level* — An attribute which designates the version of the device, excluding the microprocessor code, defined by the manufacturer. It is formatted as a text string with a maximum length of 5 characters.

7.3.1.7 *Serial Number (optional)* — An attribute which designates the number defined and assigned by the manufacturer that uniquely identifies each individual device produced. It is formatted as a text string with a maximum length of 30 characters.

7.3.1.8 *Device Configuration (optional)* — An attribute which designates a manufacturer-defined device configuration (beyond model number). It is formatted as a text string with a maximum length of 50 characters.

7.3.1.9 *Device Status* — An attribute which designates the state of the DM object. The attribute (unsigned integer) values and corresponding object states specified by this attribute are enumerated in Table 18.

Table 18 Device Status Attribute Values

Attribute Value	*DM Object State*
0	Unknown
1	INITIALIZED/SELF TESTING
2	IDLE
3	SELF TEST EXCEPTION
4	EXECUTING
5	ABORT FROM IDLE OR EXECUTING
6	ABORT FROM INITIALIZED/SELF TESTING OR SELF TEST EXCEPTION
7 – 255	Reserved

This attribute is set and updated internally by the DM object as part of the object behavior associated with transition between the various states (see Section 7.3.3 and Figure 6).

7.3.1.10 *Reporting Mode* — A network-settable attribute defined as a single byte containing two nibbles: A low nibble (bits 0 – 3) and a high nibble (bits 4 – 7). The high nibble contains the value for alarm conditions. The low nibble contains the value for warning conditions. The value assigned to each nibble indicates how the alarm/warning status variable is to be reported. The possible attribute values and corresponding reporting modes specified by this attribute are enumerated in Table 19.

Table 19 Reporting Mode Attribute Values for Alarms and Warnings

Reporting Mode	*Attribute High Nibble Value (Alarm)*	*Attribute Low Nibble Value (Warning)*
Request	0	0
RequestLatched (optional)	1	1
EventTriggeredOn (optional)	2	2
EventTriggeredOn/Off (optional)	3	3
TimeTriggered (optional)	4	4
EventOnOrTimeTriggered (optional)	5	5
EventOn/OffOrTimeTriggered (optional)	6	6

The behavior of the DM object associated with this attribute and each of these reporting modes is described in Section 7.3.3.

7.3.1.11 *Exception Status Report Interval (optional)* — A network-settable attribute which specifies the time interval for periodic exception status reporting (when a time-triggered reporting mode is indicated, see definition of Reporting Mode attribute and Section 7.3.3). It is formatted as an unsigned integer whose least significant bit corresponds to a value of 1/100 second. The requirement to store this attribute in non-volatile memory (so that the value upon "power-up" is the last value active) is device-specific. Note that a value of 0.00 indicates that this timer, and thus time-triggered exception reporting, is not enabled (see also Section 7.3.3).

7.3.1.12 *Exception Status* — A single-byte attribute whose value indicates the status of the alarms and warnings for the device (see Table 19). This indication may be provided in one of two methods: Basic or Expanded. In the Basic method, bit seven of the Exception Status attribute is set to zero; all exceptions are reported exclusively through

communication of this Exception Status attribute. The format of bits zero through six in this mode is device-specific; the format may be further specified in an appropriate sensor/actuator network specific device model specification; if it is not specified, then the format of bits zero through six is equivalent to that specified in the expanded mode. In the Expanded method, bit seven of Exception Status attribute is set to one; exceptions are reported through the communication of this Exception Status attribute, formatted as specified in Table 20, and exception detail attributes as indicated by the Exception Status attribute.

Table 20 Exception Status Attribute Value*

	Exception Status Bit Map, Bit 7 set to 0	*Exception Status Bit Map, Bit 7 set to 1*
bit	*Function*	*Function*
0	Device-Specific	ALARM/device-common**
1		ALARM/device-specific
2		ALARM/manufacturer-specific
3	See sensor/actuator network specific device model for further specification	reserved
4		WARNING/device-common**
5		WARNING/device-specific
6		WARNING/manufacturer-specific
7	0 == Basic Method	1 == Expanded Method

* Behavior associated with the reporting of exception status attribute values is described in Section 7.3.3.

** The alarm or warning is not specific to the device type or device type manufacturer.

7.3.1.13 *Exception Detail Alarm and Exception Detail Warning (optional)* — Attributes that relate the detailed status of the alarms or warnings associated with the device. Each attribute is a structure containing three members; these three members respectively relate the detailed status of exceptions that are device-common (i.e., not device-specific), device-specific but not manufacturer-specific, and manufacturer-specific. Each of the three structure members is defined as a structure containing an ordered list (i.e., array) of bytes of length SIZE, and an unsigned integer whose value is SIZE. Each of the bytes in each array has a specific mapping. This mapping is formatted as 8 bits representing 8 independent conditions, whereas a value of 1 indicates that the condition is set (or present), and a value of 0 indicates that the condition is cleared (or not present). Note that if a device does not support an exception detail, the corresponding bit is never set. The bitmaps for alarms and warnings in the corresponding attributes are structured in parallel so that a condition may have either alarm or warning set, depending on severity. If a condition inherently cannot be both alarm and warning, then the parallel bit position corresponding to the other state will remain “0.”

The existence of an exception detail variable structure is dependent on the value of the Exception Status Attribute; the existence of an exception detail variable structure is only required if bit seven of the Exception Status attribute is set to 1 (indicating Expanded mode reporting) and the bit (among bits zero through six) of the Exception Status attribute corresponding to the particular exception type is also set to 1 (see Table 20).

7.3.1.14 *Common Exception Detail (optional)* — This structure relates exception conditions (i.e., alarms or warnings) which are common to all devices on the network. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element Size. For each byte in the Detail field, all bits which are not identified are reserved for future standardization. The first byte in this attribute is CommonExceptionDetail[0]. Additional exception details, if provided, are named CommonExceptionDetail[1], CommonException Detail[SIZE]. The specific exception associated with each of the bitmaps is given in Table 21. The criteria details for each exception condition are outside the scope of this document.

Table 21 Common Exception Detail Attribute Value

	*Common Exception Detail**
bit	*Detail[0]*
0	internal diagnostic exception
1	microprocessor exception
2	EPROM exception
3	EEPROM exception
4	RAM exception
5	communications exception
6	internal real-time exception
7	calibration expiration

	*Common Exception Detail**
bit	*Detail[1]*
0	power supply overcurrent
1	reserved power supply
2	power supply output voltage
3	power supply input voltage
4	routine maintenance due
5	notify manufacturer
6	reset exception
7	reserved

* Note that if a device does not support an exception detail, the corresponding bit is never set.

7.3.1.15 *Device Exception Detail (optional)* — This structure, similar in form to Common Exception Detail, relates exception conditions which are specific to individual devices on the network and are defined in their respective device model documents. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element size. For a detailed description of this attribute, consult the appropriate specific device model documentation.

7.3.1.16 *Manufacturer Exception Detail (optional)* — This structure, similar in form to Common Exception Detail, relates exception conditions which are specific to the manufacturers of individual devices on the network and are defined by them in their product documentation. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element size. For a detailed description of this attribute, consult the appropriate specific device manufacturer documentation.

Additional attributes may be supported by a DM object that are device-specific. A complete definition of any additional attributes may be found in an appropriate sensor/actuator network specific device model specification.

7.3.1.17 *VisualIndicator* (Optional) — An attribute which defines the behavior of a visual indicator allowing the device to be visually located. This attribute is an enumerated byte that can take on one of the following values:

0 = Visual Indicator OFF

1 = Visual Indicator ON

2–63 = Reserved

64–255 = Manufacturer-Specified

For values visual indicator OFF and visual indicator ON, the visual indicator is defined to be a Light Emitting Diode (LED). When the value is visual indicator ON, the device is expected to flash the LED at a frequency of approximately 0.5 to 1.5 Hz at a duty cycle of approximately 30 to 70 percent. The location and labeling of the LED are manufacturer-specified.

7.3.1.18 *AlarmEnable* (Optional) — An attribute which defines whether or not the object will report alarms (as indicated by the ReportingMode, ExceptionStatus, ExceptionDetailType, and ExceptionDetailAlarm attributes). When this attribute is set to TRUE, then alarming is enabled. When this attribute is set to FALSE, all DM exception attributes are set and maintained at values that indicate a no alarm state. Thus bits 0 through 2 of the ExceptionStatus attribute are set and maintained at zero, and any ExceptionDetailAlarm attribute bytes are set and maintained at a value of zero.

7.3.1.19 *WarningEnable* (Optional) — An attribute which defines whether or not the object will report warnings (as indicated by the ReportingMode, ExceptionStatus, ExceptionDetailType, and ExceptionDetailWarning attributes). When this attribute is set to TRUE, then warning exception reporting is enabled. If this attribute is set to FALSE, all DM exception attributes are set and maintained at values that indicate a no warning state. Thus, bits 4 through 6 of the ExceptionStatus attribute are set and maintained at zero, and any ExceptionDetailWarning attribute bytes are set and maintained at a value of zero.

7.3.1.20 *ExceptionDetailType* (Optional) — An attribute consisting of two enumerated nibbles that designate the method of alarm and warning detail reporting respectively that is supported. The possible attribute values and corresponding reporting mode support indications are enumerated in Table 22. Note that if this attribute is not supported, the reporting mode is either zero (basic) or one (expanded) as indicated in bit seven of the ExceptionStatus attribute.

Table 22 ExceptionDetailType Attribute Values for Alarm and Warning Reporting Methods

Reporting Method	*Attribute High Nibble Value (Alarm)*	*Attribute Low Nibble Value (Warning)*
Basic	0	0
Expanded	1	1
Queued	2	2
Reserved for Future Extension of CDM	3 – 15	3 – 15

7.3.1.21 *ExceptionDetailAlarmQueue* (Optional) — An attribute consisting of an ordered list of alarm events; the data structure of the individual alarm events (i.e., queue elements) is determined by the values of the ExceptionStatus attribute; if bit 7 of the ExceptionStatus attribute indicates a Basic Mode reporting data format then each queue element is of the same data type as ExceptionStatus; if bit 7 of the ExceptionStatus attribute indicates an Expanded Mode reporting data format then each queue element is of the same data type as ExceptionDetail (Alarm/Warning). The behavior of this queue in storing and reporting alarms is described in Section 7.3.3.

7.3.1.22 *ExceptionDetailWarningQueue* (Optional) — An attribute consisting of an ordered list of warning events; the data structure is determined as described in Section 7.3.1.21. The behavior of this queue in storing and reporting warnings is described in Section 7.3.3.

7.3.1.23 *Date and Time* — A structure that relates the current time as perceived by the device. The form of the structure depends on the value of the *Data and Time Type* attribute as follows:

Date and Time Type	*Date and Time Structure*	*Semantics*
0 [default]	Struct of: UDINT UINT	{*Standard Time and Date*} (6 Bytes) Number of milliseconds since midnight Number of days since 01 January, 1972
1	Struct of: UDINT UINT INT	{*Standard Time and Date with Offset*} (8 Bytes) Number of milliseconds since midnight Number of days since 01 January, 1972 Number of minutes displacement from the Greenwich Meridian
2	Struct of: LINT	{*Universal Time Coordinated* (UTC)} (8 Bytes) Number of 100 nanoseconds since 15 October, 1582, 00:00:00
3	Struct of: LINT INT	{*UTC with Time Displacement Factor*} (10 Bytes) Number of 100 nanoseconds since 15 October, 1582, 00:00:00 Number of minutes displacement from the Greenwich Meridian
4 - 63	Reserved	
64 - 127	Specific Device Model definitions as required	
128 - 255	Manufacturer specified as required	

7.3.1.23.1 If the *Date and Time Type* attribute is not supported, the default value of zero (0) is assumed.

7.3.1.23.2 The *Standard Time and Date* (Type 0) uses 32 bits to represent the number of milliseconds since midnight followed by 16 bits to represent the number of days since 01 January, 1972. The reference for this format is either Greenwich Mean Time or Local Time as defined by the user. This format has a range of 01 January, 1972 to 06 June, 2151.

7.3.1.23.3 The *Standard Time and Date with Offset* (Type 1) uses the above format followed by 16 bits to represent the number of minutes displacement from the Greenwich Meridian. Meridians to the East are positive, while those to the West are negative.

7.3.1.23.4 The *Universal Time Coordinated* (UTC) (Type 2) is based on the format used by the WWV radio station of NIST. It uses 64 bits to represent the number of 100 nanoseconds since 15 October, 1582, 00:00:00. The reference for this format is always Greenwich Mean Time. This format has an approximate range of BC 27,000 to AD 30,000.

7.3.1.23.5 The *UTC with Time Displacement Factor* (TDF) (Type 3) uses the above format followed by 16 bits to represent the number of minutes displacement from the Greenwich Meridian. Meridians to the East are positive, while those to the West are negative.

7.3.2 *DM Object Services* — All DM object instances must provide the services listed in Table 23. (The DM object is the "service provider" of these services.) Note that all of these services are Request services, with the exception of the PublishAttribute service (which is a notification service). DM services are identified with a service identifier of the form "DmSn", where "n" is a positive decimal number. Dm service identifiers DmS1 through DmS8 are used by this common device model standard. DM service identifiers DmS9 through DmS32 are reserved. Service parameters that are specified in this standard are summarized in Table 24.

Table 23 DM Object Service Resource Definition

Service	*Service Identifier*	*Required*	*Type*	*Description*
Reset	DmS1	Y	R	Used to place object in INITIALIZED state.
Abort	DmS2	Y	R	Used to place object in ABORT state.
Recover	DmS3	Y	R	Used to move object from ABORT state to RECOVERED state.
GetAttribute	DmS4	Y	R	Used to read object attribute.
SetAttribute	DmS5	Y	R	Used to modify object attribute.
Execute	DmS6	Y	R	Used to move object from IDLE state to EXECUTING state.
PerformDiagnostics	DmS7	Y	R	Used to instruct object to perform diagnostic test.
PublishAttribute	DmS8	N	N	Used to proactively report exception status attribute.
Lock	DmS9	N	R	Used to restrict access to READ only attributes.
Unlock	DmS10	N	R	Used to make READ only attributes modifiable.
Get Exception Queue	DmS11	N	R	Used to retrieve all, or a specified number of, exception events from the front of the ExceptionDetailAlarm/WarningQueue.
Clear Exception Queue	DmS12	N	R	Used to clear all exception events from the ExceptionDetail Alarm or WarningQueue.
Reserved by CDM	DmS13 – DmS32	—	—	Reserved for CDM expansion.

Table 24 DM Object Service Parameter Dictionary

Parameter	*Form*	*Description*
AttributeID	Enumerated	DM object attribute identifier (see Table 17)
AttributeValue	Context-specific (see Table 17)	Value of DM object attribute.
TestID	Non-negative integer, < 256	Type and possibly detail of diagnostic test to be performed.

A detailed description of these services follows.

7.3.2.1 *Reset (Service Identifier: DmS1)* — Used to place the DM object in its INITIALIZED state. There are no parameters specified for this service.

7.3.2.2 *Abort (Service Identifier: DmS2)* — Used to place the DM object in its ABORT state. There are no parameters specified for this service.

7.3.2.3 *Recover (Service Identifier: DmS3)* — Used to move the DM object from its ABORT state to a recovered state (which is the IDLE state for this object). There are no parameters specified for this service.

7.3.2.4 *GetAttribute (Service Identifier: DmS4)* — Used to read a DM object attribute.

GetAttribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttributeID	M	C**, =	Attribute Identifier of attribute whose value is being requested. Inclusion of attribute in Rsp/Conf. is optional.
AttributeValue	—	M	Value of attribute requested.

* See Section 6.4.3 for a further description of terminology used in this table.

** The determination of whether Attribute ID is supplied in the service response/confirm is outside the scope of this document.

7.3.2.5 *SetAttribute (Service Identifier: DmS5)* — Used to modify a DM object attribute.

Set_Attribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
AttributeID	M	C, =	Attribute Identifier of attribute whose value is to be modified. Inclusion of Attribute in Rsp/Conf. is optional.
AttributeValue	M	C	Value to which the attribute is to be modified. Inclusion of Attribute in Rsp/Conf. is optional. Conditions under which the value may differ from that supplied in the associated Req/Ind. message (e.g., negative response) are not defined in this document.

* See Section 6.4.3 for a further description of terminology used in this table.

7.3.2.6 *Execute (Service Identifier: DmS6)* — Used to move the DM object from its IDLE state to its EXECUTING state. Note that this service may be internally generated by the DM object. There are no parameters specified for this service.

7.3.2.7 *PerformDiagnostics (Service Identifier: DmS7)* — Used to instruct the DM object to perform a diagnostic test. A diagnostic test is either of type "common" or "device-dependent." "Common" diagnostic tests could include: RAM, EPROM, non-volatile memory, and communications. The structure of "common" type diagnostic tests is implementation-specific. All detail of "device-dependent" diagnostics is outside the scope of this document. The following table describes parameters specified for this service.

Perform_Diagnostics Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Description*
TestID	M	C, =	Indicates the type and possibly detail of the test to be performed.

* See Section 6.4.3 for a further description of terminology used in this table.

Additional services and/or service parameters that are device-specific and/or implementation-specific may be provided by a DM object. These services may be Notification or Request type services (see Section 6.4.1). A complete definition of additional services and service parameters that are device-specific may be found in an appropriate sensor/actuator network specific device model specification.

7.3.2.8 *PublishAttribute (Optional — Service Identifier: DmS8)* — Used to proactively communicate a DM object attribute. The following table describes parameters specified for this service.

PublishAttribute Service Message Definition Table*

Parameter	*Ind*	*Description*
AttributeID	M	Attribute Identifier of attribute whose value is being reported.
AttributeValue	M	Value of attribute being reported.

* The utilization of a service response message and its format is outside the scope of this document.

7.3.2.9 *Lock* (Optional) — This service is used to restrict network access to all READ only attributes of all objects of this device. This service guarantees that the READ only attributes of this device are not modified over the network. There are no parameters specified for this service.

7.3.2.10 *Unlock* (Optional) — This service is used to make READ only attributes of objects of this device modifiable over the network. Not only is support of this object optional, but the degree to which it is supported is also specified by the manufacturer (i.e., only some attributes may become "unlocked"). Table 25 describes the parameters specified for this service.

Table 25 DM Object Unlock Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
Password Key	C	C	Manufacturer-Specific	Password may be specified to restrict access or to provide a manufacturer-specified security scheme.

7.3.2.11 *GetExceptionQueue* (Optional) — This service is used to read all or portions of the AlarmQueue and WarningQueue elements. Table 26 describes the parameters specified for this service.

Table 26 DM Object GetExceptionQueue Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
ExceptionType	M	C**, =	Byte, enumerated0 = Alarm1 = Warning	Used to indicate which exception queue is being queried.
ElementNumb	M	M	unsigned integer < 65,536	Used to indicate the number of Exception queue events to be returned with the service response. A zero value implies that all elements in the queue should be returned.
ExceptionList	—	M	List of elements; element data type. Dependent on the Reporting Mode (Basic or Expanded — See Sections 7.3.1.12, 7.3.1.19, 7.3.1.20, and 7.3.3).	An array of exception queue events of size ElemNumb returned with the response.
Clear	M	C	Boolean	Used to indicate whether or not the exceptions are to be removed from the exception queue upon reporting.

7.3.2.12 *ClearExceptionQueue* (Optional) — This service is used to clear all elements out of either the AlarmQueue or WarningQueue. Table 27 describes the parameters specified for this service. If the ExceptionType attribute is set to 0, 1, or 2 with the service request, the Alarm, Warning, or both queues respectively are cleared completely of their members.

Table 27 DM Object Clear ExceptionQueue Parameter Definitions

Parameter	*Request/ Indication*	*Response/ Confirmation*	*Form*	*Description*
ExceptionType	M	C**, =	Byte, enumerated 0 = Alarm 1 = Warning 2 = Both	Used to indicate which exception queue(s) is/are to be cleared.

7.3.3 *DM Object Behavior* — The DM object behavior is illustrated in Figure 6 and in Tables 28 and 29.

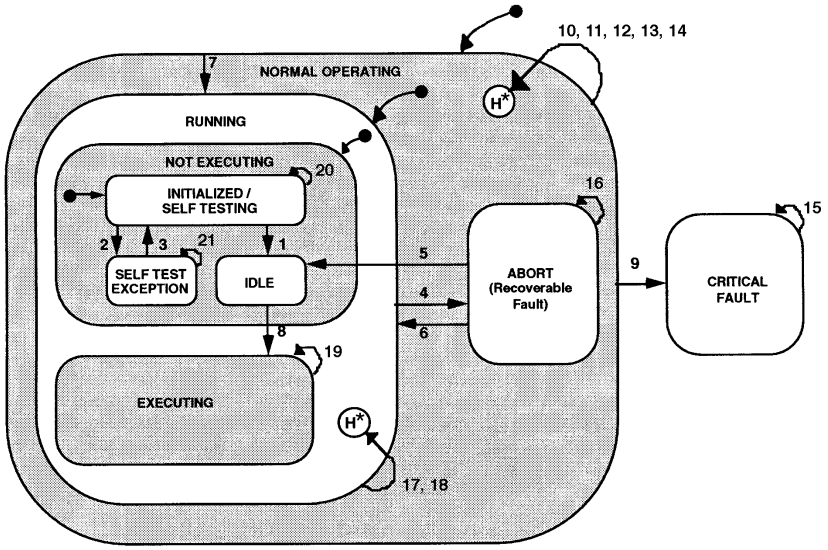

Figure 6
DM Object Instance Behavior*

*Transitions 10 through 21 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

Table 28 DM Instance Behavior State Description

State	*Description*
NORMAL OPERATING	Object instance exists. Object services can be processed.
RUNNING	This is the entry sub-state to NORMAL OPERATING. Object instance is not in an aborted state. Object is capable of reporting attribute values, including exceptions.
NOT EXECUTING	Device is not executing (e.g., it is not performing its device-specific function). A detailed description of this state is outside the scope of this document.
	This state may be further specified in an appropriate sensor/actuator network specific device model specification.
EXECUTING	Device is executing (e.g., it is performing its device-specific function). A detailed description of this state is outside the scope of this document.
	This state may be further specified in an appropriate sensor/actuator network specific device model specification.
INITIALIZED/SELF TESTING	This is the entry sub-state to NORMAL OPERATING and RUNNING.
	Object instance exists and has been initialized; all attributes have appropriate initial values(as indicated herein and in an appropriate sensor/actuator network specific device model specification).
	Device is performing device-specific and device type-specific tests to determine if it is qualified to be running.
	Note that as this is a sub-state of RUNNING, object is not initialized unless it is capable of reporting attribute values, including exceptions.
SELF TEST EXCEPTION	Object has detected an exception condition during self testing. The details of the exception are stored in the appropriate attribute values of the DM object.
IDLE	Object and device have been initialized and have successfully completed self testing.
ABORT	Object instance is in an aborted state; a detailed description of this state is outside the scope of this document. The object will not initiate communication over the network while in this state.
CRITICAL FAULT	The object (and device) are in a fault state from which there is no recovery. Object services cannot be processed. The conditions required for exit from a critical fault are outside the scope of this document.

Table 29 DM Instance Behavior State Transition Matrix*

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
1	INITIALIZED/ SELF TESTING	DM Object determines internally that the device is initialized and self test passed.	IDLE	Set Device Status attribute to appropriate value.	Device is in an idle state.
2	INITIALIZED/ SELF TESTING	One or more exceptions reported as a result of device testing.	SELF TEST EXCEPTION	Report exceptions through appropriate attribute settings in DM object. Set Device Status attribute to appropriate value.	Conditions resulting in the reporting of a self test exception are device-specific.
3	SELF TEST EXCEPTION	Exception condition cleared.	INITIALIZED/ SELF TESTING	Set Device Status attribute to appropriate value. Initiate self testing procedure from beginning.	Enter RUNNING/NOT EXECUTING/INITIALIZED/SELF TESTING.
4	RUNNING	Abort request.	ABORT	Abort response. Set Device Status attribute to appropriate value.	Abort request may be generated internally (e.g., as a result of Self Test catastrophic failure).
5	ABORT	Recover request; Device Status attribute value == ABORT FROM IDLE OREXECUTING.	IDLE	Recover response. Set Device Status attribute to appropriate value.	Enter IDLE state.
6	ABORT	Recover request; Device Status attribute value == ABORT FROM INITIALIZED/ SELF TESTING OR SELF TEST EXCEPTION.	INITIALIZED/SELF TESTING	Recover response. Initiate self testing procedure from beginning. Set Device Status attribute to appropriate value.	Enter RUNNING/NOT EXECUTING/ INITIALIZED/SELF TESTING.
7	NORMAL OPERATING	Reset request.	RUNNING	Reset response. DM Object is initialized. Set Device Status attribute to appropriate value.	Valid for all sub-states of NORMAL OPERATING. Move to RUNNING/NOT EXECUTING/ INITIALIZED/SELF TESTING.
8	IDLE	Execute service request (this service request may be internally generated).	EXECUTING	Execute response. Set Device Status attribute to appropriate value.	Device is now executing.
9	NORMAL OPERATING	Critical Fault detected.	CRITICAL FAULT	The object (and device) is in a fault state from which there is no recovery. Object services generally cannot be processed.	The mechanism for exit from a critical fault is device-specific.
10	NORMAL OPERATING	GetAttribute or Set Attribute request.	NORMAL OPERATING	Get or Set Attribute appropriate response.	Valid for all sub-states of NORMAL OPERATING.
11	NORMAL OPERATING	PerformDiagnostics request.	NORMAL OPERATING	PerformDiagnostics response. Diagnostic is performed if possible. Exceptions are set within DM object as necessary.	Conditions which are deemed exceptions are device-specific.
12	NORMAL OPERATING	Lock or Unlock request.	NORMAL OPERATING	Adjust write access to READ only attributes of all device objects as appropriate. Lock or Unlock service response.	The mechanism for exit from a critical fault is device-specific.
13	NORMAL OPERATING	GetExceptionQueue or Clear Exception Queue request.	NORMAL OPERATING	GetExceptionQueue response with appropriate exception queue data. Clear appropriate	Valid for all substates of NORMAL OPERATING.

				exception queue and send service response.	
14	NORMAL OPERATING	Invalid service request.	NORMAL OPERATING	Error response to appropriate request.	Valid for all sub-states of NORMAL OPERATING.
15	CRITICAL FAULT	Any service request.	CRITICAL FAULT	none	Object services generally cannot be processed.
16	ABORT	Abort or Execute request.	ABORT	Error response.	Not valid requests in this state.
17	RUNNING	Recover request.	RUNNING	Error response.	Recover request only valid while in ABORT state.
18	RUNNING	Internally generated PublishAttribute Notification Service.	RUNNING	Attribute Identified by Attribute ID in PublishAttribute Service is Published.	Notification only occurs if specific conditions are met (see Section 7.3.3).
19	EXECUTING	Execute Request.	EXECUTING	Error response.	Not a valid request in this state.
20	INITIALIZED/ SELF TESTING	Execute service request.	INITIALIZED/ SELF TESTING	Error response.	Not a valid request in this state.
21	SELF TEST EXCEPTION	Execute service request.	SELF TEST EXCEPTION	Error response.	Not a valid request in this state.

* Transitions 10 through 21 are recursive (i.e., the New State is the same as the Current State (including all nested sub-states)).

The following also applies to DM behavior:

A DM instance service may be requested internally by the DM object.

If a SetAttribute request is received over the network for an attribute that is not network-settable, the attribute value is not changed and an error service response is generated.

When processing an Abort, Recover, or Reset service request, the DM object issues a corresponding Abort, Recover, or Reset service request to the SAC object.

While in the Abort state, the DM object will not initiate communications over the network (e.g., it will not initiate communication of exceptions over the network). It may, however, communicate over the network in response to service requests.

All DM services have the same behavior associated with invalid service requests unless otherwise indicated. Common service error responses are listed in Appendix 2.

The following identifies additional behavior associated with specific values of DM object attributes:

The Device Status attribute value indicates the state of the DM object (see Table 18). It is updated on appropriate state transitions within the DM object (see Table 29). Attribute values 1 through 6 represent valid states. A value of zero indicates that the DM state is unknown; conditions under which a zero value may occur are outside the scope of this document.

The Reporting Mode attribute determines five different conditions under which alarms and warnings are reported through communication of the Exception Status attribute (see Table 19). Both warnings and alarms have separate values within Reporting Mode. This allows alarms to be reported more aggressively than warnings because alarms are presumably more time-critical. The behavior associated with the five Reporting Mode conditions which are used for both alarms and warnings is described in the following paragraphs. Optionality indicates that the mode may or may not be supported; this optionality may be further specified in a sensor/actuator network specific device model and/or device manufacturer specification.

_ *Request* — When a bit in the status attribute transitions from cleared to set, the device will not report until requested. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared. In this mode, a device may go in and out of an exception condition without ever reporting the condition because there has been no request. The device must always respond to a request for exception conditions, regardless of Reporting Mode state.

_ Optional *RequestLatched* — When a bit in the status attribute transitions from cleared to set, the device will not report until requested. When the exception condition no longer exists, the exception status bit and the specific bit in the exception detail attribute are not cleared until they have been requested and reported. This mode guarantees that a brief occurrence of an exception condition will always be reported, barring an unrecoverable communications error. Within each exception variable, active bits will be cleared as they are reported.

_ Optional *EventTriggeredOn* — When a bit in the status attribute transitions from cleared to set, the device will automatically report the exception status attribute. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail variable are cleared.

_ Optional *EventTriggeredOn/Off* — When a bit in the status attribute transitions from cleared to set, the device will automatically report the exception status attribute. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared, and the device will again automatically report the exception status attribute.

_ Optional *TimeTriggered* — If this mode is supported, the Exception Status Timer attribute value indicates the time interval for periodic status reporting. When the indicated time period expires, the device will automatically report the exception status attribute. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared.

_ Optional *EventOnOrTimeTriggered* — This is a logical "or" of EventTriggeredOn mode and TimeTriggered mode. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared.

_ Optional *EventOn/OffOrTimeTriggered* — This is a logical "or" of EventTriggeredOn/Off mode and Time-Triggered mode. When the exception condition no longer exists, both the exception status bit and the specific bit in the exception detail attribute are cleared.

The Reporting Mode attribute value defaults to zero upon object initialization (i.e., request mode), unless otherwise specified in an appropriate sensor/actuator network specific device model specification.

The Reporting Mode attribute is defined as network-settable; however, it is only settable to (valid) values corresponding to supported reporting modes (see Table 19, appropriate sensor/actuator network specific device model specification, and device manufacturer specification). Note that at least one reporting mode, Request, must be supported. If an attempt is made, via a Set_Attribute service request, to set the Reporting Mode attribute to an invalid value, an error service response is returned.

All exceptions are communicated through reporting the values of the Exception Status and, conditionally, the Exception Detail Alarm and Exception Detail Warning attributes. An Exception Detail attribute may only be reported if both bit 7 and the corresponding exception bit of the Exception Status variable are set to 1.

If the Reporting Mode attribute indicates a reporting mode of TimeTriggered, EventOnOrTimeTriggered, or EventOn/OffOrTimeTriggered, the time interval for reporting is the value of the Exception Status Report Interval attribute. Time zero is defined as the time at which the Reporting Mode attribute is last modified utilizing the SetAttribute service. The first report is generated at time zero (not after the first time interval). If this value of the Exception Status Report Interval attribute is zero, then time triggered exception reporting is disabled.

If the Reporting Mode attribute indicates a reporting mode of EventOnOrTimeTriggered, or EventOn/OffOrTimeTriggered, the reporting associated with the occurrence of the appropriate event and the reporting associated with the report interval are independent (e.g., the occurrence of reporting due to an event does not impact the timing associated with reporting according to a specified time interval).

The Exception Status Report Interval attribute value defaults to 0.0 seconds upon object initialization, unless otherwise specified in an appropriate sensor/actuator network specific device model specification.

Additional behavior may be provided by a DM object that is device-specific. A complete definition of this additional behavior may be found in an appropriate sensor/actuator network specific device model specification.

The PublishAttribute service would be used for automatic reporting of the DM object exception status attribute when (1) an exception exists, (2) an automatic reporting mode is indicated by the Reporting Mode attribute (see Section 7.3.1), and (3) the indicated reporting mode is supported by the device (in its current state).

Behavior associated with the AlarmEnable and WarningEnable attributes is as follows. If the alarm/warning attribute is TRUE or if the attribute is not supported, alarm/warning reporting behavior as described elsewhere in this section is not impacted by this attribute. If the alarm/warning attribute is set to FALSE, all alarm/warning DM exception attributes are set and maintained at values that indicate a no alarm state. Thus, bits 0 through 2 of the ExceptionStatus attribute (alarms) or bits 4 through 6 (warnings) are set and maintained at zero, and any ExceptionDetail Alarm/Warning attribute bytes are set and maintained at a value of zero. When the Alarm/WarningEnable is toggled to TRUE, the device immediately determines its alarm/warning state, updates the appropriate DM attributes as necessary, and reports any alarms/warnings according to the mode defined by the ReportingMode attribute.

Behavior associated with the (Alarm and Warning) Exception Queues (see Sections 7.3.1.21 and 7.3.1.22), if supported, is as follows: Upon device initialization (start-up) or reset, both exception queues are cleared. The data type of queue elements is determined by the value of bit 7 of the status attribute (as explained in Section 7.3.1.21). As alarm events are issued to the DM object, the appropriate exception parameters are set (ExceptionStatus and ExceptionDetailAlarm/Warning) and the events are also logged onto the appropriate Alarm or Warning Exception Queue on a first-in, first-out basis. Any alarm or warning clear events are also logged onto these queues in the same manner; these events are logged in the same data format as the exception, but with the data value indicating that the exception no longer exists. The maximum number of elements that are retained by the queue is application-specific. Once the maximum queue length is reached, new exception events depose the oldest exception event from the queue in a first-in, first-out fashion (similar to a shift register). The GetExceptionQueue service may be used to retrieve any number or all exception events from either (alarm or warning exception) queue. If the value of the Clear attribute is TRUE, retrieved events are deposed from the queue; otherwise, they are left in the queue.

Behavior associated with the optional "Date and Time" attribute is as follows. This attribute indicates the current date and time as perceived by the device whenever it is reported. The resolution of the "Time of Day" reported shall be milliseconds unless the "Date and Time Type" attribute is supported and it indicates a resolution other than milliseconds.

7.4 *Sensor, Actuator, and Controller Objects* — The Sensor (S), Actuator (A), and Controller (C) objects are the model components that model the high level sensory, actuation, and/or control element capabilities of the device as viewed from the network. Although the structure and behavior of these elements can vary widely from device to device, there are common aspects of structure and behavior that can be identified. Specifically, there are structure and behavior characteristics common to all active elements (i.e., all S, A, and C objects), common to all elements of an element type (i.e., all S objects, A objects, or C objects), or common to all elements of an element sub-type (e.g., all analog input sensor objects). This commonality of structure and behavior among these object types is depicted in the class generalization hierarchy of Figure 7. With this generalization hierarchy in place, objects inherit structure and behavior from their parent object classes, thus reducing the level of redundancy in description. For example, object instances of the Sensor-AI class inherit (i.e., also have) the attributes, services, and behavior of the Sensor and Active Element parent classes.

Note that the only classes from which accessible object instances can be created in this hierarchy exist at the lowest level (i.e., Sensor-AI, Sensor-BI, Actuator-AO, Actuator-BO, and Controller objects). Also note that the object identifiers for these object instances are designated as indicated in Table 1 (e.g., a Sensor-AI object instance has an Object Identifier of SenI'n', where 'n' is the instance number of the sensor object for the device).

In the following sub-sections, the attributes, services, and behavior at each level of the active element class generalization hierarchy are identified and defined. Note that the specific device model would complete the definition of each sensor, actuator, and controller element of a specific device type through definition of additional structure and behavior for the S, A, and C object instances that collectively realize the sensory/ actuation/control capability portion of the device model.

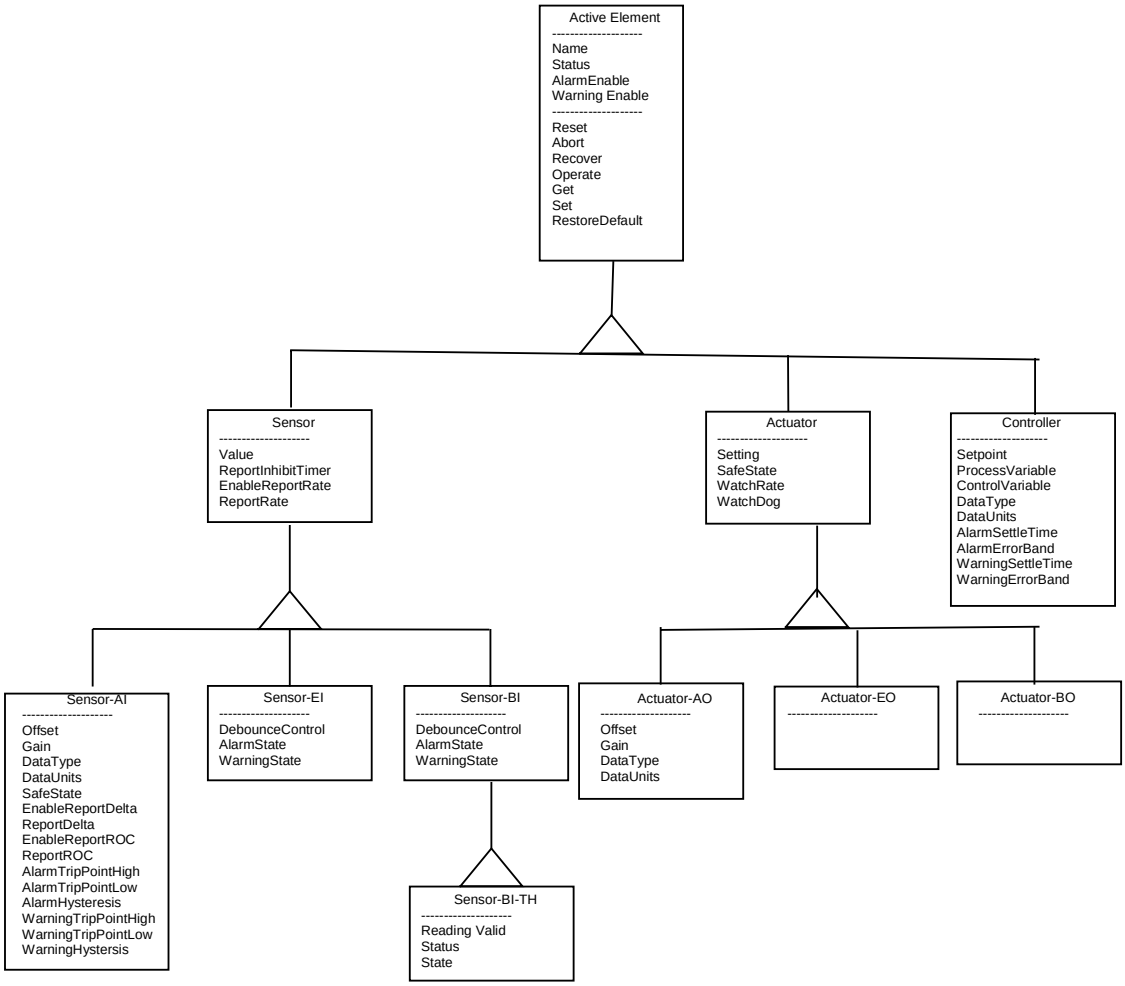

Figure 7
Active Element Class Generalization Hierarchy

7.4.1 *Active Element (AE) Class* — The AE class at the top of the active element generalization hierarchy contains structure and behavior common to all active element instances in a device.

7.4.1.1 *Active Element Class Attributes* — All required and "common optional" AE class attributes are listed in Table 30 and described below. For an explanation of the table format, see Section 7.3.1.

Table 30 AE Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Name	nA1**	RO	N	Text, 16 characters maximum
Status	nA2	RO	Y	Context-specific
AlarmEnable	nA3	RW	N	Boolean
WarningEnable	nA4***	RW	N	Boolean

* A "Definition" column is not included. For attribute definitions, see text.

** Where 'n' is the type of the object instance to which the particular attribute is associated (e.g., Sai, Sbi, Aao, Ado, or C), see Sections 7.4.4 through 7.4.8.

*** ID's nA5 through nA15 are reserved for future AE class attribute definition.

7.4.1.1.1 *Name* — A label used to identify the active element associated with the object instance.

7.4.1.1.2 *Status* — An indication of the state of the active element associated with the object instance. The form and interpretation of the value of this attribute is context-specific.

7.4.1.1.3 *AlarmEnable* — An attribute which specifies whether alarm exception conditions of the active element associated with the object instance are to be reported. Reporting is enabled if this variable is set to TRUE. The method of reporting is beyond the scope of this document.

7.4.1.1.4 *WarningEnable* — An attribute which specifies whether warning exception conditions of the active element associated with the object instance are to be reported. Reporting is enabled if this variable is set to TRUE. The method of reporting is beyond the scope of this document.

7.4.1.2 *Active Element Class Services* — All Active Element common services are listed in Table 31 (the specific AE object is the "service provider" of these services). Note that all of these services are Request services. Service parameters that are specified in this standard are summarized in Table 32.

Table 31 AE Class Service Resource Definition

Service	*Service ID*	*Type*	*Description*
Reset	nS1*	R	Used to place object in INITIALIZED state.
Abort	nS2	R	Used to place object in ABORT state.
Recover	nS3	R	Used to move object from ABORT state to RECOVERED state.
Operate	nS4	R	Used to move object to the operating state from INITIALIZING, or RECOVERING state.
GetAttribute	nS5	R	Used to read object attribute.
SetAttribute	nS6	R	Used to modify object attribute.
RestoreDefault	nS7**	R	Used to restore object attributes to their default values.

* Where 'n' is the type of the object instance to which the particular service is associated (e.g., Sai, Sbi, Aao, Ado, or C), see Sections 7.4.4 through 7.4.8.

** ID's nS8 through nS15 are reserved for future AE class service definition.

Table 32 AE Class Service Parameter Dictionary

Parameter	*Form*	*Description*
AttributeID	Positive Integer	Object instance Attribute Identifier.
AttributeValue	Context-specific	Value of object instance attribute.
RestoreConditions	Byte enumerated	Level of restore requested.

A detailed description of these services follows:

7.4.1.2.1 *Reset* — Used to place the object instance in its INITIALIZED state. There are no parameters specified for this service.

7.4.1.2.2 *Abort* — Used to place the object instance in its ABORT state. There are no parameters specified for this service.

7.4.1.2.3 *Recover* — Used to move the object instance from its ABORT state to a RECOVERED state. There are no parameters specified for this service.

7.4.1.2.4 *Operate* — Used to move the object instance to the OPERATING state from the INITIALIZING or RECOVERING state. There are no parameters specified for this service.

7.4.1.2.5 *GetAttribute* — Used to read a value of an object instance attribute over the network. The following table describes the parameters specified for this service.

GetAttribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Form*	*Description*
AttributeID	M	C**, =	Network-Specific	Attribute Identifier of attribute whose value is being requested. Inclusion of attribute in Rsp/Conf. is optional.
AttributeValue	-	M	Context-Specific	Value of attribute requested.

* See Section 6.4.3 for further description of terminology used in this table.

** The determination of whether Attribute ID is supplied in the service response/confirm is outside the scope of this document.

7.4.1.2.6 *SetAttribute* — Used to modify the value of an object instance attribute over the network which has been identified as modifiable. The following table describes the parameters specified for this service.

SetAttribute Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Form*	*Description*
AttributeID	M	C, =	Network-Specific	Attribute Identifier of attribute whose value is to be modified. Inclusion of Attribute in Rsp/Conf. is optional.
AttributeValue	M	C	Context-Specific	Value to which the attribute is to be modified. Inclusion of Attribute in Rsp/Conf. is optional. Conditions under which the value may differ from that supplied in the associated Req/Ind. message (e.g., negative response) are not defined in this document.

* See Section 6.4.3 for further description of terminology used in this table.

7.4.1.2.7 *RestoreDefault* — Used to restore attributes of this object instance to their default values. The following table describes the parameters specified for this service.

RestoreDefault Service Message Definition Table*

Parameter	*Req/Ind*	*Rsp/Conf*	*Form*	*Description*
Restore Conditions	M	C	byte, enumerated	0 = Restore specific attribute of this object 1 = Restore all attributes of this object 2 – 63 = Reserved 64 – 255 = Application-Defined
AttributeID	C	C	Network-Specific	Attribute ID of the attribute whose value is being restored.

* See Section 6.4.3 for further description of terminology used in this table.

7.4.1.3 *Active Element Class Behavior* — The AE class behavior is illustrated in Figure 8. Associated states and state transitions are described in Tables 33 and 34. The following also applies to AE class behavior:

An AE class service may be requested internally by the AE object instance.

Note that the behavior associated with the Reset, Abort, Recover, Operate, and Restore Default services differs from the SAC object instance behavior in that no service requests are necessarily issued to any other object instances.

Upon instantiation or upon receipt of a reset request, an AE object instance responds by entering or staying in its INITIALIZED state.

The object INITIALIZING application process is described as follows: When a request to initialize is received (e.g., through a Reset service request), all object instance attributes are restored to their initial value; the interpretation of these values and how they are stored/retrieved is beyond the scope of this document. The object performs any manufacturer-specified self tests and diagnostics to determine if it is qualified to enter the OPERATING state. Upon completion of these tests, the appropriate Pass or Fail service response is reported to the requesting object instance. The content, format, and protocol of this service response are specified by the network for requests made over the network.

Upon receipt of an abort request, an AE object instance responds by entering or staying in its ABORT state.

Upon receipt of a recover request, an AE object instance responds by transitioning to its RECOVERED state if, and only if, it is currently in its ABORT state. Otherwise, it responds with an error indication.

The object RECOVERING application process is described as follows: Upon receipt of a Recover service request, the object performs any manufacturer-specified self tests and diagnostics to determine if it is qualified to enter the OPERATING state.

Upon completion, the appropriate Pass or Fail service response is reported to the requesting object instance. The content, format, and protocol of this service response are specified by the network for requests made over the network.

The AlarmEnable attribute determines whether or not alarms are reported through the ExceptionStatus and ExceptionDetail attributes of the DM object (see Section 7.3). If the AlarmEnable attribute is set to the enabled state, then any alarm is reported to the DM object instance upon its occurrence. The WarningEnable attribute works analogously with respect to reporting a warning.

Additional behavior may be exhibited by an AE object instance that is defined elsewhere in its class generalization hierarchy, or is device-specific. A complete definition of AE object type instance behavior includes the appropriate sensor/actuator network specific device model specification.

Behavior associated with receipt of an invalid service request identified in this document applies to all services required for this class. Unless otherwise specified (e.g., in an appropriate sensor/actuator network specific device model specification), behavior associated with the receipt of an invalid service request for additional services defined (elsewhere) for this class shall be the same as that for required services.

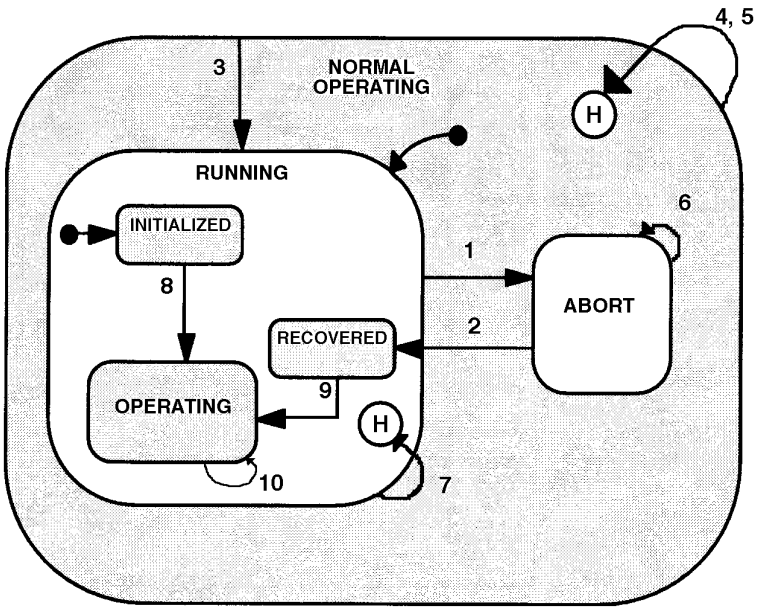

Figure 8
AE Class Behavior

NOTE: Only generic behavior is indicated in this chart. A complete state chart necessarily requires the inclusion of device-specific behavior.

Table 33 AE Class Behavior State Description

State	*Description*
NORMAL OPERATING	Object instance exists. Object services can be processed.
RUNNING	This is the entry sub-state to NORMAL OPERATING. Object instance is not in its ABORT state.
INITIALIZED	This is the entry sub-state to NORMAL OPERATING and RUNNING. Object instance exists and has been initialized. All attributes set to appropriate initial values (as indicated in an appropriate sensor/actuator network specific device model specification).
RECOVERED	Object instance is in an abort recovered state.
OPERATING	This represents the entire collection of sub-states within RUNNING except INITIALIZED and RECOVERED. The further delineation of this sub-state and transition into this sub-state is outside the scope of this document.
ABORT	Object instance is in an aborted state and is not running any application process; a detailed description of this state is outside the scope of this document.

Table 34 AE Class Behavior State Transition Matrix*

#	*Current State*	*Trigger*	*New State*	*Action*	*Comment*
1	RUNNING	Abort request.	ABORT	Abort. Abort response.	ABORT state is device-specific.
2	ABORT	Recover request.	RECOVERED	Recover. Recover response.	RECOVERED state is device-specific.
3	NORMAL OPERATING	Reset request.	RUNNING	Reset. Reset response.	Valid for all sub-states of NORMAL OPERATING. Move to RUNNING/ INITIALIZED.
4	NORMAL OPERATING	GetAttribute, SetAttribute, RestoreDefault.	NORMAL OPERATING	Get Attribute, Set Attribute, Restore Defaults. Appropriate Response.	Valid for all sub-states of NORMAL OPERATING.
5	NORMAL OPERATING	Invalid service request.	NORMAL OPERATING	Error response.	Valid for all substates of NORMAL OPERATING.
6	ABORT	Abort request, Operate request.	ABORT	Error response.	Invalid request in this state.
7	RUNNING	Recover request.	RUNNING	Error response.	Recover service can only be invoked while in ABORT state.
8	INITIALIZING	Operate request.	OPERATING	Device-specific.	Behavior associated with this transition may be further specified in an appropriate sensor/ actuator network specific device model specification.
9	RECOVERED	Operate request.	OPERATING	Device-specific.	Behavior associated with this transition may be further specified in an appropriate sensor/actuator network specific device model.
10	OPERATING	Operate request.	OPERATING	Operate response.	Object remains in the OPERATING state. Additional behavior associated with this transition may befurther specified in an appropriate sensor/actuator network specific device model.

* NOTE: Only generic behavior is indicated in this table.

7.4.2 *Sensor (S) Class* — The S class at the second level of the active element generalization hierarchy contains structure and behavior common to all sensing element instances in a device.

7.4.2.1 *Sensor Class Attributes* — All required and "common optional" S class attributes are listed in Table 35 and described below. For an explanation of the table format, see Section 7.3.1. Note that the S class also inherits all AE class attributes.

Table 35 S Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Value	nA16**	RO	Y	Context-Specific
ReportInhibitTimer	nA17	RW	N	Context-Specific
EnableReportRate	nA18	RW	N	Boolean
ReportRate	nA19***	RW	N	Context-Specific

* A "Definition" column is not included. For attribute definitions, see text.

** Where 'n' is the type of the object instance to which the particular attribute is associated (e.g., Sai or Sbi), see Sections 7.4.4 through 7.4.8.

*** ID's nA20 through nA63 are reserved for future S class attribute definition.

7.4.2.1.1 *Value* — The value of the sensor (associated with the object instance) that is to be conveyed. The form and content of this attribute is context-specific.

7.4.2.1.2 *ReportInhibitTimer* — An attribute which is used to identify the absolute minimal time interval (i.e., maximum rate) for reporting the attribute "Value"; expressed in seconds. A value of zero indicates that there is no minimal time interval.

7.4.2.1.3 *EnableReportRate* — An attribute which enables the reporting of the attribute "Value" based on the "ReportingRate" attribute value; a value of TRUE signifies that the reporting at the "ReportingRate" is enabled.

7.4.2.1.4 *ReportRate* — An attribute which defines the time interval at which the attribute "Value" will be reported; expressed in seconds.

7.4.2.2 *Sensor Class Services* — No S class common services are defined. Note that the S class inherits all AE class services.

7.4.2.3 *Sensor Class Behavior* — The following applies to S class behavior:

Some form of measurement or other sensory activity is made by the active sensory element. This measurement is converted into the appropriate data and becomes the value of the "Value" attribute. The method of conversion is beyond the scope of this document.

No other S class common behavior is defined. Note that the S class inherits all AE class-defined behavior.

7.4.3 *Actuator (A) Class* — The A class at the second level of the active element generalization hierarchy contains structure and behavior common to all actuator element instances in a device.

7.4.3.1 *Actuator Class Attributes* — All required and "common optional" A class attributes are listed in Table 36 and described below. For an explanation of the table format, see Section 7.3.1. Note that the A class also inherits all AE class attributes.

Table 36 A Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Setting	nA16**	RW	Y	Context-Specific
SafeState	nA17	RW	N	Context-Specific
WatchRate	nA18	RW	N	Unsigned Integer < 65,536
WatchDog	nA19***	RW	N	Boolean

* A "Definition" column is not included. For attribute definitions, see text.

** Where 'n' is the type of the object instance to which the particular attribute is associated (e.g., Aao or Ado), see Sections 7.4.4 through 7.4.8.

*** ID's nA20 through nA63 are reserved for future A class attribute definition.

7.4.3.1.1 *Setting* — Specifies the value that is to be actuated by the actuator associated with the object instance. The form and content of this attribute is context-specific.

7.4.3.1.2 *SafeState* — Specifies the state in which the actuator will be placed for ABORT and INITIALIZING states. Unless otherwise specified, this attribute is a Boolean type corresponding to Zero (or "no output") or One (or "full output").

7.4.3.1.3 *WatchRate* — Specifies the rate at which the "Setting" attribute is to be refreshed (i.e., set externally). If the "Setting" attribute is not refreshed at this rate, it reverts to its default value; expressed in (integer/100) hertz.

7.4.3.1.4 *WatchDog* — Enables or disables the capability of setting the "Setting" attribute to its default value due to a refresh rate lower than specified by the "WatchRate" attribute value; a value of TRUE signifies that the WatchRate functionality is enabled.

7.4.3.2 *Actuator Class Services* — No A class common services are defined. Note that the A class inherits all AE class services.

7.4.3.3 *Actuator Class Behavior* — The following applies to A class behavior:

Some form of actuation activity is made by the active actuation element. The value of the "Setting" attribute is converted into the appropriate signal to cause the desired actuation.

A class-defined behavior associated with the "WatchRate" and "WatchDog" attributes is defined in Section 7.4.3.1 (above). No other A class common behavior is defined. Note that the A class inherits all AE class-defined behavior.

7.4.4 *Controller (C) Class* — The C class at the second level of the active element generalization hierarchy contains structure and behavior common to all controller element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device type (see Figure 7), object instances can be created as instances of this class type.

7.4.4.1 *Controller Class Attributes* — All required and "common optional" C class attributes are listed in Table 37 and described below. For an explanation of the table format, see Section 7.3.1. Note that the C class also inherits all AE class attributes.

Table 37 C Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Setpoint	CA16	RW	Y	Context-Specific
ProcessVariable	CA17	RW	N	Context-Specific
ControlVariable	CA18	RW	N	Context-Specific
DataType	CA19	RW	N	Context-Specific
AlarmSettleTime	CA21	RW	N	Context-Specific
AlarmErrorBand	CA22	RW	N	Context-Specific
WarningSettleTime	CA24	RW	N	Context-Specific
WarningErrorBand	CA25	RW	N	Context-Specific

* A "Definition" column is not included. For attribute definitions, see text.

** ID's CA27 through CA63 are reserved for future C class attribute definition.

7.4.4.1.1 *Setpoint* — This attribute specifies the value to which the Process Variable will be controlled via the Control Variable. This is one of the inputs of a classic single-stage closed-loop controller. The setpoint attribute value is expressed in units identified by the "DataUnits" attribute.

7.4.4.1.2 *ProcessVariable* — This attribute value is the measurement of the process being controlled. This is one of the inputs of a classic single-stage closed-loop controller.

7.4.4.1.3 *ControlVariable* — This attribute value is the control signal being sent to the process controlling element. This is the output of a classic single-stage closed-loop controller.

7.4.4.1.4 *DataType* — An attribute which defines the format of the data associated with "Value" attribute (inherited) of the object instance.

7.4.4.1.5 *DataUnits* — An attribute which defines the current units associated with "Value" attribute (inherited) of the object instance.

7.4.4.1.6 *AlarmSettleTime* — An attribute which specifies a time used in the determination of a Controller object instance alarm exception condition. This value is expressed in units of seconds. The minimum allowable value for this attribute is specified by the manufacturer. An out-of-range error will result from an attempt to set this attribute to a value lower than that specified as the minimum allowable.

7.4.4.1.7 *AlarmErrorBand* — An attribute which specifies an amount by which the process variable may not equal the setpoint used in the determination of a Controller object instance alarm exception condition. This value is expressed in units identified by the "DataUnits" attribute and in a format specified by the "DataType" attribute.

7.4.4.1.8 *WarningSettleTime* — An attribute which specifies a time used in the determination of a Controller object instance warning exception condition. This value is expressed in units of seconds. The minimum allowable value for this attribute is specified by the manufacturer. An out-of-range error will result from an attempt to set this attribute to a value lower than that specified as the minimum allowable.

7.4.4.1.9 *WarningErrorBand* — An attribute which specifies an amount by which the process variable may not equal the setpoint used in the determination of a Controller object instance warning exception condition. This value is expressed in units identified by the "DataUnits" attribute and in a format specified by the "DataType" attribute.

7.4.4.2 *Controller Class Services* — No C class common services are defined. Note that the C class inherits all AE class services.

7.4.4.3 *Controller Class Behavior* — The following applies to C class behavior:

The controller element uses the value of the Process-Variable and Setpoint attributes and determines the error signal. It then adjusts the ControlVariable attribute value as necessary and appropriate to minimize this error signal. The method of, and conditions for, adjustment are beyond the scope of this document.

Note that the C class inherits all AE class-defined behavior.

7.4.5 *Sensor-Analog Input (SAI) Class* — The SAI class at the third level of the active element generalization hierarchy contains structure and behavior common to all analog input sensor element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.5.1 *Sensor-Analog Input Class Attributes* — All required and "common optional" SAI class attributes are listed in Table 38 and described below. For an explanation of the table format, see Section 7.3.1. Note that the SAI class also inherits all AE and S class attributes.

Table 38 SAI Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Offset	SaiA64	RW	N	Context-Specific
Gain	SaiA65	RW	N	Context-Specific
DataType	SaiA66	RW	N	Context-Specific
DataUnits	SaiA67	RW	N	Context-Specific
SafeState	SaiA68	RW	N	Enumerated Byte
EnableReportDelta	SaiA69	RW	N	Boolean
ReportDelta	SaiA70	RW	N	Context-Specific
EnableReportROC	SaiA71	RW	N	Boolean
ReportROC	SaiA72	RW	N	Context-Specific
AlarmTripPointHigh	SaiA73	RW	N	Context-Specific
AlarmTripPointLow	SaiA74	RW	N	Context-Specific
AlarmHystersis	SaiA75	RW	N	Context-Specific
WarningTripPointHigh	SaiA76	RW	N	Context-Specific
WarningTripPointLow	SaiA77	RW	N	Context-Specific
WarningHystersis	SaiA78**	RW	N	Context-Specific

* A "Definition" column is not included. For attribute definitions, see text.

** ID's SaiA78 through SaiA127 are reserved for future Sai class attribute definition.

7.4.5.1.1 *Offset* — An attribute which specifies the amount of offset correction being applied to derive the value of the "Value" attribute (inherited) of the object instance. It is expressed in terms of the units specified by the "DataUnits" attribute.

7.4.5.1.2 *Gain* — An attribute which is used to define the range of the "Value" attribute (inherited) of the object instance. The method of conveying this range with this attribute is beyond the scope of this document.

7.4.5.1.3 *DataType* — An attribute which defines the format of the data associated with "Value" attribute (inherited) of the object instance.

7.4.5.1.4 *DataUnits* — An attribute which defines the current units associated with "Value" attribute (inherited) of the object instance.

7.4.5.1.5 *SafeState* — An attribute which defines the Value (inherited attribute) to be reported by the object instance when it is in a non-OPERATING state. The enumeration of the SafeState attribute is given in Table 39.

Table 39 SafeState Attribute Values

Attribute Value	*DM Object State*
0	Zero
1	Full-Scale
2	LVV
3	Undefined
4 – 63	Reserved
64 – 255	Open

7.4.5.1.6 *EnableReportDelta* — An attribute which enables the reporting of the attribute "Value" based on the "ReportDelta" attribute value (see below); a value of TRUE signifies that reporting using the "ReportDelta" method is enabled.

7.4.5.1.7 *ReportDelta* — The value of this attribute indicates the amount by which the sensor value (indicated by the "Value" attribute (inherited)) must change before it is published. When the value of the EnableReportDelta attribute indicates that this method of reporting is "enabled", and when the value of the "Value" attribute changes by an amount in excess of the amount indicated by the ReportDelta attribute value, the object will publish the Value attribute.

7.4.5.1.8 *EnableReportROC* — The Enable ReportROC (rate of change) attribute enables the reporting of the attribute "Value" based on the "ReportROC" attribute value (see below); a value of TRUE signifies that reporting using the rate of change method is enabled.

7.4.5.1.9 *ReportROC* — The value of this attribute indicates the rate by which the sensor value (indicated by the "Value" attribute (inherited)) must change before it is published. The attribute value context is DataUnits/Time; the specific context of DataUnits is defined in the DataUnits attribute, while the unit of Time is context-specific. When the value of the EnableReportDelta attribute indicates that this method of reporting is "enabled", and when the value of the "Value" attribute changes at a rate in excess of the amount indicated by the ReportDelta attribute value, the object will publish the Value attribute.

7.4.5.1.10 *AlarmTripPointHigh* — An attribute which specifies the value above which the object instance "Value" attribute (inherited) value must be to generate an alarm status indication. The behavior associated with the generation of an alarm status indication (beyond that described in Section 7.4.1.3) is beyond the scope of this document.

7.4.5.1.11 *AlarmTripPointLow* — An attribute which specifies the value below which the object instance "Value" attribute (inherited) value must be to generate an alarm status indication. The behavior associated with the generation of an alarm status indication (beyond that described in Section 7.4.1.3) is beyond the scope of this document.

7.4.5.1.12 *AlarmHystersis* — An attribute which specifies the amount of hysteresis to be applied in the clearing of an alarm condition. For example: a Trip Point High value of 100 with a hysteresis value of 2 will result in an exception condition being set when the Value attribute exceeds 100 in the positive direction and cleared when the Value attribute exceeds 98 in the negative direction. Similarly, a Trip Point Low value of 100 with a hysteresis value of 2 will result in an exception condition being set when the Value attribute exceeds 100 in the negative direction and cleared when the Value attribute exceeds 102 in the positive direction. The behavior associated with the generation and clearing of exception conditions is beyond the scope of this document.

7.4.5.1.13 *WarningTripPointHigh* — An attribute which specifies the value above which the object instance "Value" attribute (inherited) value must be to generate a warning status indication. The behavior associated with the generation of a warning status indication is beyond the scope of this document.

7.4.5.1.14 *WarningTripPointLow* — An attribute which specifies the value below which the object instance "Value" attribute (inherited) value must be to generate a warning status indication. The behavior associated with the generation of a warning status indication is beyond the scope of this document.

7.4.5.1.15 *WarningHystersis* — An attribute which specifies the amount of hysteresis to be applied in the clearing of an alarm condition (see Section 7.4.5.1.12). The behavior associated with the generation and clearing of exception conditions is beyond the scope of this document.

7.4.5.2 *Sensor-Analog Input Class Services* — No SAI class common services are defined. Note that the SAI class inherits all AE and S class services.

7.4.5.3 *Sensor-Analog Input Class Behavior* — No SAI class common behavior is defined. Note that the SAI class inherits all AE and S class-defined behavior.

7.4.6 *Sensor-Binary Input (SBI) Class* — The SBI class at the third level of the active element generalization hierarchy contains structure and behavior common to all binary input sensor element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.6.1 *Sensor-Binary Input Class Attributes* — All required and "common optional" SBI class attributes are listed in Table 40 and described below. For an explanation of the table format, see Section 7.3.1. Note that the SBI class also inherits all AE and S class attributes.

Table 40 SBI Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
DebounceControl	SbiA64	RW	N	Context-specific
AlarmState	SbiA65	RW	N	Boolean
WarningState	SbiA66**	RW	N	Boolean

* A "Definition" column is not included. For attribute definitions, see text.

** ID's Sbi67 through SbiA127 are reserved for future Sbi class attribute definition.

7.4.6.1.1 *DebounceControl* — This attribute is used to control sensitivity to 'false' transitions. The method of utilization of this attribute for debounce control is application-specific.

7.4.6.1.2 *AlarmState* — The value of the Value attribute (i.e., TRUE or FALSE) that will cause an alarm condition.

7.4.6.1.3 *WarningState* — The value of the Value attribute (i.e., TRUE or FALSE) that will cause a warning condition.

7.4.6.2 *Sensor-Binary Input Class Services* — No SBI class common services are defined. Note that the SBI class inherits all AE and S class services.

7.4.6.3 *Sensor-Binary Input Class Behavior* — No SBI class common behavior is defined. Note that the SBI class inherits all AE and S class-defined behavior.

7.4.6.4 *Sensor-Enumerated Input Class Services* — No SEI class common services are defined. Note that the SEI class inherits all AE and S class services.

7.4.6.5 *Sensor-Enumerated Input Class Attributes* — All required and "common optional" SEI class attributes are listed in Table 41 and described below. For an explanation of the table format, see Section 7.3.1. Note that the SEI class also inherits all AE and S class attributes.

Table 41 SEI Class Attributes*

Attribute Name	*Attribute Identifier*	*Access Network*	*Rqmt*	*Form*
DebounceControl	SeiA64	RW	N	Context-Specific
AlarmState	SeiA65	RW	N	Enumerated
WarningState	SeiA66	RW	N	Enumerated

* A "Definition" column is not included. For attribute definitions, see text.

7.4.6.5.1 *DebounceControl* — This attribute is used to control sensitivity to 'false' transitions. The method of utilization of this attribute for debounce control is application-specific.

7.4.6.5.2 *AlarmState* — The value of the Value attribute that will cause an alarm condition.

7.4.6.5.3 *WarningState* — The value of the Value attribute that will cause a warning condition.

7.4.7 *Actuator-Analog Output (AAO) Class* — The AAO class at the third level of the active element generalization hierarchy contains structure and behavior common to all analog output actuator element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.7.1 *Actuator-Analog Output Class Attributes* — All required and "common optional" AAO class attributes are listed in Table 42 and described below. For an explanation of the table format, see Section 7.3.1. Note that the AAO class also inherits all AE and A class attributes.

Table 42 AAO Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Offset	AaoA64	RW	N	Context-specific
Gain	AaoA65	RW	N	Context-specific
DataType	AaoA66	RW	N	Context-specific
DataUnits	AaoA67**	RW	N	Context-specific

* A "Definition" column is not included. For attribute definitions, see text.

** ID's AaoA68 through AaoA127 are reserved for future Aao class attribute definition.

7.4.7.1.1 *Offset* — An attribute which specifies the amount of offset correction being applied to derive the value of the "Value" attribute (inherited) of the object instance. It is expressed in terms of the units specified by the "DataUnits" attribute.

7.4.7.1.2 *Gain* — An attribute which is used to define the range of the "Value" attribute (inherited) of the object instance. The method of conveying this range with this attribute is beyond the scope of this document.

7.4.7.1.3 *DataType* — An attribute which defines the format of the data associated with "Value" attribute (inherited) of the object instance.

7.4.7.1.4 *DataUnits* — An attribute which defines the current units associated with "Value" attribute (inherited) of the object instance.

7.4.7.2 *Actuator-Analog Output Class Services* — No AAO class common services are defined. Note that the AAO class inherits all AE and A class services.

7.4.7.3 *Actuator-Analog Output Class Behavior* — No AAO class common behavior is defined. Note that the AAO class inherits all AE and A class-defined behavior.

7.4.8 *Actuator-Binary Output (ABO) Class* — The ABO class at the third level of the active element generalization hierarchy contains structure and behavior common to all binary output actuator element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.8.1 *Actuator-Binary Output Class Attributes* — No ABO class common attributes are defined. Note that the ABO class inherits all AE and A class attributes.

7.4.8.2 *Actuator-Binary Output Class Services* — No ABO class common services are defined. Note that the ABO class inherits all AE and A class services.

7.4.8.3 *Actuator-Binary Output Class Behavior* — No ABO class common behavior is defined. Note that the ABO class inherits all AE and A class-defined behavior.

7.4.9 *Actuator-Enumerated Output (AEO) Class* — The AEO class at the third level of the active element generalization hierarchy contains structure and behavior common to all Enumerated output actuator element instances in a device. Note that, as discussed in Section 7.4, since this is the lowest level of class description in the generalization hierarchy for this device element type (see Figure 7), object instances can be created as instances of this class type.

7.4.9.1 *Actuator-Enumerated Output Class Attributes* — No AEO class common attributes are defined. Note that the AEO class inherits all AE and A class attributes.

7.4.9.2 *Actuator-Enumerated Output Class Services* — No AEO class common services are defined. Note that the AEO class inherits all AE and A class services.

7.4.9.3 *Actuator-Enumerated Output Class Behavior* — No AEO class common behavior is defined. Note that the AEO class inherits all AE and A class-defined behavior.

7.4.10 *Sensor-Binary Input-Threshold (SBITH) Class* — The SBITH class at the fourth level of the active element generalization hierarchy contains structure and behavior common to all binary input threshold sensor element instances in a device.

7.4.10.1 *Sensor-Binary Input Threshold Class Attributes* — All required and "common optional" SBITH class attributes are listed in Table 43 and described below. For an explanation of the table format see Section 7.3.1. Note that the SBITH class also inherits all AE, S, an SBI class attributes.

Table 43 SBITH Class Attributes*

Attribute Name	*Attribute Identifier*	*Access (Network)*	*Rqmt*	*Form*
Reading Valid	SbithA64	R	N	Enumerated, Byte
State	SbithA65	R	N	Enumerated, Byte
Status	SbithA66	R	N	Enumerated, Byte

* A "Definition" column is not included. For attribute definitions, see text.

** ID's Sbith67 through Sbith127 are reserved for future Sbith class attribute definition.

7.4.10.1.1 *Reading Valid* — This attribute is used to specify the validity of the "*Value*" attribute of the SBITH object. The method of utilization of this attribute is application specific.

7.4.10.1.2 *State* — This attribute is used to record the current state of the SBITH object. The method of utilization of this attribute is application specific.

7.4.10.1.3 *Status* — This attribute is used to record the current active reporting status of the SBITH object. The method of utilization of this attribute is application specific.

7.4.10.2 *Sensor-Binary Input-Threshold Class Services* — No SBITH class common services are defined. Note that the SBITH class inherits all AE, S, and SBI class services.

7.4.10.3 *Sensor-Binary Input-Threshold Class Behavior* — No SBITH class common behavior is defined. Note that the SBITH class inherits all AE, S, and SBI class behavior.

APPENDIX 1
DATA UNIT ENUMERATION

NOTE: This appendix was approved as an official part of SEMI E54.1 by full letter ballot procedure.

The Data Units data type term is defined in Section 5.2. The following is a partial enumeration of that data type; further enumeration may be provided in an application-specific context.

Table A1-1 General

Data Units	*Description*
0	Counts
1	Counts per Second
2	Counts per Millisecond
3	Counts per Microsecond
4	Counts per Minute
5	Counts per Hour
6	Counts per Day
7	Percent
8–191	Reserved
192–255	Open

Group 1 — Time and Frequency (256–511)

Table A1-2 Time and Frequency

Data Units	*Description*
256	Seconds
257	Milliseconds
258	Microseconds
259	Minutes
260	Hours
261	Days
262	Hertz
263	KiloHertz
264	MegaHertz
265	GigaHertz
266–447	Reserved
448–511	Open

Group 2 — Temperature (512–767)

Table A1-3 Temperature

Data Units	*Description*
512	Centigrade/Celsius
513	Fahrenheit
514	Kelvin
515–703	Reserved
704–767	Open

Group 3 — Pressure (768–1023)

Table A1-4 Pressure

Data Units	*Description*
768	PSI
769	Torr
770	MilliTorr
771	mm Hg (0°C)
772	inches Hg (0°C)
773	cm H_2O (25°C)
774	inches H_2O (25°C)
775	Bar
776	MilliBar
777	Pascal
778	KiloPascal
779	Atmosphere
780–959	Reserved
960–1023	Open

Group 4 — Flow (1024–1279)

Table A1-5 Flow

Data Units	*Description*
1024	SCCM
1025	SLM
1026	CFM
1027	Pa-m^3/s
1028–1215	Reserved
1216–1279	Open

Group 5 — Electrical (1280–1535)

Table A1-6 Electrical

Data Units	*Description*
1280	Volts
1281	MilliVolts
1282	MicroVolts
1283	NanoVolts
1284	PicoVolts
1285	FemtoVolts
1286	KiloVolts
1287	MegaVolts
1288	GigaVolts
1290	Amps
1291	MilliAmps

1292	MicroAmps
1293	NanoAmps
1294	PicoAmps
1295	FemtoAmps
1296	KiloAmps
1297	MegaAmps
1298	GigaAmps
1300	Ohms
1301	MilliOhms
1302	MicroOhms
1303	NanoOhms
1304	PicoOhms
1305	FemtoOhms
1306	KiloOhms
1307	MegaOhms
1308	GigaOhms
1310	Farads
1311	MilliFarads
1312	MicroFarads
1313	NanoFarads
1314	PicoFarads
1315	FemtoFarads
1320	Henries
1321	MilliHenries
1322	MicroHenries
1323	NanoHenries
1324	PicoHenries
1325	FemtoHenries
1326–1471	Reserved
1472–1535	Open

APPENDIX 2
ERROR RESPONSES

NOTE: This appendix was approved as an official part of SEMI E54.1 by full letter ballot procedure.

The following is a listing of common service error responses. The details of presentation of these responses over the network (such as response codes) are network-specific and, thus, are not part of this document. Also, note that there may be additional service error responses delineated in the appropriate SDM specification or manufacturer specification.

A2-1 Error Responses

A2-1.1 *Resource Unavailable* — Resources needed for the object to perform the requested service are unavailable.

A2-1.2 *Service Not Supported* — The requested service is not implemented or is not defined for this Object Class/Instance.

A2-1.3 *Invalid Attribute Value* — Invalid attribute data detected.

A2-1.4 *Already in Requested Mode/State* — The object is already in the mode/state being requested by the service.

A2-1.5 *Object State Conflict* — The object cannot perform the requested service in its current mode/state.

A2-1.6 *Attribute Not Settable* — A request to modify a non-modifiable attribute was received.

A2-1.7 *Device State Conflict* — The device's current mode/state prohibits the execution of the requested service.

A2-1.8 *Service Parameter Data Not Complete* — Not enough data supplied with the service request.

A2-1.9 *Attribute Not Supported* — The attribute specified in the request is not supported.

A2-1.10 *Too Much Data for Service Parameters* — The service supplied more data than was expected.

A2-1.11 *Object Does Not Exist* — The object specified does not exist in the device.

A2-1.12 *Vendor-Specific Error* — Vendor-specific error.

A2-1.13 *Invalid Service Parameter* — A parameter associated with the request was invalid.

SEMI E54.2-0698 (Reapproved 0704)
GUIDE FOR WRITING SENSOR/ACTUATOR NETWORK (SAN) STANDARD BALLOTS

This guide was technically reapproved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 14, 2004. Initially available at www.semi.org May 2004; to be published July 2004. Originally published June 1998.

1 Purpose

1.1 This guide recommends the method, form, and content for adding specific device models and associated Sensor Actuator Network Communications Standard (SANCS) extensions to the SEMI Sensor Actuator Network (SAN) standard. This guide facilitates consistency of these ballots, leading to better consistency and clarity of the resulting specifications and to easier development of SAN standard-conformant devices that achieve a high level of interoperability.

2 Scope

2.1 This guide provides written templates for ballots that will add either a specific device model (SDM), an SANCS ancillary standard, or SANCS extensions to support an SDM to the SEMI SAN standards. The guide provides directions for using the templates.

2.2 Physical devices will not be compliant with this guide; they will be compliant to standards which are spawned from the templates specified herein.

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

3 Referenced Standards

3.1 *SEMI Standards*

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

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

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services

SEMI E54 — Sensor/Actuator Network Standard

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

SEMI E54.3 — Standard for Sensor/Actuator Network Specific Device Model for Mass Flow Device

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

4 Terminology

4.1 *Abbreviations and Acronyms*

4.1.1 *A* — Actuator (a CDM class definition)

4.1.2 *AE* — Active Element (a CDM class definition)

4.1.3 *C* — Controller (a CDM class definition)

4.1.4 *CDM* — (SEMI) Common Device Model

4.1.5 *DM* — Device Manager (a CDM class definition)

4.1.6 *ISO-OSI* — International Organization for Standardization - Open Systems Interconnect (model)

4.1.7 *NCS* — Network Communications Standard

4.1.8 *OMT* — Object Modeling Technique

4.1.9 *S* — Sensor (A CDM object)

4.1.10 *SAC* — Sensor/Actuator/Controller (a CDM class definition)

4.1.11 *SAN* — Sensor/Actuator Network (or Sensor Bus)

4.1.12 *SANCS* — Sensor/Actuator Network Communications Standard, frequently abbreviated to NCS

4.1.13 *SDM* — (SEMI) Specific Device Model

4.2 *Definitions*

4.2.1 *connection oriented* — a situation between communicating objects wherein they are connected in a mode analogous to a two-way phone conversation.

4.2.2 *specific device model (SDM)* — a model used to specify each type of device, such as a Mass Flow Controller or Thermocouple.

5 Background

5.1 The SEMI sensor bus (SAN) communications model consists of three components: a CDM, an SDM, and an SANCS (NCS). The following subsections provide a brief description of the specifications which define these components:

5.1.1 *Network Communications Standard Specification* — An NCS specification is required for each network technology. Each one consists of two major parts:

5.1.1.1 *Part 1, Enabling Protocol and CDM Object Presentation* — This part of an SANCS specification should achieve the following goals:

- provide an ISO-OSI layered specification of the protocol, specifying the protocol requirements at each layer,
- specify the object oriented communication environment, including object representation and addressing, attributes and services addressing, etc.,
- specify how CDM objects are represented and addressed to/from the network, and
- specify how AE class and subclass objects, defined in the CDM and the SDMs, are represented and addressed to/from the network.

5.1.1.2 *Part 2, Enabling Specific Device Models* — This part provides a mapping of SDM objects for the SANCS specification. There should be a section for each SDM which identifies specific objects that must be supported to enable the SDM and how the SDM structure fits into the protocol application layer foundation.

5.1.2 *Common Device Model* — Each device's functions on a network should comply with the specifications of the Common Device Model, SEMI E54.1. Three types or classes of objects may be combined to create models of real devices. The types are Active Element (AE) and its subclasses, Sensor/Actuator Controller (SAC) and its embedded objects, and the Device Manager (DM).

NOTE 1: The CDM specifies the relationship between these objects. It specifies some of the structure and behavior for the DM and SAC objects. The subclasses of the AE object provide capabilities which are useful to most devices.

5.2 *Specific Device Model*

5.2.1 Specific Device Models are defined in the SEMI SAN SDM (SEMI E54.3). Each SDM is added as a separate new section (similar to the way a new section is added to SEMI E5 when a new stream is added).

5.2.2 The Specific Device Models (SDM's) should be a specialization of the CDM and may extend the attributes, services, and behavior of any of the Sensor, Actuator, or Controller (subclasses of the AE class) objects of which they are made. SDM's may add to the attributes, services, and behavior of the SAC object.

6 Ballot Preparation

6.1 *The Cover Letter* — All ballots are submitted with a cover letter. The cover letter sets the context for the ballot. The Ballot Cover Letter Template provides guidance to prepare the cover letter.

6.2 *Specific Device Model Ballots* — A ballot to add an ancillary specification for a new SDM should be prepared as defined in the template in the section titled Template for SDM Ballot Proposals.

6.2.1 SEMI E39 is the basis for the tables and figures called out in the template. Refer to SEMI E39 for clarifications on using and defining object models. General instructions for using the template are contained in the subsections of this section. Specific instructions are included within the templates.

6.2.2 Device application models should be presented using Rumbaugh's Object Modeling Technology notation for information models. (See SEMI E39 for details on using object notation for defining standards.)

6.2.3 The behavior of objects should be defined using Harel State charts, state definition tables, and state transition tables. A discussion of the notation may be found in Appendix A.5 of SEMI E30. An alternative table, Object Behavior State Transition Matrix, may be used in place of the more traditional (State) Transition Table. Cells in the Transition Matrix indicate the transition action which is taken for an event for any of the states in which it can be detected. This can be particularly useful for describing the variation in device response depending on a sub-state when the parent state detects an event.

6.2.4 The set of tables which define the device's interfaces are the only testable parts of the specification for the application models. The interface for each device model should be defined in a series of tables, as follows:

6.2.4.1 *Attribute Tables* — For each object defined as an element of a device, a table which defines its network visible attributes should be specified. Each attribute is named and tagged. Its network access type (read only or read/write) and storage class (volatile, non-volatile, or constant) are also specified. The table should also indicate which attributes have standardized values and which are device supplier reserved. The definition or description of an attribute can either be included in the table or else should immediately follow the table. The "Required" column indicates if the attribute must be supported for a device to be SDM-compliant. Use the letter "Y" to indicate yes and the letter "N" to indicate no. The form field is used to indicate the data type of the attribute, such as integer,

floating point, array of characters, etc. Actual format of these data types is network-dependent.

6.2.4.2 *Optional Service Parameter Table* — If this table is included in the specification of an SDM, the parameters, which are used by the device' s services, are defined in it, including the data type and length.

6.2.4.3 *Service Definition Table* — Each service and its parameters are defined in a table. If the optional parameter table is not used in the SDM specification, then the form field should be included in the service definition table.

6.3 *SANCS Ballots* — A ballot to add an ancillary specification for a new SANCS should be prepared as defined by the template in the section titled Template for Part 1 of an SANCS Ancillary Standard.

6.3.1 When a new SDM is added to the SAN standard, an update to Part 2 of the ancillary SANCS specifications is needed. A ballot should be prepared as defined in the template in the section titled Template for an SANCS ballot to support an SDM.

6.3.2 The nature of the Template for an SANCS ballot to support an SDM is that it adds SDM information to only one SANCS specification at a time. Additions and changes to support each new SDM are balloted separately for each SANCS specification.

6.3.3 *Coordination with Industry User Groups* — Most of the network technologies are supported by an industry user group or association. These groups are responsible for the assignment (i.e., mapping) of identifiers to the variables or attributes, parameters, and messages/services that will be managed by their protocol. When the SEMI SAN standards require new identifiers, they should be requested from the appropriate industry group.

6.3.3.1 *Technical Notes* — SEMI will make technical notes available, for each protocol supported by the SEMI SAN standards, that list the protocol' s identifiers.

6.4 *Template Conventions*

6.4.1 Instructions or text that is meant to be replaced with ballot-specific information are included within the templates in italics. Non-italicized text and tables are intended to be included in the ballot as depicted in the template.

6.4.2 Section, figure, and table numbers specified using letters are meant to indicate structure and format. Specific numbers should be generated by the ballot author. Ballots which require adding to or modifying existing SAN standards will be edited by SEMI for consistency with the existing standard where necessary.

7 Ballot Cover Letter Template

7.1 Each ballot is accompanied by a cover letter which includes the following sections:

7.1.1 *Proposal* — Provide a brief description, typically one or two sentences, of the content of the ballot proposal.

7.1.2 *Background* — In 1–3 paragraphs, give the readers the background they need to understand the content and value of the proposed standard.

7.1.3 *Impact* — Use one or two paragraphs to describe how this standard affects the SEMI community.

7.1.4 *Ballot Description* — Describe the form of the ballot proposal. Use one or two paragraphs.

7.1.4.1 Note that SDM ballots are additions to the SEMI SAN SDM Specification (SEMI E54.3). This is a parent document which already has sections for overall purpose, scope, terminology, conventions, etc. for SDMs. Each individual SDM ballot should include only device-specific purpose, scope, and terminology. Items that are used in multiple SDMs should be balloted as updates to the SEMI E54.3 parent document.

8 Template for SDM Ballot Proposals

SEMI Document Number

Title

1 Purpose

Describe the purpose of the document. This is typically to provide a network-independent application model for the specific device - NamedDevice.

2 Scope

Describe the scope of the document.

3 Referenced Documents

List documents referenced from within this specification.

4 Terminology

Define terms used within this document. This is only for terms that are used within this document and not defined elsewhere (in the SEMI E54.3 parent document or other referenced documents).

5 NamedDevice High Level Structure

5.1 *General Description* — Describe what this device does, how it is used, etc.

5.1.1 *NamedDevice Description* — Provide a description of NamedDevice, including information model(s) in figure(s) that use a Rumbaugh OMT (see SEMI E39 Appendix A1-2) diagram.

NOTE 2: The device may actually consist of multiple devices (e.g., Mass Flow Device includes a Controller device and a Meter device), in which case multiple sections should be created - one per device.

5.1.x *General Requirements*

5.1.x.1 *Device Objects* — All objects are defined in terms of their object name and instance identifier. Identifiers for all objects described in this document are summarized in Table m.

Table m NamedDevice Device Objects

Referenced Document Section	*Object Name*	*Object Identifier*	*Minimum Instances*	*Maximum Instances*

All parameters used by the device' s services can be summarized using the table (Table n). This is optional, as the parameters are further defined in association with specific objects (see 5.x.2).

Table n Parameter Definition Table

Parameter Name	*Data Type*	*Tag*	*Description*

For each Object in Table m, a section (5.x) is created as below, beginning with 5.2.

5.x *Name1 Object*

5.x.1 *Name1 Object Attributes* — Provide the object attribute information using the table (Table o) below.

Table o Object Name1 Attributes

Attribute Name	*Definition**	*Attribute Identifier*	*Network Access*	*Form*	*Storage Class*	*Required*

* The definition column can be omitted if the definitions immediately follow the table. The definitions can be provided as subsections.

Where appropriate, provide initial and default values for object attributes. This can be done as a table.

5.x.2 *Name1 Object Services* — Describe the services provided by this object using the following table (Table p) to list the services:

Table p Object Name1 Services

Service	*Service Identifier*	*Type*	*Description*

Provide a detailed description of each service, including a definition of the parameters for the service. The parameters can be defined using the following table (Table q) format:

Table q Service1 Service Parameter Definitions

Parameter	*Request/Indication*	*Response/Confirmation*	*Data Type**	*Description*

* This column can be removed if there is an overall parameter definition table (Table n) included in the specification.

5.x.3 *Name1 Object Behavior* — Describe the object states and sub-states as needed with a Harel State Diagram(s):

Figure

Harel State Diagram

Provide a state definition table as given in Table r. More than one state table could be added if a device is sufficiently complex. Put in any text needed to explain object states.

Table r Name1 Object Behavior State Descriptions

State Name	*Description*

Describe the state transitions using a table as given in Tables s or t. More than one table could be needed.

Table s Name1 Object Behavior State Transition Table

Event or Transition #	*Current State*	*Trigger*	*New State*	*Action(s)*	*Comments*

Table t Name1 Object Behavior State Transition Matrix

Event Number	*State 1*	*State 2*	*State 3*	
			Sub A	*Sub B*
1				
2				
3				
4				

9 Template for Part 1 of an SANCS Ancillary Standard

9.1 The body of the ballot should then contain the following sections:

1 Purpose

This section contains wording describing the purpose with customization for the specific protocol. A “Background and Motivation” subsection could be included to generally describe the protocol.

2 Scope

This section contains wording describing the scope with customization for the specific protocol.

3 Limitations

This section contains wording describing the limitations customized for the specific protocol. Any limitations of CDM or SDMs support, due to the protocol, should be identified.

4 Referenced Documents

This section contains wording describing the referenced documents with additional references to protocol specification documentation and any support documentation as required for complete specification of the protocol (two subsections). A clearly defined mechanism is specified for obtaining any documentation that is not publicly available or available from SEMI.

5 Terminology

The various protocols and the CDM invariably use different terminology. A mapping is made between the CDM terminology and the terminology used in discussing the protocol. For example, a CDM Service implementation may be defined as an Action in the NCS. This terminology mapping is explicitly covered completely in the "Terminology" section. A mapping table is desirable that contains a column listing of the CDM terms in the same order as defined in the CDM document, mapped to terminology used in the NCS document. This NCS terminology and any additional terminology is defined in a subsection (e.g., Section 5.2).

6 Communication Protocol High Level Structure

A basic description of the protocol is provided in Section 6. The main purpose of Section 6 is to give the reader a general understanding of what will be specifically presented in the remainder of the subsections in Section 6.

The rest of the subsections in Section 6 should be devoted to describing the communications standard in terms of the OSI seven-layer model (one subsection for each layer) and any network management services.

Each of the OSI seven layers should be addressed and specified either directly or through reference; for each:

- If an OSI layer is not supported/defined, this is stated explicitly.
- Choices among protocol options are made, specified, and explained where appropriate. A complete summary of protocol choices specified and protocol options allowed should be summarized in a subsection of this section of the specification.

The Physical layer is specified either directly or through reference (signaling, bus arbitration, baud rate, transceivers, cabling, connectors, etc.).

The Data link layer is specified either directly or through reference (packet/frame specification and node addressing).

The Network layer is optional.

The Transport layer elements are specified either directly or through reference; message segmentation and reassembling should be supported as no message length limits are indicated in CDM or SDM' s. This support should be provided as a transport layer service. Note that this functionality may alternatively be provided at the application layer by protocols unable to support it at the transport layer; if this is the case, it is explicitly stated in the description of transport layer support. Connection support would be specified here as appropriate.

The Session layer is optional.

The Presentation layer is optional.

The Application layer is specified. The document defines (explicitly or through reference) how objects are structured, identified, and addressed. Thus, it should also define how object attributes are addressed and how object request and notify services are addressed and communicated. This section also defines the application object to object communication mechanism which should include or reference a communication state model. This communication model should clearly indicate whether or not the protocol is connection oriented.

Network Management Services may be defined in a separate subsection.

7 Required Object Types

This section identifies specific objects that must be supported to enable the CDM. The document should identify how the CDM structure (Objects and relations) fits into the protocol application layer foundation. Any identifiable protocol object hierarchy should be introduced here. Specifically, the following are defined:

- The implementation of all CDM objects (DM, SAC, AE, and all subclasses) defined to the extent that these objects are defined in CDM documentation.
- Specific addressing/referencing for CDM objects (attributes, services, etc.) defined as necessary so as to specify access to CDM objects as required by the CDM standard specification. Attributes and services are further specified as necessary so that the protocol required to access specific attributes and services is fully defined. For example, a CDM service may be defined as a "new" SANCS service or mapped to an existing SANCS service (if an appropriate candidate has already been defined in the protocol specification).
- Any limitations among CDM object options are identified.
- Any additional capabilities required of, or optional for, CDM objects (e.g., attributes, services, and/or behavior) are identified.
- Any additional required objects (e.g., network management object) identified and their interaction with the CDM objects (e.g., connection management object) should be specified.

8 Protocol Compliance

This section should specify a method or reference for testing protocol compliance. It should contain in subsection 8.1 a "protocol specification sheet" that lists all protocol options that have been specified in the document or must be resolved by the user in order to achieve interoperability.

10 Template for an SANCS Ballot to Support an SDM

Section 1 Table of Contents

Request that the table of contents for the SANCS be updated to contain a pointer to the paragraphs which specify the SDM mappings which are being added by the ballot.

Section 2 <SDM> Object Mapping

Each of the subsections within this section should identify specific objects that must be supported to enable the SDM and should identify how the SDM structure (objects and relations) fits into the protocol application layer foundation.

Specifically, the following issues are addressed:

Terminology

- A mapping is made between the SDM terminology and the terminology used in discussing the protocol. This is explicitly covered in a "Terminology" section. A mapping table is desirable that contains a column listing of the SDM terms in the same order as defined in the CDM document, mapped to terminology used by the network technology.

Required Object Types

- Address implementation of all SDM objects to the extent that they are defined in SDM documentation.
- Identify and map the specific addressing/referencing for SDM object attributes, services, and parameters to the protocol. Note that, for example, this may be achieved by fully specifying a SDM service as a "new" NCS service or by mapping the service to an existing NCS service.

Protocol Compliance

- Provide methods or references (in addition to those identified in the section titled Template for Part 1 of an SANCS Ancillary Standard, Section 8) for testing protocol compliance of the SDM.

SEMI E54.3-0698 (Reapproved 0704)
SPECIFICATION FOR SENSOR/ACTUATOR NETWORK SPECIFIC DEVICE MODEL FOR MASS FLOW DEVICE

This specification was technically reapproved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 14, 2004. Initially available at www.semi.org May 2004; to be published July 2004. Originally published June 1998.

1 Purpose

1.1 This specification is part of a suite of standards which specify the implementation of SEMI standards for the Sensor/Actuator Network. The specific purpose of this specification is to describe a network-independent application model comprised of device objects which are common to all Mass Flow Devices on a semiconductor equipment Sensor/Actuator communications network.

2 Scope

2.1 This specification specifically addresses the minimum attributes, services, and behavior a Mass Flow Controller (MFC) and Mass Flow Meter (MFM) device must support to be interoperable on the Sensor/Actuator Network.

2.2 This specification is intended to ensure a high-degree of device interoperability on the Sensor/Actuator Network, while still allowing flexibility for product differentiation and technology evolution.

2.3 The model specified in this specification is used in conjunction with the Sensor/Actuator Network Common Device Model (CDM) to completely describe the MFC or MFM as it appears from the network interface.

2.4 This specification, together with the Sensor/Actuator Network Standard, the Sensor/Actuator Network Common Device Model, and a Sensor/Actuator Network Communication Specification, form a complete interoperability specification for the MFC and MFM.

2.5 To comply with this specification, a device must implement and support, at a minimum, the required attributes, services, and behavior identified in these documents. Support for **optional** attributes, services, and behavior are not required to be compliant to this specification. Optional attributes, services, and behavior are specified in these documents to promote further device interoperability as features evolve and are adopted by more manufacturers. If optional attributes, services, and behavior are implemented for this device, they must be implemented as identified in this document.

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

3 Limitations

3.1 This specification is a companion to a suite of specifications which together make up the Sensor/Actuator Network Communication standard. Therefore, using portions of this specification that relate to network communications necessarily requires an understanding of the associated network specification.

3.2 As this document is a specification for the Mass Flow Device Model, it does not contain any definition of objects, attributes, services, or behavioral descriptions that are already defined in the Sensor/Actuator Network Common Device Model (CDM). Additional attributes, attribute assignments, services, and/or service parameters that are Mass Flow Device-specific and/or implementation-specific are contained in this specification.

3.3 While this specification is sufficient to completely describe the MFC or MFM as it appears from the network, it does not fully describe behavior of the MFC or MFM which is not visible from the network. This allows flexibility in implementation techniques and product differentiation between manufacturers. Manufacturer-specific objects may be defined by the manufacturer, but are, by definition, outside the scope of this standard.

3.4 This specification is compatible, but not compliant, with SEMI E39. This means that although this specification does not require compliance with SEMI E39, it is extensible such that implementations may be developed that are fully compliant with both standards. Note that the concepts and terminology of this specification are compatible with those of SEMI E39. However, SEMI E39 has specific requirements that are intended for higher level applications and, thus, are not applied to the Mass Flow Device Model.

3.5 Operation over the entire range specified for an attribute within a specific object instance is not a requisite for compliance with this specification.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E12 — Standard for Standard Pressure, Temperature, Density, and Flow Units Used in Mass Flow Meters and Mass Flow Controllers

SEMI E18 — Guideline for Temperature Specifications of the Mass Flow Controller

SEMI E39 — Object Services Standard: Concepts, Behavior, and Services

SEMI E52 — Practice for Referencing Gases and Gas Mixtures Used in Digital Mass Flow Controllers

SEMI E54.1 — Standard for Sensor/Actuator Network Common Device Model

4.2 *IEEE Document*[1]

IEEE 754 — Floating Point Definition

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

5 Terminology

5.1 *Terminology Defined in Standard for Sensor/Actuator Network Common Device Model (SEMI E54.1)*

5.1.1 Attribute

5.1.2 Behavior

5.1.3 Byte

5.1.4 Character

5.1.5 Device

5.1.6 Device Manager (DM) Object

5.1.7 Device Model

5.1.8 Instance

5.1.9 Nibble

5.1.10 Object

5.1.11 S, A, and C Objects

5.1.12 Sensor Actuator Controller (SAC) Object

5.1.13 Service

5.1.14 State Diagram

5.2 *Definitions*

5.2.1 *boolean (BOOL)* — a binary bit representing 0 and 1 corresponding to FALSE and TRUE or DISABLE and ENABLE respectively.

5.2.2 *common device model (CDM)* — refers to Sensor/Actuator Network Common Device Model (SEMI E54.1).

5.2.3 *data type* — an unsigned short integer formatted as an enumerated byte to specify attribute data format. The intended use of this attribute type is in cases where an attribute, or set of attributes, may be defined, allowing for more than one level of support (e.g., INT or REAL). The following values are defined:

0 = INT

1 = REAL

2 = USINT

3 = SINT

4 = DINT

5 = LINT

6 = UINT

7 = UDINT

8 = ULINT

9 = LREAL

10–99 = reserved for CDM

100–199 = reserved for SDMs

200–255 = manufacturer-specified

5.2.4 *data units* — an unsigned integer formatted as an enumerated byte to specify attribute data units. The intended use of this attribute type is in cases where an attribute, or set of attributes, may be defined, allowing for more than one unit's context. The values are defined in an appendix of this document.

5.2.5 *date* — a data structure of four bytes used to represent a calendar date. Table 1 defines the format of the date data type.

Table 1 Date Format

Data #	*Description*	*Range*
0–1	Year	Unsigned Integer
2	Month	Unsigned Short Integer(range of 1 to 12)
3	Day	Unsigned Short

1 Institute of Electrical and Electronics Engineers, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, New Jersey 08855-1331, USA. Telephone: 732.981.0060; Fax: 732.981.1721, Website: www.ieee.org

		Integer(range of 1 to 31)

5.2.6 *double integer (DINT)* — an integer, four bytes long, in the range -2^{31} to $2^{31}-1$.

5.2.7 *enumerated byte* — a byte with assigned meaning to the values 0 through 255. May take on one of a limited set of possible values.

5.2.8 *full scale range* — the defined 100% value of an attribute in its assigned units. This value is not necessarily the maximum value for the attribute. As an example, the *indicated flow* attribute value may attain 120% of the full scale range.

5.2.9 *gas calibration* — a reference to a set of parameters or methods which are used to calibrate or correct the device for a particular gas type, range, and units.

5.2.10 *gas standard number* — a number that references a gas type. The number and its referenced gas type are defined in SEMI E52.

5.2.11 *gas standard symbol* — a text symbol that references a gas type. The symbol and its referenced gas type are defined in SEMI E52.

5.2.12 *last valid value (LVV)* — the most recent value successfully assigned to an attribute.

5.2.13 *long integer (LINT)* — an integer, eight bytes long, in the range -2^{63} to $2^{63}-1$.

5.2.14 *long real (LREAL)* — a double floating point number, eight bytes long, as defined in IEEE 754.

5.2.15 *manufacturer* — in the context of this document, this refers to the manufacturer of the device.

5.2.16 *mass flow controller (MFC)* — a self-contained device, consisting of a mass flow transducer, control valve, and control and signal-processing electronics, commonly used in the semiconductor industry to measure and regulate the mass flow of gas.

5.2.17 *mass flow device (MFD)* — a device which is either a mass flow controller or mass flow meter.

5.2.18 *mass flow meter (MFM)* — a self-contained device, consisting of a mass flow transducer and signal-processing electronics, commonly used in the semiconductor industry to measure the mass flow of gas.

5.2.19 *null character* — a byte with a value of zero.

5.2.20 *programmed gas calibration* — a reference to a particular gas type, range, and units for which the device is currently calibrated.

5.2.21 *real (REAL)* — a floating point number, four bytes long, as defined by IEEE 754.

5.2.22 *short integer (SINT)* — an integer, one byte long, in the range -128 to 127.

5.2.23 *signed integer (INT)* — an integer, two bytes long, in the range -32768 to 32767.

5.2.24 *text string* — a string of one-byte characters. See Section 5.1 for a definition of a character.

5.2.25 *unsigned integer (UINT)* — an integer, two bytes long, in the range 0 to 65535.

5.2.26 *unsigned short integer (USINT)* — an integer, one byte long, in the range 0 to 255.

6 Requirements

6.1 In order to implement this standard in a Mass Flow Device, it is necessary to also implement SEMI E54.1 and one of the Sensor/Actuator Network Communication standards. See Section 2 for more information on a complete interoperability standard.

7 Conventions

7.1 This document embraces the Harel State Chart notation, the transition table definition format, the object attribute representation formats, service message definition formats, and behavior definition formats as specified in SEMI E54.1.

7.2 Figure 1 describes the convention for object representation used throughout this specification.

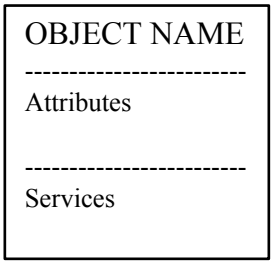

Figure 1
Object Representation

8 Device High Level Structure

8.1 *General Description* — The high level object view of a Mass Flow Meter (MFM) device and a Mass Flow Controller (MFC) Device is shown in Figure 2.

8.1.1 Note that the "MFM" and the "MFC" objects are depicted in Figure 2 only for the purposes of illustrating a high level view of the device and its component objects. In the context of this document, these objects are not addressable, do not have addressable attributes, do not have accessible services, nor do they exhibit any

defined behavior. These objects are only included in the figures to aid in the visualization of the device.

8.1.2 This document defines in detail all of the component objects unique to the MFC and the MFM devices. References, rather than definitions, are included for the DM, the SAC, and other objects defined in SEMI E54.1.

8.1.3 Many of the objects defined in this document inherit properties from other objects. The properties inherited include attribute, service, and behavior definitions. These other objects are specified here or in SEMI E54.1.

8.1.4 This document provides for future extensions, as well as manufacturer-specific enhancements, by reserving object attribute identifiers and object service identifiers. Specifically, all object definitions in this document specify or reserve the first 64 attribute identifiers (A1 through A64) and the first 64 service identifiers (S1 through S64), allowing manufacturers to specify identifiers beyond these ranges. Additionally, byte-enumerated attributes are specified or reserved from 0 to 63, allowing manufacturers to specify enumerations beyond this range (64 to 255).

8.1.5 *Mass Flow Meter (MFM) Device Description* — A Mass Flow Meter device profile is composed of the component objects and object relationships shown in Figure 2.

8.1.6 *Mass Flow Controller (MFC) Device Description* — A Mass Flow Controller device profile is composed of (1) all of the component objects and object relationships of the Mass Flow Meter Device in addition to (2) the component objects and object relationships as shown in Figure 2.

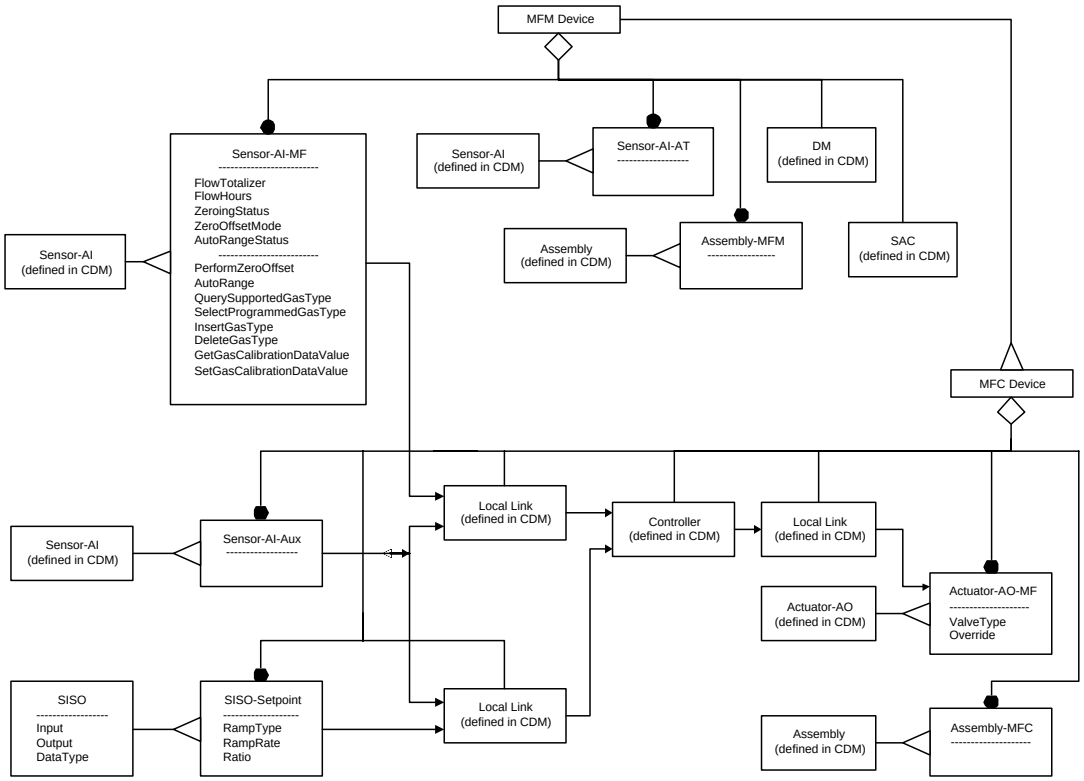

Figure 2
Mass Flow Meter Device and Mass Flow Controller Device High Level Structure

8.1.7 *General Requirements*

8.1.7.1 *Device Objects* — All objects are defined in terms of their object name and instance identifier. Identifiers for all objects described in this document are summarized in Table 2.

Table 2 Mass Flow Device Objects

Referenced Document Section	*Object Name*	*Object Identifier*	*MFC Minimum Instances*	*MFM Minimum Instances*	*Maximum Instances*
8.2	Device Manager (DM)	MFD1	1	1	1
8.3	Sensor Actuator Controller (SAC)	MFD2	1	1	1
8.4	Sensor-AI-MF	MFD3	1	1	Manufacturer-Specified
8.5	Sensor-AI-AT	MFD4	0	0	Manufacturer-Specified
8.6	Assembly-MFM	MFD5	0	0	1
8.7	Sensor-AI-Aux	MFD6	0	0	Manufacturer-Specified
8.8	Actuator-AO-MF	MFD7	1	0	Manufacturer-Specified
8.9	Controller	MFD8	1	0	1
8.10	Local Link	MFD9	2	0	Manufacturer-Specified
8.11	SISO	MFD10	0	0	Manufacturer-Specified
8.12	SISO-Setpoint	MFD11	0	0	Manufacturer-Specified
8.13	Assembly-MFC	MFD12	0	0	Manufacturer-Specified
—	Reserved	MFD13–MFD64	–	–	—
—	Manufacturer-Specified	> MFD64	–	–	—

8.1.7.2 *Object Services* — Not all object services listed in this document can necessarily be requested over the network. They are included in this document because their behavior may generate network activity.

8.1.7.3 *Object Behavior* — For all service requests received over the network that are unsupported by the object, or contain a parameter value which is beyond the supported range, or which is otherwise invalid, a network-specific service error response is generated as specified in SEMI E54.1.

8.2 *Device Manager Object (DM)*

8.2.1 The Device Manager object instance is the device component responsible for managing and consolidating the device operation as specified in SEMI E54.1. The following sections specify the components of the DM object that are not specified in the Common Device Model.

8.2.2 *Device Manager Object Attributes* — Required and optional DM object instance attributes are listed in Table 3.

Table 3 DM Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Device Type	A1	R	Yes	Refer to CDM (SEMI E54.1).
Exception Detail Alarm	A13	R	No	Refer to CDM (SEMI E54.1).
Exception Detail Warning	A14	R	No	Refer to CDM (SEMI E54.1).
Reserved	A33–A64	—	—	Reserved for SDM future expansion.
Manufacturer-Specified	> A64	—	—	Manufacturer-Specific attributes

8.2.2.1 *Device Type* — An attribute which uniquely identifies the type of the device on the network. The device type attribute is assigned as follows:

Mass Flow Controller Device = "MFC"

Mass Flow Meter Device = "MFM"

8.2.2.2 *Exception Detail Alarm (Optional)* — An attribute which identifies the detailed alarm status of the device. Table 4 defines the bit assignments associated with the alarm exception detail.

Table 4 Exception Detail Alarm Bit Assignments

Bit	*Device-Specific Alarm [0]*
0	Reserved
1	Indicated Flow High
2	Indicated Flow Low
3	Flow Controller
4	Flow Valve Actuator
5	Reserved
6	Reserved
7	Reserved

8.2.2.3 *Exception Detail Warning (Optional)* — An attribute which identifies the detailed warning status of the device. Table 5 defines the bit assignments associated with the warning exception detail.

Table 5 Exception Detail Warning Bit Assignments

Bit	*Device-Specific Warning [0]*
0	Reserved
1	Indicated Flow High
2	Indicated Flow Low
3	Flow Controller
4	Flow Valve Actuator
5	Reserved
6	Reserved
7	Reserved

8.2.2.4 *Initial and Default Values*

Table 6 DM Object Attribute Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Device Type	MFC, MFM	MFC, MFM	MFM = Mass Flow Meter Device MFC = Mass Flow Controller Device
Exception Detail Alarm	0	0	
Exception Detail Warning	0	0	

8.3 *Sensor Actuator Controller Object (SAC)* — The Sensor Actuator Controller object instance is the device component responsible for coordinating the interaction of the mass flow device with the sensory/actuation/control environment as specified in SEMI E54.1.

8.4 *Sensor-AI-MF Object* — The Sensor-AI-MF object inherits the attributes, services, and behavior of the Sensor-AI as defined in SEMI E54.1. The Sensor-AI-MF object instance is the device component responsible for retrieving a reading from a physical flow sensor, optionally correcting the reading with a manufacturer-specified algorithm, or algorithms, then making the value available to feed the Controller object instance via a Local Link object instance.

8.4.1 *Sensor-AI-MF Object Attributes*

Table 7 Sensor-AI-MF Object Attributes

Attribute Name	*Attribute Identifier*	*Access Network*	*Required*	*Form*
Flow Totalizer	A1	RW	No	REAL
Flow Hours	A2	R	No	ULINT
Zero Offset Mode	A5	RW	No	Enumerated Byte
Zeroing Status	A6	R	No	Enumerated Byte
Autorange Status	A7	R	No	Enumerated Byte
Reserved	A8–A64	—	—	Reserved for future expansion
Manufacturer-Specified	> A64	—	—	Manufacturer-Specific attributes

8.4.1.1 *Flow Totalizer (Optional)* — An attribute that maintains the volume of gas in standard cubic centimeters (SCC) that has flowed through the device since the last time the flow *totalizer* attribute value was set to zero.

8.4.1.2 *Flow Hours (Optional)* — An attribute which identifies the number of hours that the device has been flowing gas since the last time the *flow hours* attribute value was set to zero, as specified by the manufacturer. The attribute is an unsigned long integer with a resolution of 1 hour.

8.4.1.3 *Zero Offset Mode (Optional)* — An attribute which specifies the zero offset formula to be applied to produce the value attribute. This attribute is an enumerated byte that can take on one of the following values:

0 = Disable
1 = Enable Application of Formula
2 = Enable Automatic Zeroing
3–63 = Reserved
64–255 = Manufacturer-Specified

8.4.1.4 *Zeroing Status (Optional)* — An attribute which specifies whether the object is in the ZEROING substate. This attribute is an enumerated byte that can take on one of the following values:

0 = Not Zeroing
1 = Zeroing

8.4.1.5 *Autorange Status (Optional)* — An attribute which specifies whether the object is in the AUTORANGING sub-state. This attribute is an enumerated byte that can take on one of the following values:

0 = Not Autoranging
1 = Autoranging

8.4.1.6 *Initial and Default Values*

Table 8 Sensor-AI-MF Object Attributes Initial and Default Values

Attribute	*Initial Value*	*Default Value*	*Comment*
Flow Totalizer	LVV	0	
Flow Hours	LVV	0	
Zero Offset Mode	LVV	Disable	
Zeroing Status	Not Zeroing	Not Zeroing	
Autorange Status	LVV	Not Autoranging	

8.4.2 *Sensor-AI-MF Object Services* — The services provided by the Sensor-AI-MF object instance are defined in SEMI E54.1. The Sensor-AI-MF object supports the additional services listed below.

8.4.2.1 All Gas Correction services are optional. Gas correction may be handled within the device as a pre-assigned gas calibration or may not be implemented at all. If these services are available, it is manufacturer-specific as to