



2 **Reader Management 1.0**

3 Ratified Standard
4 December 05, 2006

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

Abstract

This document defines Version 1.0 of the wire protocol used by management software to monitor the operating status and health of EPCglobal compliant RFID Readers. This document complements the EPCglobal Reader Protocol Version 1.1 specification [RP1]. In addition, this document defines Version 1.0 of the EPCglobal SNMP RFID MIB.

Audience for this document

The target audience for this specification includes:

- EPC Middleware vendors
- Reader vendors
- Application developers
- Management Software developers
- System integrators

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained by EPCglobal. This document is the first public working draft.

This is an **EPCglobal Working Draft** for review by EPCglobal Members of the Reader Management Working Group. It is a draft document and may be updated, replaced or made obsolete by other documents at any time. It is inappropriate to use EPCglobal Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by the EPCglobal membership.

Comments on this document should be sent to the EPCglobal Software Action Group Reader Management Working Group mailing list
sag_reader_mgmt@epclinklist.epcglobalinc.org
sag_rm_wg@lists.epcglobalinc.org.

TO BE THE BEST

49 Table of Contents

50	Abstract.....	2
51	Audience for this document.....	2
52	Status of this document.....	2
53	1 Introduction	10
54	2 Terminology.....	10
55	3 Protocol Layers	10
56	4 Object Model.....	13
57	4.1 Monitored Objects	14
58	4.1.1 ReaderDevice Object	14
59	4.1.2 CommandChannel.....	16
60	4.1.3 Source Object.....	16
61	4.1.4 ReadPoint Object	18
62	4.1.5 AntennaReadPoint Object.....	18
63	4.1.6 Trigger Object.....	19
64	4.1.7 IOPort Object.....	20
65	4.1.8 AlarmChannel Object	20
66	NotificationChannel Object.....	21
67	4.2 AlarmControl Objects.....	22
68	4.3 Alarm Objects.....	23
69	4.4 Enumerations.....	24
70	5 Reader Layer – Commands.....	25
71	5.1 ReaderDevice.....	27
72	5.1.1 ReaderDevice.getDescription	27
73	5.1.2 ReaderDevice.setDescription.....	28
74	5.1.3 ReaderDevice.getLocationDescription	28
75	5.1.4 ReaderDevice.setLocationDescription.....	29
76	5.1.5 ReaderDevice.getContact.....	30
77	5.1.5.1 ReaderDevice.setContact	30
78	5.1.6 ReaderDevice.getSerialNumber.....	31
79	5.1.7 ReaderDevice.getOperStatus	32
80	5.1.8 ReaderDevice.getOperStatusAlarmControl.....	32
81	5.1.9 ReaderDevice.getFreeMemory	33

82	5.1.10	ReaderDevice.getFreeMemoryAlarmControl	34
83	5.1.11	ReaderDevice.getNTPServers	34
84	5.1.12	ReaderDevice.getDHCPServer.....	35
85	5.1.13	ReaderDevice.getIOPort.....	36
86	5.1.14	ReaderDevice.getAllIOPorts	36
87	5.1.15	ReaderDevice.resetStatistics.....	37
88	5.1.16	ReaderDevice.removeAlarmChannels	38
89	5.1.17	ReaderDevice.removeAllAlarmChannels	39
90	5.1.18	ReaderDevice.getAlarmChannel	39
91	5.1.19	ReaderDevice.getAllAlarmChannels	40
92	5.2	NotificationChannel.....	41
93	5.2.1	NotificationChannel.getLastNotificationAttempt.....	41
94	5.2.2	NotificationChannel.getLastSuccessfulNotification.....	41
95	5.2.3	NotificationChannel.getOperStatus	42
96	5.2.4	NotificationChannel.setAdminStatus.....	43
97	5.2.5	NotificationChannel.getAdminStatus	43
98	5.2.6	NotificationChannel.getOperStatusAlarmControl.....	44
99	5.3	AlarmChannel.....	45
100	5.3.1	AlarmChannel.create	45
101	5.3.2	AlarmChannel.getName.....	46
102	5.3.3	AlarmChannel.getAddress.....	46
103	5.3.4	AlarmChannel.setAddress	47
104	5.4	ReadPoint.....	48
105	5.4.1	ReadPoint.getClassName.....	48
106	5.4.2	ReadPoint.getDescription	49
107	5.4.3	ReadPoint.setDescription.....	49
108	5.4.4	ReadPoint.getAdminStatus	50
109	5.4.5	ReadPoint.setAdminStatus.....	51
110	5.4.6	ReadPoint.getOperStatus	51
111	5.4.7	ReadPoint.getOperStatusAlarmControl.....	52
112	5.5	AntennaReadPoint.....	53
113	5.5.1	AntennaReadPoint.getIdentificationCount	53
114	5.5.2	AntennaReadPoint.getFailedIdentificationCount	53

115	5.5.3	AntennaReadPoint.getMemReadCount.....	54
116	5.5.4	AntennaReadPoint.getFailedMemReadCount.....	55
117	5.5.5	AntennaReadPoint.getFailedMemReadAlarmControl	56
118	5.5.6	AntennaReadPoint.getWriteCount.....	56
119	5.5.7	AntennaReadPoint.getFailedWriteCount.....	57
120	5.5.8	AntennaReadPoint.getFailedWriteAlarmControl.....	58
121	5.5.9	AntennaReadPoint.getKillCount	58
122	5.5.10	AntennaReadPoint.getFailedKillCount	59
123	5.5.11	AntennaReadPoint.getFailedKillAlarmControl	60
124	5.5.12	AntennaReadPoint.getEraseCount	60
125	5.5.13	AntennaReadPoint.getFailedEraseCount	61
126	5.5.14	AntennaReadPoint.getFailedEraseAlarmControl.....	61
127	5.5.15	AntennaReadPoint.getLockCount	62
128	5.5.16	AntennaReadPoint.getFailedLockCount	63
129	5.5.16.1		64
130	5.5.17	AntennaReadPoint.getTimeEnergized	65
131	5.5.17.1		66
132	5.5.18	AntennaReadPoint.getNoiseLevel.....	66
133	5.6	Source Object	67
134	5.6.1	Source.getUnknownToGlimpsedCount.....	67
135	5.6.2	Source.getGlimpsedToUnknownCount.....	67
136	5.6.3	Source.getGlimpsedToObservedCount	68
137	5.6.4	Source.getObservedToLostCount.....	69
138	5.6.5	Source.getLostToGlimpsedCount.....	69
139	5.6.6	Source.getLostToUnknownCount	70
140	5.6.7	Source.getOperStatus.....	70
141	5.6.8	Source.getAdminStatus.....	71
142	5.6.9	Source.setAdminStatus	72
143	5.6.10	Source.getOperStatusAlarmControl.....	72
144	5.7	Trigger Object.....	73
145	5.7.1	Trigger.getFireCount	73
146	5.8	IOPort Object.....	74
147	5.8.1	IOPort.getName	74

148	5.8.2	IOPort.getDescription	74
149	5.8.3	IOPort.setDescription.....	75
150	5.8.4	IOPort.getOperStatus.....	76
151	5.8.5	IOPort.getAdminStatus.....	76
152	5.8.6	IOPort.setAdminStatus	77
153	5.8.7	IOPort.getOperStatusAlarmControl.....	78
154	5.8.8	AlarmControl.getName.....	80
155	5.8.9	AlarmControl.setEnabled	81
156	5.8.10	AlarmControl.setEnabled	81
157	5.8.11	AlarmControl.getLevel.....	82
158	5.8.12	AlarmControl.setLevel	82
159	5.8.13	AlarmControl.getSuppressInterval.....	83
160	5.8.14	AlarmControl.setSuppressInterval	83
161	5.9	EdgeTriggeredAlarmControl.....	84
162	5.9.1	EdgeTriggeredAlarmControl.getAlarmThreshold.....	85
163	5.9.2	EdgeTriggeredAlarmControl.setAlarmThreshold	86
164	5.9.3	EdgeTriggeredAlarmControl.getRearmThreshold	87
165	5.9.4	EdgeTriggeredAlarmControl.setRearmThreshold.....	87
166	5.9.5	EdgeTriggeredAlarmControl.getDirection	88
167	5.9.6	EdgeTriggeredAlarmControl.setDirection.....	88
168	5.9.7	EdgeTriggeredAlarmControl.getStatus	89
169	5.10	TTOperationalStatusAlarmControl.....	90
170	5.10.1	TTOperationalStatusAlarmControl.getTriggerFromState.....	91
171	5.10.2	TTOperationalStatusAlarmControl.setTriggerFromState	92
172	5.10.3	TTOperationalStatusAlarmControl.getTriggerToState.....	92
173	5.10.4	TTOperationalStatusAlarmControl.setTriggerToState	93
174	6	Reader Layer – Alarm Notifications	94
175	6.1	Alarm Objects.....	94
176	6.2	Alarm.....	95
177	6.2.1	Alarm.getReaderDeviceEPC	97
178	6.2.2	Alarm.getReaderDeviceName	97
179	6.2.3	Alarm.getReaderDeviceHandle	98
180	6.2.4	Alarm.getReaderDeviceRole	98

181	6.2.5	Alarm.getTimeTicks	99
182	6.2.6	Alarm.getTimeUTC.....	99
183	6.2.7	Alarm.getName.....	100
184	6.2.8	Alarm.getAlarmLevel	100
185	6.2.9	Alarm.getSuppressCount	101
186	6.3	FreeMemoryAlarm	102
187	6.3.1	FreeMemoryAlarm.getFreeMemory.....	102
188	6.4	FailedWriteAlarm.....	103
189	6.4.1	FailedWriteAlarm.getReadPointName	103
190	6.4.2	FailedWriteAlarm.getFailedWriteCount	104
191	6.4.3	FailedWriteAlarm.getNoiseLevel.....	104
192	6.5	FailedEraseAlarm	105
193	6.5.1	FailedEraseAlarm.getReadPointName	106
194	6.5.2	FailedEraseAlarm.getFailedEraseCount.....	106
195	6.5.3	FailedEraseAlarm.getNoiseLevel	107
196	6.6	FailedKillAlarm.....	107
197	6.6.1	FailedKillAlarm.getReadPointName.....	108
198	6.6.2	FailedKillAlarm.getFailedKillCount	109
199	6.6.3	FailedKillAlarm.getNoiseLevel.....	109
200	6.7	FailedLockAlarm.....	110
201	6.7.1	FailedLockAlarm.getReadPointName.....	110
202	6.7.2	FailedLockAlarm.getFailedLockCount	111
203	6.7.3	FailedLockAlarm.getNoiseLevel.....	111
204	6.8	FailedMemReadAlarm	112
205	6.8.1	FailedMemReadAlarm.getReadPointName.....	113
206	6.8.2	FailedMemReadAlarm.getFailedMemReadCount	113
207	6.8.3	FailedMemReadAlarm.getNoiseLevel	114
208	6.9	TTOperStatusAlarm	115
209	6.9.1	TTOperStatusAlarm.getFromState	115
210	6.9.2	TTOperStatusAlarm.getToState	116
211	6.10	ReaderDeviceOperStatusAlarm	116
212	6.11	IOPortOperStatusAlarm.....	117
213	6.11.1	IOPortOperStatusAlarm.getIOPortName.....	117

214	6.12	ReadPointOperStatusAlarm	118
215	6.12.1	ReadPointOperStatusAlarm.getReadPointName	118
216	6.13	SourceOperStatusAlarm.....	119
217	6.13.1	SourceOperStatusAlarm.getSourceName.....	119
218	6.14	NotificationChannelOperStatusAlarm	120
219	6.14.1	NotificationChannelOperStatusAlarm.getNotificationChannelName	120
220	7	Enumerated types	121
221	7.1.1	AdministrativeStatus.....	121
222	7.1.2	OperationalStatus.....	121
223	7.1.3	EdgeTriggeredAlarmDirection	122
224	7.1.4	EdgeTriggeredAlarmStatus.....	122
225	7.1.5	AlarmLevel	122
226	8	Error Handling.....	123
227	8.1	Error Conditions	123
228	8.2	Communication Errors	123
229	8.2.1	Communication Host-to-Reader	123
230	8.2.2	Communication Reader-to-Host	123
231	8.3	Command Errors.....	124
232	9	Vendor Extensions	126
233	10	Message/Transport Bindings (MTBs).....	128
234	10.1	Address Notation.....	128
235	10.2	Vendor Extension Details.....	128
236	10.3	XML Message Format	129
237	10.3.1	Command XML Message Encoding (Host-To-Reader).....	129
238	10.3.2	Reply XML Message Encoding (Reader-To-Host).....	148
239	10.3.3	Alarm Notification XML Message Encoding (Reader-To-Host).....	175
240	10.3.4	Common Data Formats.....	182
241	10.3.5	EPCglobal Standard Header	183
242	10.4	SNMP MIB	184
243	10.4.1	Vendor Extension Details.....	186
244	10.4.2	EPCglobal RFID Reader Management MIB.....	188
245	10.4.2.1	EPCglobal SMI MIB.....	188
246	10.4.2.2	EPCglobal Reader MIB	188

247	11	Acknowledgements.....	233
248	12	References.....	233
249			
250			

1 Introduction

This document defines Version 1.0 of the wire protocol used by management software to monitor the operating status and health of EPCglobal compliant tag Readers. This document complements the EPCglobal Reader Protocol Version 1.1 specification [RP1]. In addition, this document defines Version 1.0 of the EPCglobal SNMP RFID MIB, and specifies the set of SNMP MIBII groups [MIBII] required to comply with this EPCglobal Reader Management Specification over SNMP. The terms “tag Reader” and “Reader” include RFID tag Readers, supporting any combination of RF protocols, fixed and hand-held, etc. It also includes Readers of other kinds of tags, such as bar codes. Tag Readers, despite the name, may also have the ability to write data into tag memory.

The Reader Management Protocol specifies the interaction between a device capable of interfacing with tags, and management software. These two parties are herein referred to as the Reader and the Host. The Host may be a fully featured Management Console capable of processing SNMP messages, or a dedicated application capable of communicating with the Reader to interface with RFID tags and monitor its health.

The collection of tag data between the Reader and the Host is defined in the EPCglobal Reader Protocol Version 1.1 [RP1]. This document focuses on the communication protocol required to monitor the health of the Reader.

2 Terminology

Within this specification, the terms SHALL, SHALL NOT, SHOULD, SHOULD NOT, MAY, NEED NOT, CAN, and CANNOT are to be interpreted as specified in Annex G of the ISO/IEC Directives, Part 2, 2001, 4th edition [ISODir2]. When used in this way, these terms will always be shown in ALL CAPS; when these words appear in ordinary typeface they are intended to have their ordinary English meaning.

All sections of this document are normative, with the exception of where it is explicitly noted as non-normative.

The following typographical conventions are used throughout the document:

- ALL CAPS type is used for the special terms from [ISODir2] enumerated above.
- Monospace type is used to denote programming language, UML, XML identifiers and other coding constructs, as well as for the text of XML documents.

3 Protocol Layers

This document defines two separate but related management protocol specifications.

1. Specifies the EPCglobal SNMP MIB for monitoring the health of a Reader.
2. Specifies the EPCglobal Reader Management Protocol for monitoring the health of a Reader.

Number 2 follows the same layering structure defined by the EPCglobal Reader Protocol Version 1.1 [RP1]. The Reader Management Protocol is specified in three distinct layers, as illustrated below.

Content of Reader/Host
Exchanges (abstract syntax)

Reader Layer

Message formatting - Message
framing - Security services -
Connection establishment

Messaging Layer

O/S-provided network facility

Transport Layer

*Each pair is
called a
Messaging
/Transport
Binding
(MTB)*

Figure 1 Protocol Layers

The layers are:

- *Reader Layer* This layer specifies the content and abstract syntax of messages exchanged between the Reader and Host. This layer is the heart of the Reader Protocol and the Reader Management Protocol, defining the operations that Readers expose to monitor their health.
- *Messaging layer* This layer specifies how messages defined in the Reader Layer are formatted, framed, transformed, and carried on a specific network transport. Any security services are supplied by this layer. (Examples of security services include authentication, authorization, message confidentiality, and message integrity.) The Messaging Layer specifies how an underlying network connection is established, any initialization messages required to establish synchronization or to initialize security services, and any processing such as encryption that is performed on each message.
- *Transport Layer* This layer corresponds to the networking facilities provided by the operating system or equivalent, and is specified elsewhere.

The Reader Management Protocol specification provides for multiple alternative implementations of the Messaging Layer. Each such implementation is called a Messaging/Transport Binding (MTB). Different MTBs provide for different kinds of transport, e.g., TCP/IP versus Bluetooth versus serial line. Different MTBs may also provide different kinds of security services, means for establishing connections (e.g., whether the Reader contacts the Host or the Host contacts the Reader), and means for provisioning of configuration information.

This specification defines an XML MTB which complements that defined by the Reader Protocol. The Transport details can be found in the Reader Protocol Specification 1.1. The SNMP MIB components are also defined in detail in this specification.

320

321

Content of Reader/Host
Exchanges (abstract syntax)

Reader Management Command Set

Message formatting -
Message framing - Security
services - Connection
establishment

XML

SNMP

O/S-provided network
transport facility

Serial

TCP

UDP

322

323 **Figure 2 Protocol Layers Mapping**

324

325 The SNMP MIB is an example of another Message/Transport binding. The MIB is a
326 structuring and representation of Reader Object Model elements that conforms to the
327 SNMP specification “Structure of Management Information Version 2 (SMIv2)” [STD
328 58, RFC 2578]. The SNMP protocol has a well defined messaging protocol and transport
329 layer for getting and setting information, event notification, and security facilities. The
330 figure above depicts two MTB examples. On the left, the Reader Protocol Message
331 Format using XML as input and output, and TCP as the transport. On the right, SNMP
332 defining the Message formatting and UDP as the message transport. Note that the same
333 Reader Management Command Set applies to both MTBs.

334 The interface between the Reader Layer and the Messaging Layer is defined in terms of
335 message channels, each representing an independent communication between Reader and
336 Host. The Reader Management Protocol leverages the message channels defined in the
337 EPCglobal Reader Protocol Specification 1.1:

- 338 • *Command Channel* The command channel carries requests issued by the Host to the
339 Reader, and responses to these requests carried from the Reader to the Host. All
340 messages exchanged on the command channel follow this request/response pattern.
341 Most configuration interaction between Reader and Host takes place through the
342 command channel. The Reader MAY support one or more *Command Channels*.
- 343 • *Alarm Channel(s)* The alarm channel carries messages issued asynchronously by
344 the Reader to the Host. Messages on the alarm channel only flow to the Host. The
345 alarm channel is primarily used to support a mode of operation in which the Reader
346 asynchronously delivers health and status alerts to the Host, The Reader controls
347 when alarm channel messages are sent to the Host; the Host does not request alarm
348 channel messages. The Reader MAY support one or more Alarm Channels for the
349 delivery of Management Alarms.

- *Notification Channel(s)* In addition, the Reader MAY support one or more Notification Channels for delivery of tag data, as specified in the Reader Protocol Specification 1.1.

The Command channel provides the means to set and get the state, status, or value of attributes of a given object. Under certain conditions, the end-user may choose to disable the setter capabilities, therefore a Reader MAY provide the means to globally disable all *set* methods for all objects. If a set method is invoked after it has been disabled, the ERROR_AUTHORIZATION SHALL be returned.

4 Object Model

The Reader Management Specification defines a set of conceptual objects and operations (a command set) which enables the Host to query the status of these objects in a standard way. This is described using an object model. The Reader Management specification shares a common object model with the Reader Protocol Specification 1.1; however some objects and attributes are not relevant to the Reader Management specification. Figure 3 Reader Monitored Objects UML **Error! Reference source not found.**Figure 3 Reader Monitored Objects UML**Error! Reference source not found.**through

Figure 14 Enumerations UML

Figure 14 Enumerations UML present all of the objects in order to provide a complete model. Only the objects, attributes, and methods relevant to Reader Management are discussed in detail in this specification. Objects that are not relevant to this specification are shown in gray. Object attributes and methods from the Reader Protocol Specification 1.1 and this Reader Management specification are shown in the detailed UML diagrams in Figure 4 ReaderDevice Object UML, Figure 5 Source Object UML, Figure 6 ReadPoint Object UML, Figure 8 Trigger Object UML, and Figure 11 NotificationChannel Object UML. The specific commands defined by this Reader Management specification are listed in detail in Section 5 Reader Layer – Commands, section 6 Reader Layer – Alarm Notifications, and section 7 Enumerated types of this document. Refer to the Reader Protocol Specification 1.1 for a description of the attributes and commands specific to that specification. A compliant Reader NEED NOT implement an object model internally, as long as the command messages can be correctly interpreted and executed, and alarms be correctly generated.

The health of a Reader can be monitored by actively querying Reader for the status of the various monitored objects, and/or by setting asynchronous notification alarms. Each object maintains a set of counters and status monitors that can be queried synchronously. Some objects also implement alarms that are configured to be triggered on a given set of conditions. The AlarmControl Objects define when alarms are generated. The Alarm Objects represent the actual content of the notification or alarm. AlarmControl Objects are defined in Section 4.2. Alarm Objects are defined in Section 4.3.

The following section provides a brief description of the major objects and their relationship to each other. For details on the objects and their operations, refer to Section 5: Reader Layer – Commands

4.1 Monitored Objects

The following sections provide a brief description of the conceptual objects. Each object exposes a set of operations which are described in detail throughout this specification. Figure 3 Reader Monitored Objects UML illustrates the high level relationship between the components. The list of attributes and commands defined for each object are illustrated in each of the sections describing the objects. Objects with no visible attributes and/ or commands, all in white, are illustrated in further detail in a subsequent UML diagram in the specification.

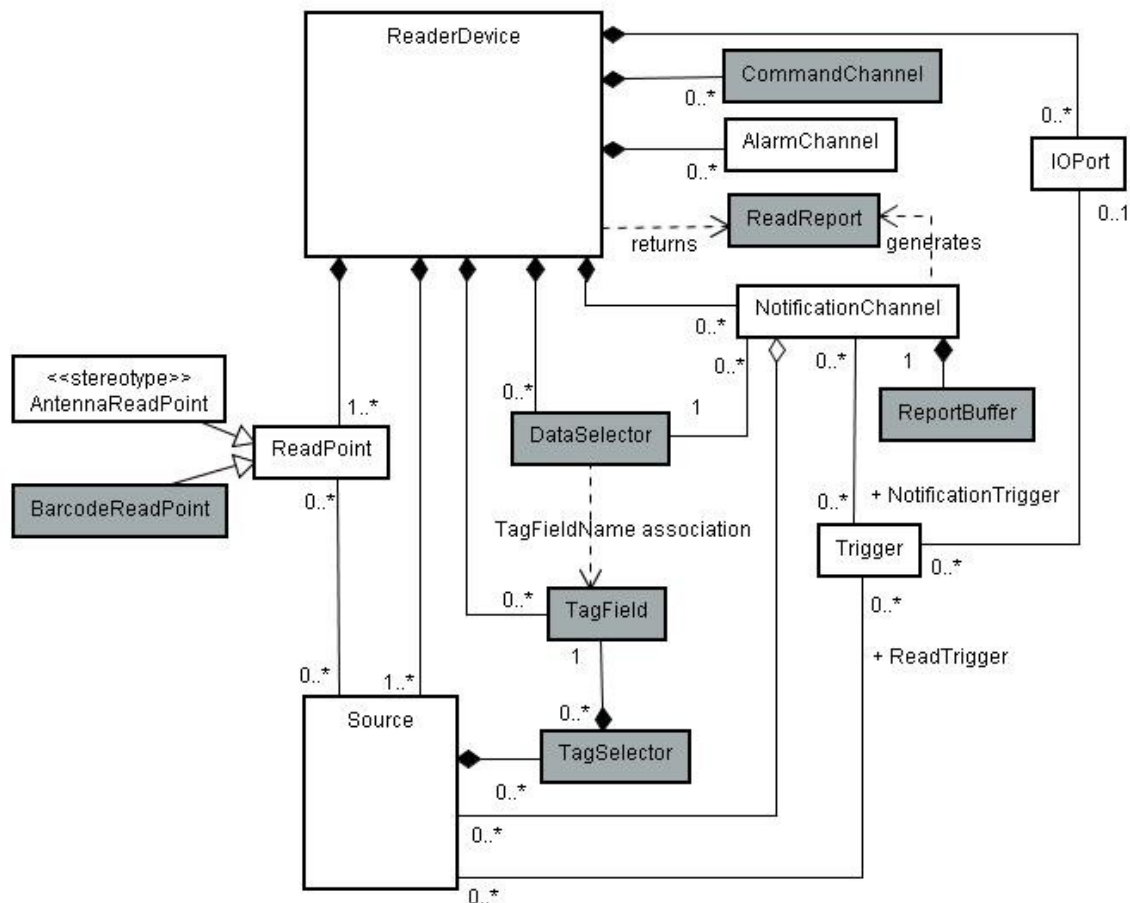


Figure 3 Reader Monitored Objects UML

4.1.1 ReaderDevice Object

The ReaderDevice object is the base container for most other objects in the Reader object model including at least one pre-configured CommandChannel object. The ReaderDevice object also contains several attributes that are used to manage the Reader device and, for network Readers, attributes used to manage the Reader's network interface.



410

411 **Figure 4 ReaderDevice Object UML**

4.1.2 CommandChannel

There are no explicit operations on this object. A compliant Reader MAY have one or more Command Channels with a default address where it is listening to incoming commands. If implemented, this default address SHALL be listed in the documentation of a compliant reader.

4.1.3 Source Object

Sources read tags and present their data to the Data Acquisition stage of the Read Subsystem in a Reader. Two examples of Sources are an RF antenna and a barcode scanner. Source objects are associated to the ReaderDevice object. One or more Sources MAY be associated with a NotificationChannel. In the Reader object model, a Source object can encapsulate one or more input sensors called ReadPoints. Refer to the Reader Protocol Specification 1.1 for further detail.

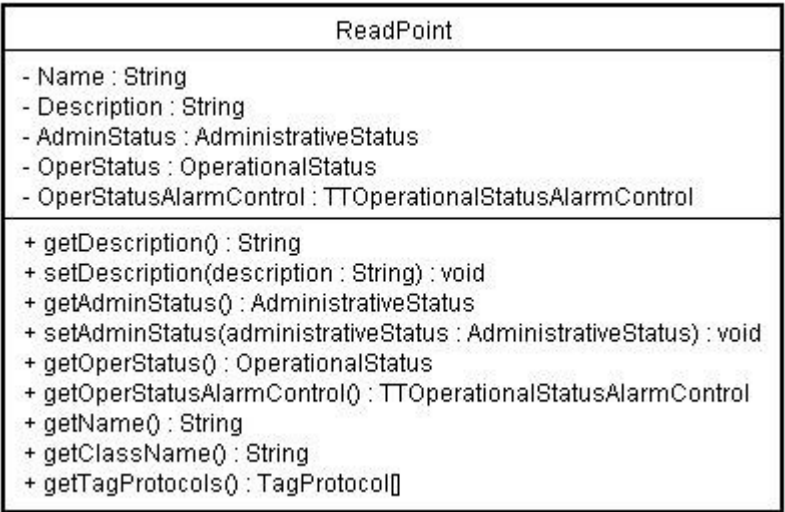
Source
- Name : String - ReadCyclesPerTrigger : int - ReadDutyCycle : int - ReadTimeout : int - GlimpsedTimeout : int - ObservedThreshold : int - ObservedTimeout : int - LostTimeout : int - UnknownToGlimpsedCount : int - GlimpsedToUnknownCount : int - GlimpsedToObservedCount : int - ObservedToLostCount : int - LostToGlimpsedCount : int - LostToUnknownCount : int - AdminStatus : AdministrativeStatus - OperStatus : OperationalStatus - OperStatusAlarmControl : TTOperationalStatusAlarmControl - Session : int
+ getUnknownToGlimpsedCount() : int + getGlimpsedToUnknownCount() : int + getGlimpsedToObservedCount() : int + getObservedToLostCount() : int + getLostToObservedCount() : int + getLostToUnknownCount() : int + getOperStatus() : OperationalStatus + setAdminStatus(administrativeStatus : AdministrativeStatus) : void + getAdminStatus() : AdministrativeStatus + getOperStatusAlarmControl() : TTOperationalStatusAlarmControl + create(name : String) : Source + getName() : String + isReadOnly() : boolean + addReadPoints(readPoints : ReadPoint[]) : void + removeReadPoints(readPoints : ReadPoint[]) : void + removeAllReadPoints() : void + getReadPoint(name : String) : ReadPoint + getAllReadPoints() : ReadPoint[] + addReadTriggers(triggers : Trigger[]) : void + removeReadTriggers(triggers : Trigger[]) : void + removeAllReadTriggers() : void + getReadTrigger(name : String) : Trigger + getAllReadTriggers() : Trigger[] + addTagSelectors(selectors : TagSelector[]) : void + removeTagSelectors(selectors : TagSelector[]) : void + removeAllTagSelectors() : void + getTagSelector(name : String) : TagSelector + getAllTagSelectors() : TagSelector[] + getGlimpsedTimeout() : int + setGlimpsedTimeout(timeout : int) : void + getObservedThreshold() : int + setObservedThreshold(timeout : int) : void + getObservedTimeout() : int + setObservedTimeout(timeout : int) : void + getLostTimeout() : int + setLostTimeout(timeout : int) : void + rawReadIDs(dataSelector : DataSelector) : ReadReport + readIDs(dataSelector : DataSelector) : ReadReport + writeID(ID : binary, lockcode : binary) : void + kill(tagSelector : TagSelector[], data : TagFieldValue[]) : void + getReadCyclesPerTrigger() : int + setReadCyclesPerTrigger(cycles : int) : void + getReadDutyCycle() : int + setReadDutyCycle(dutyCycle : int) : void + getReadTimeout() : int + setReadTimeout(timeout : int) : void + write(tagSelector : TagSelector[], data : TagFieldValue[]) : void + read(tagSelector : TagSelector[], dataSelector : DataSelector) : ReadReport + getSession() : int + setSession(session : int) : void

424

425 **Figure 5 Source Object UML**

426 **4.1.4 ReadPoint Object**

427 A `ReadPoint` can be any physical entity that is capable of acquiring data. A single RF
428 tag reader antenna is a simple example of a `ReadPoint`. In this version of the
429 specification, the only supported `ReadPoint` is an `AntennaReadPoint`. Refer to
430 the Reader Protocol Specification 1.1 for further detail.



431
432 **Figure 6 ReadPoint Object UML**

433 **4.1.5 AntennaReadPoint Object**

434 `AntennaReadPoints` are `ReadPoints` which are physically implemented using
435 antennas. Each `AntennaReadPoint` maintains a number of statistics to assist in
436 monitoring the health of the reader. Refer to the Reader Protocol Specification 1.1 for
437 further detail.

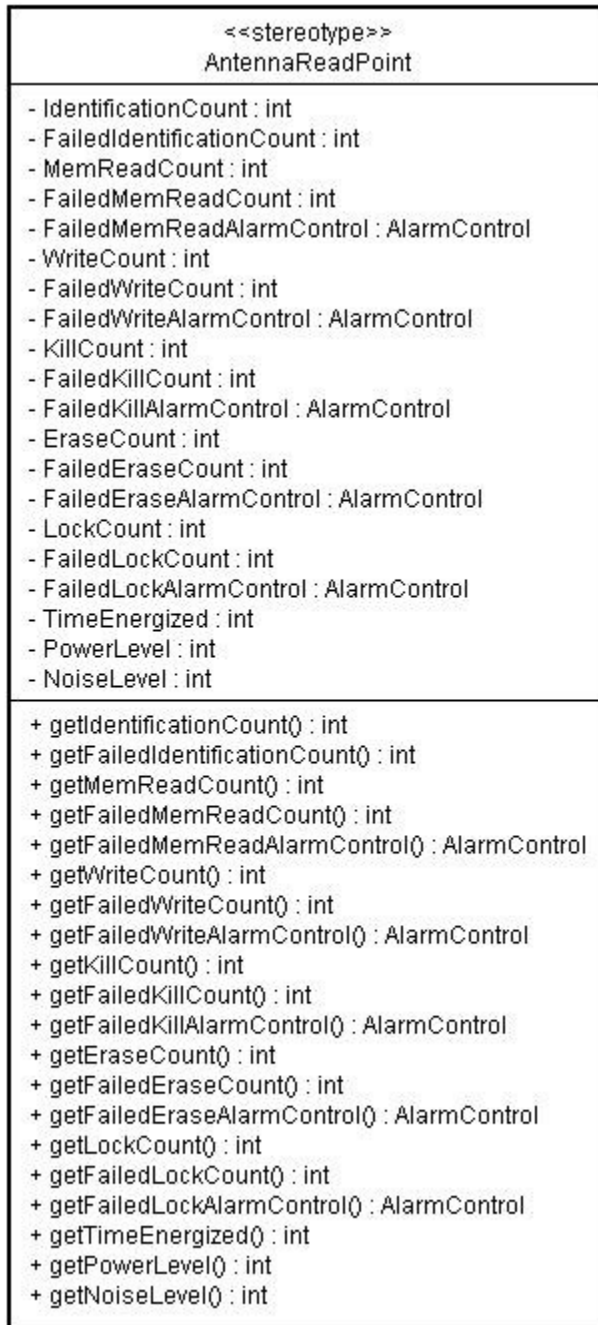


Figure 7 AntennaReadPoint Object UML

4.1.6 Trigger Object

Trigger is a non-mutable object that can exist as a composite member of either a Source or a NotificationChannel object, i.e., Trigger objects are added and removed from Source and NotificationChannel objects. Refer to the Reader Protocol Specification 1.1 for further detail.

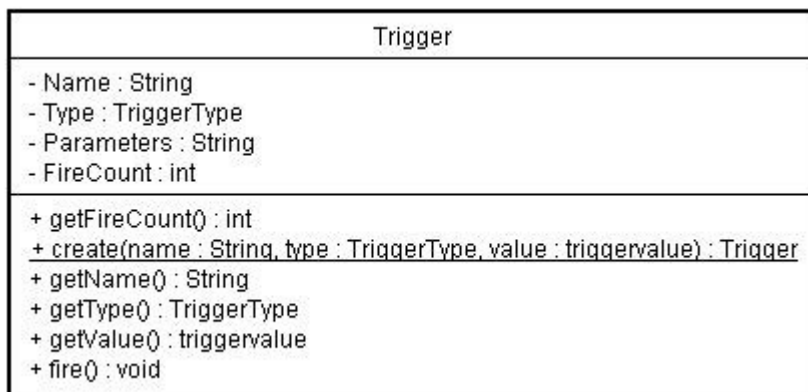


Figure 8 Trigger Object UML

4.1.7 IOPort Object

An IOPort provides the description of the hardware element that provides external input and output lines to connect to other components outside the reader device. A hardware vendor is free to define an IOPort as it best fits its needs. For example, one vendor may choose to define a single IOPort for a 9-pin RS232 connector used to interface with light switches or motion sensors. A second vendor may choose to assign a unique IOPort to each of the pins, if it is capable of configuring and monitoring the state of each pin independently. If a reader device exposes its IOPorts, it SHALL implement the methods identified as such. Refer to the Reader Protocol Specification 1.1 for further detail.

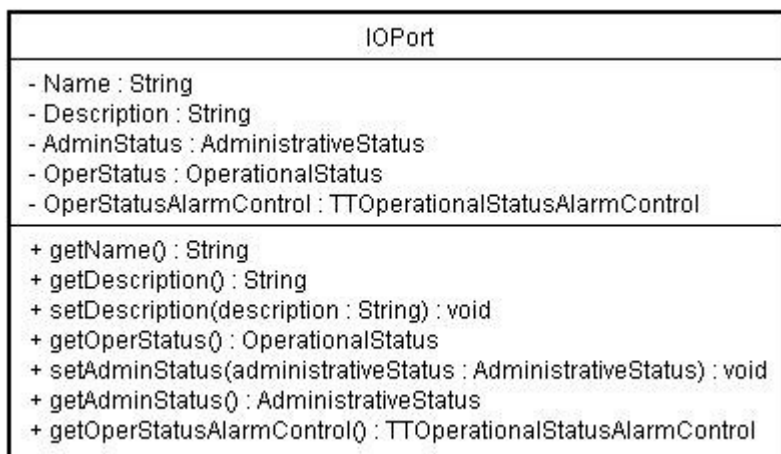
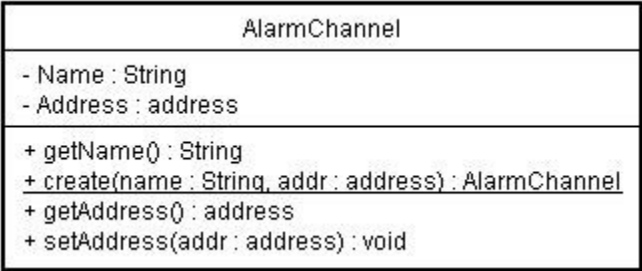


Figure 9 IOPort Object UML

4.1.8 AlarmChannel Object

The AlarmChannel carries messages issued asynchronously by the Reader to the Host. Messages on an AlarmChannel only flow in this direction. AlarmChannels MAY be implemented if a Reader is capable of notifying a host of changes in its health status.

463 It is strongly advised, but not required, to create/define `AlarmChannels` using the
464 commands specified in this specification. A compliant system SHALL provide the
465 means to create/ define `AlarmChannels` either using the commands specified in this
466 specification, or through a Reader specific means, such as through the use of an out-of-
467 band protocol, or via a configuration file.



468

469 **Figure 10 AlarmChannel Object UML**

470 **NotificationChannel Object**

471 A `NotificationChannel` object is responsible for communicating information to a
472 host asynchronously. One or more Sources MAY be associated with a
473 `NotificationChannel`. A `DataSelector` determines what data is reported to the host. Refer
474 to the Reader Protocol Specification 1.1 for further detail.



Figure 11 NotificationChannel Object UML

4.2 AlarmControl Objects

AlarmControl is the base class for all classes responsible for controlling the generation of Alarms. AlarmControls SHALL be implemented if AlarmChannels are implemented. In order to configure an Alarm on an object, the monitoring host needs to obtain a reference to the AlarmControl and manipulate its attributes. An Alarm object represents the actual notification sent to the monitoring host. The UML for the AlarmControl objects is illustrated in Figure 12 Alarm Control UML.

There are two types of AlarmControls defined by the subclasses:

- EdgeTriggeredAlarmControl
-
- TTOperationalStatusAlarmControl

•

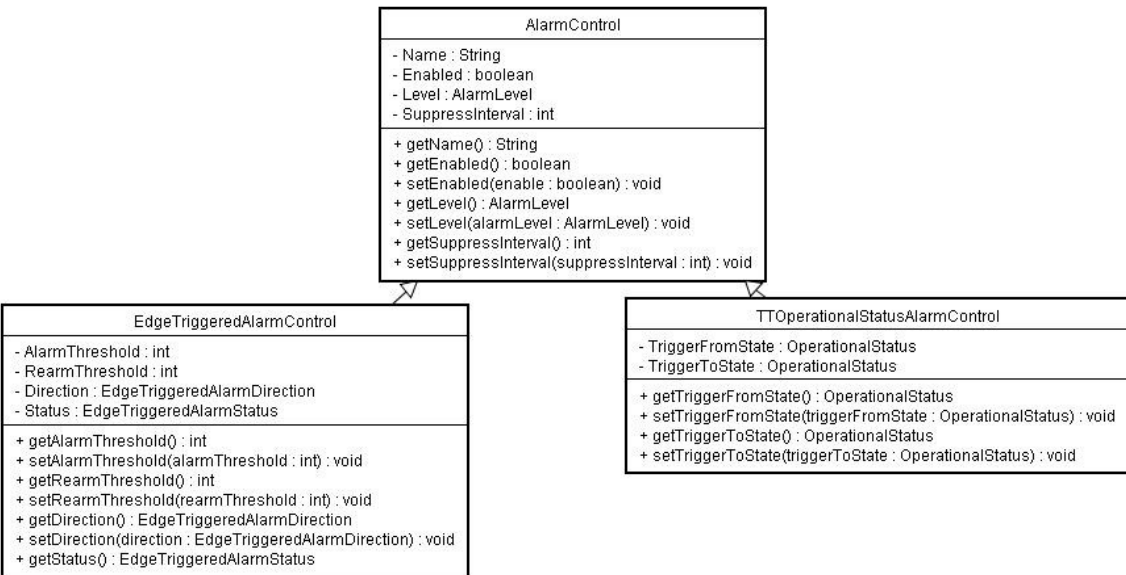


Figure 12 Alarm Control UML

4.3 Alarm Objects

The alarms themselves are messages that are sent to management systems and host applications. Alarm objects specify, in an abstract manner, the contents of these alarm messages. These Alarm objects do not specify persistent data objects maintained by the readers; rather they are instantiated in the course of a reader's generation or a management system's receipt and processing of alarms. Alarm objects SHALL be implemented if AlarmChannels are implemented. The UML for the Alarm objects is illustrated in Figure 13 Alarm UML.

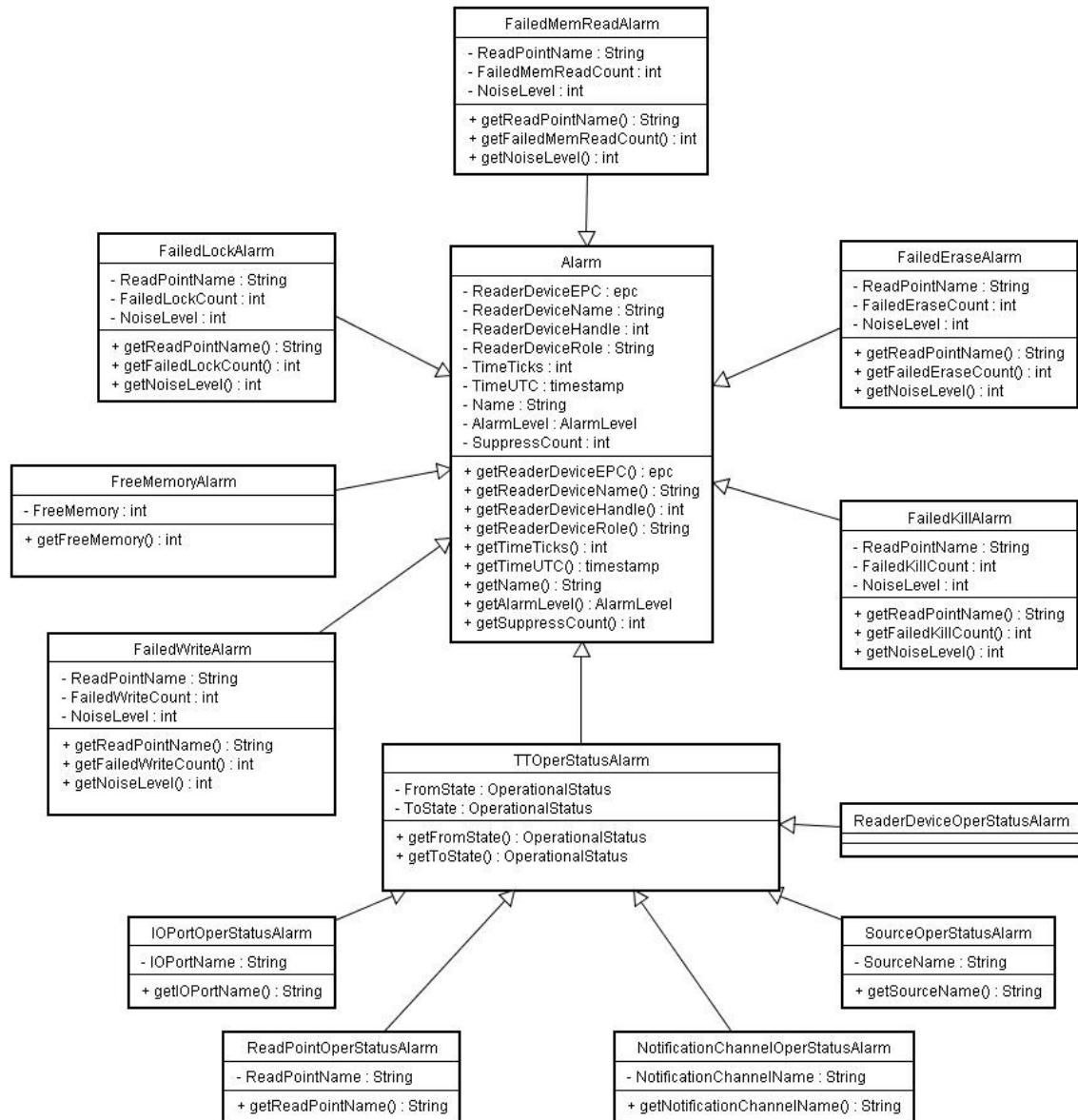


Figure 13 Alarm UML

4.4 Enumerations

The Enumerations contain sets of values used by other objects, such as the administrative Status, Operational Status, Alarm levels, the status of EdgeTriggered Alarms, directions of an alarm value, and the list of operators used to compare alarm levels. The UML for the Enumerations is illustrated in

Figure 14 Enumerations UML

Figure 14 Enumerations UML.

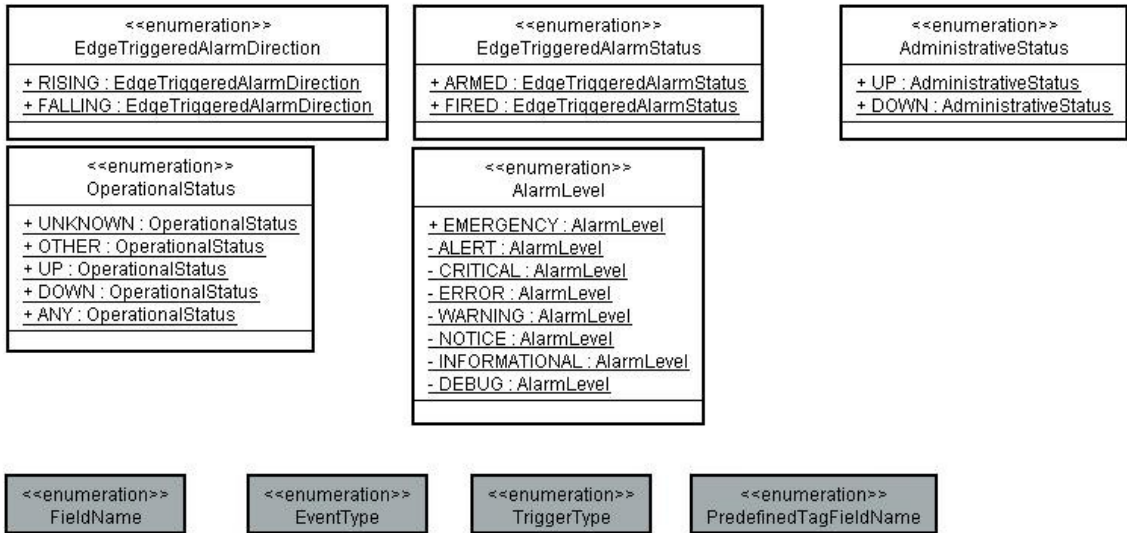


Figure 14 Enumerations UML

5 Reader Layer – Commands

This Reader Management Specification defines a set of operations (a command set) which enables the Host to query the status of these objects in a standard way.

This section specifies messages exchanged at the Reader Layer in order to monitor the health of the Reader.

Messages on the command channel follow a request/response pattern, where the Host sends a request message, and the Reader responds later with a response message (be it a normal response or error response). Messages on the alarm channel are sent asynchronously from Reader to Host.

Reader Layer messages fall into the main functional groups defined by the objects in the previous section. The messages described in this specification are complementary to those defined in the Reader Management section of the Reader Protocol Version 1.1 Specification.

Commands are described below in an abstract manner using a UML notation. This notation is identical to that used by the Reader Protocol Version 1.1.

Commands fall into three categories:

- DO-type commands that cause the Reader to take action.

575 A timestamp (date and time) with millisecond accuracy. The exact format depends on
576 the MTB in use.

577 • ***type[]***

578 A list of the specified data type. The list is unordered unless explicitly stated
579 otherwise.

580 • ***void***

581 No data, indicating no parameters or return value (depending on where it's used).

582

583 In addition, many commands have object types as parameter or return value. The MTB
584 defines how objects are encoded in the command and reply messages. The following
585 section defines the Reader Management Command Set. A list of allowed errors is listed
586 for each command. Section 8 Error Handling describes the Error conditions in detail.

587

588 **5.1 ReaderDevice**

589 **5.1.1 ReaderDevice.getDescription**

590 The Host queries the Reader for its user defined description. This is a text string that
591 describes the general use of this particular reader.

592

593 **Compliance Requirement:** Compliant systems SHALL implement this command.

594

595 **Usage:**

596 `ReaderDevice.getDescription (void): string`

597

598 **Parameter(s):**

599 Data Type: `void`. This command takes no parameters.

600

601 **Return Value(s):**

602 Data Type: `string`. Reader will return its user defined description.

603

604 **Possible Error Conditions:**

605

606 `ERROR_UNKNOWN`

607

608

5.1.2 ReaderDevice.setDescription

The Host sets the Readers user-defined description. This is a text string that describes the general use of this particular reader.

Compliance Requirement: Compliant systems SHALL implement this command.

Usage:

```
ReaderDevice.setDescription (description: string): void
```

Parameter(s):

description – Data Type: string. A text string that describes the general use of this particular reader.

Return Value(s):

Data Type: void. This command will not return a value.

Possible Error Conditions:

ERROR_PARAMETER_MISSING
ERROR_PARAMETER_INVALID_DATATYPE
ERROR_PARAMETER_LENGTH_EXCEEDED
ERROR_UNKNOWN
ERROR_AUTHORIZATION

5.1.3 ReaderDevice.getLocationDescription

The Host queries the Reader for its user defined location description. This is a textual description of the location of the reader device.

Compliance Requirement: Compliant systems SHALL implement this command.

Usage:

```
ReaderDevice.getLocationDescription (void): string
```

642

643 **Parameter(s):**

644 Data Type: `void`. This command takes no parameters.

645

646 **Return Value(s):**

647 Data Type: `string`. Reader will return its user defined location description.

648

649 **Possible Error Conditions:**

650

651 `ERROR_UNKNOWN`

652

653

654 **5.1.4 ReaderDevice.setLocationDescription**

655 The Host sets the Readers user-defined location description. This is a textual description
656 of the location of the reader device.

657

658 **Compliance Requirement:** Compliant systems SHALL implement this command.

659

660 **Usage:**

661 `ReaderDevice.setLocationDescription (locationDescription:`
662 `string): void`

663

664 **Parameter(s):**

665 `locationDescription` – Data Type: `string`. A textual description of the
666 location of the reader device.

667

668 **Return Value(s):**

669 Data Type: `void`. This command will not return a value.

670

671 **Possible Error Conditions:**

672

673 `ERROR_PARAMETER_MISSING`

674 `ERROR_PARAMETER_INVALID_DATATYPE`

675 ERROR_PARAMETER_LENGTH_EXCEEDED

676 ERROR_UNKNOWN

677 ERROR_AUTHORIZATION

678

679 **5.1.5 ReaderDevice.getContact**

680 The Host queries the Reader for its user defined contact description. This information
681 identifies the individual or organization responsible for administering the Reader.

682 **Compliance Requirement:** Compliant systems SHALL implement this command if, and
683 only if, the system implements setContact.

684

685 **Usage:**

686 ReaderDevice.getContact (void): string

687

688 **Parameter(s):**

689 Data Type: void. This command takes no parameters.

690

691 **Return Value(s):**

692 Data Type: string. Reader will return a contact information to the person responsible
693 for the device or someone acting for him/her.

694

695 **Possible Error Conditions:**

696 ERROR_COMMAND_NOT_SUPPORTED

697 ERROR_UNKNOWN

698 For a detailed error description see Reader Protocol chapter 8 Error Handling.

699

700

701 **5.1.5.1 ReaderDevice.setContact**

702 The Host sets the Readers user-defined contact description. This information identifies
703 the individual or organization responsible for administering the Reader..

704

705

706 **Compliance Requirement:** Compliant systems SHALL implement this command.

707

708 **Usage:**
709 `ReaderDevice.setContact (contact: string): void`
710
711 **Parameter(s):**
712 `contact` – Data Type: `string`. A contact information to the person responsible for
713 the device or someone acting for him/her.
714
715 **Return Value(s):**
716 Data Type: `void`. This command will not return a value.
717
718 **Possible Error Conditions:**
719 `ERROR_COMMAND_NOT_SUPPORTED`
720 `ERROR_PARAMETER_MISSING`
721 `ERROR_PARAMETER_INVALID_DATATYPE`
722 `ERROR_PARAMETER_LENGTH_EXCEEDED`
723 `ERROR_UNKNOWN`
724 `ERROR_AUTHORIZATION`
725

726 **5.1.6 ReaderDevice.getSerialNumber**

727 The Host queries the Reader for its serial number. This is the serial number of the reader
728 device. Note that normally the serial number is also an integral part of the EPC. Since the
729 serial number is set by the manufacturer, there is no `setSerialNumber()` command.
730

731 **Compliance Requirement:** Compliant systems SHALL implement this command.
732

733 **Usage:**
734 `ReaderDevice.getSerialNumber (void): string`
735

736 **Parameter(s):**
737 Data Type: `void`. This command takes no parameters.
738

739 **Return Value(s):**
740 Data Type: `string`. Reader will return the serial number of the reader device.

741

742 **Possible Error Conditions:**

743 ERROR_UNKNOWN

744 For a detailed error description see Reader Protocol chapter 8 Error Handling.

745

746

747 **5.1.7 ReaderDevice.getOperStatus**

748 The Host queries the Reader for the ReaderDevice's OperStatus attribute, which
749 is of type OperationalStatus.

750

751 **Compliance Requirement:** Compliant systems SHALL implement this command.

752

753 **Usage:**

754 ReaderDevice.getOperStatus (void): OperationalStatus

755

756 **Parameter(s):**

757 Data Type: void. This command takes no parameters.

758

759 **Return Value(s):**

760 Data Type: OperationalStatus. Reader will return the ReaderDevice object's
761 OperStatus attribute.

762

763 **Possible Error Conditions:**

764

765 ERROR_UNKNOWN

766

767 **5.1.8 ReaderDevice.getOperStatusAlarmControl**

768 The Host queries the ReaderDevice for its OperStatusAlarmControl attribute,
769 of type TOperationalStatusAlarmControl. This attribute is the object that
770 controls the conditions for generating alarms alerting a manager of changes in a
771 ReaderDevice's operational status.

772

773 **Compliance Requirement:** Compliant systems SHALL implement this command if, and
774 only if, AlarmChannels are implemented.

775

776 **Usage:**

777 `ReaderDevice.getOperStatusAlarmControl (void) :`
778 `TTOperationalStatusAlarmControl`

779

780 **Parameter(s):**

781 Data Type: `void`. This command takes no parameters.

782

783 **Return Value(s):**

784 Data Type: `TTOperationalStatusAlarmControl`. The `ReaderDevice` will
785 return its `OperStatusAlarmControl` attribute.

786

787 **Possible Error Conditions:**

788 `ERROR_COMMAND_NOT_SUPPORTED`

789 `ERROR_UNKNOWN`

790

791 **5.1.9 ReaderDevice.getFreeMemory**

792 The Host queries the Reader for the amount of free memory available on the
793 `ReaderDevice`. The `FreeMemory` attribute represents the number of kilobytes (KB)
794 of Read/Write memory available to internal or external resources for buffering or other
795 processing of information. For example, the `FreeMemory` attribute may include the
796 number of kilobytes available for building a `ReportBuffer` (see RP1).

797

798 **Compliance Requirement:** Compliant systems MAY implement this command.

799

800 **Usage:**

801 `ReaderDevice.getFreeMemory (void) : integer`

802

803 **Parameter(s):**

804 Data Type: `void`. This command takes no parameters.

805

806 **Return Value(s):**

807 Data Type: `integer`. Reader will return its available free memory specified in kilo
808 bytes (KB).

809

810 **Possible Error Conditions:**

811 ERROR_COMMAND_NOT_SUPPORTED

812 ERROR_UNKNOWN

813 For a detailed error description see Reader Protocol chapter 8 Error Handling.

814

815

816 **5.1.10 ReaderDevice.getFreeMemoryAlarmControl**

817 The Host queries the Reader for the ReaderDevice object's

818 FreeMemoryAlarmControl attribute, of type EdgeTriggeredAlarmControl.

819 This object controls the generation of alarms alerting a managing system of a reduction of
820 a reader's free memory resources below a specified threshold.

821

822 **Compliance Requirement:** Compliant systems MAY implement this command.

823

824 **Usage:**

825 ReaderDevice.getFreeMemoryAlarmControl (void) :

826 EdgeTriggeredAlarmControl

827

828 **Parameter(s):**

829 Data Type: void. This command takes no parameters.

830

831 **Return Value(s):**

832 Data Type: EdgeTriggeredAlarmControl. Reader will return its

833 FreeMemoryAlarmControl object of type EdgeTriggeredAlarmControl.

834

835 **Possible Error Conditions:**

836 ERROR_COMMAND_NOT_SUPPORTED

837 ERROR_UNKNOWN

838

839 **5.1.11 ReaderDevice.getNTPServers**

840 The Host queries the Reader for a list of NTP servers used by the device to synchronize
841 its current UTC clock (TimeUTC).

842

843 **Compliance Requirement:** Compliant systems MAY implement this command.
844
845 **Usage:**
846 `ReaderDevice.getNTPServers (void): string[]`
847
848 **Parameter(s):**
849 Data Type: `void`. This command takes no parameters.
850
851 **Return Value(s):**
852 Data Type: `string[]`. Reader will return a list of NTP servers used by the device to
853 synchronize its current UTC clock (TimeUTC).
854
855 **Possible Error Conditions:**
856 `ERROR_COMMAND_NOT_SUPPORTED`
857 `ERROR_UNKNOWN`
858

859 **5.1.12 ReaderDevice.getDHCPServer**

860 The Host queries the Reader for the DHCP server currently used by the device for DHCP
861 requests.

862

863 **Compliance Requirement:** Compliant systems MAY implement this command.

864

865 **Usage:**

866 `ReaderDevice.getDHCPServer (void): string`

867

868 **Parameter(s):**

869 Data Type: `void`. This command takes no parameters.

870

871 **Return Value(s):**

872 Data Type: `string`. Reader will return the DHCP server currently used by the device
873 for DHCP requests.

874

875 **Possible Error Conditions:**

876 ERROR_COMMAND_NOT_SUPPORTED

877 ERROR_UNKNOWN

878

879 **5.1.13 ReaderDevice.getIOPort**

880 Returns the IOPort with the specified name currently associated with this Reader. If no
881 IOPort object with the given name exists, the error ERROR_IOPORT_NOT_FOUND
882 is raised.

883

884 **Compliance Requirement:** Compliant systems SHALL implement this command.

885

886 **Usage:**

887 ReaderDevice.getIOPort (name: string): IOPort

888

889 **Parameter(s):**

890 name – Data Type: String. A name for the IOPort object which should be retrieved
891 from the reader.

892

893 **Return Value(s):**

894 Data Type: IOPort. Reader will return the named IOPort object.

895

896 **Possible Error Conditions:**

897 ERROR_COMMAND_NOT_SUPPORTED

898 ERROR_IOPORT_NOT_FOUND

899 ERROR_PARAMETER_MISSING

900 ERROR_PARAMETER_INVALID_DATATYPE

901 ERROR_PARAMETER_LENGTH_EXCEEDED

902 ERROR_UNKNOWN

903 **5.1.14 ReaderDevice.getAllIOPorts**

904 The Host queries the Reader for all its IOPort objects.

905

906 **Compliance Requirement:** Compliant systems SHALL implement this command.

907

908 **Usage:**

909 ReaderDevice.getAllIOPorts (void): IOPort[]
 910
 911 **Parameter(s):**
 912 Data Type: void. This command takes no parameters.
 913
 914 **Return Value(s):**
 915 Data Type: IOPort[] . Reader will return all its IOPort objects.
 916
 917 **Possible Error Conditions:**
 918 ERROR_COMMAND_NOT_SUPPORTED
 919 ERROR_UNKNOWN
 920

921 **5.1.15 ReaderDevice.resetStatistics**

922 Instructs the reader to reset all statistical counters.
 923 Upon receiving this command, the reader SHALL set all supported counters to zero.
 924 Counters which are affected (if the reader supports them) are:

Object	Counters to reset
AntennaReadPoint	IdentificationCount, FailedIdentificationCount, MemReadCount, FailedMemReadCount, WriteCount, FailedWriteCount, KillCount, FailedKillCount, EraseCount, FailedEraseCount, LockCount, FailedLockCount, TimeEnergized
Source	UnknownToGlimpsedCount, GlimpsedToUnknownCount, GlimpsedToObservedCount, ObservedToLostCount, LostToGlimpsedCount, LostToUnknownCount
Trigger	FireCount

925
 926 **Compliance Requirement:** Compliant systems SHALL implement this command.
 927

928 **Usage:**
 929 ReaderDevice.resetStatistics (void): void
 930
 931 **Parameter(s):**
 932 Data Type: void. This command takes no parameters.
 933

934 **Return Value(s):**

935 Data Type: void. This command will not return a value.

936

937 **Possible Error Conditions:**

938 ERROR_COMMAND_NOT_SUPPORTED

939

940 **5.1.16 ReaderDevice.removeAlarmChannels**

941 Removes the specified AlarmChannels from the list of AlarmChannels currently
942 associated with this Reader. If one or more of the AlarmChannels given are not
943 known, or if some of the AlarmChannels to be removed are currently not associated
944 with this Reader, these are ignored and all other AlarmChannels SHALL be removed
945 and the command SHALL complete successfully.

946

947 **Compliance Requirement:** Compliant systems SHALL implement this command, if and
948 only if, AlarmChannel.create() is implemented.

949

950 **Usage:**

951 ReaderDevice.removeAlarmChannels (channels:
952 AlarmChannel[]): void

953

954 **Parameter(s):**

955 channels[] – Data Type: AlarmChannel. An array with AlarmChannel objects to
956 be removed from the Reader.

957

958 **Return Value(s):**

959 Data Type: void. This command will not return a value.

960

961 **Possible Error Conditions:**

962 ERROR_COMMAND_NOT_SUPPORTED

963 ERROR_PARAMETER_MISSING

964 ERROR_PARAMETER_INVALID_DATATYPE

965 ERROR_UNKNOWN

966

5.1.17 ReaderDevice.removeAllAlarmChannels

Removes all AlarmChannels currently associated with this Reader.

Compliance Requirement: Compliant systems SHALL implement this command if, and only if, AlarmChannel is implemented.

Usage:

```
ReaderDevice.removeAllAlarmChannels (void): void
```

Parameter(s):

Data Type: void. This command takes no parameters.

Return Value(s):

Data Type: void. This command will not return a value.

Possible Error Conditions:

ERROR_COMMAND_NOT_SUPPORTED

ERROR_UNKNOWN

5.1.18 ReaderDevice.getAlarmChannel

Returns the AlarmChannel with the specified name currently associated with this Reader. If no AlarmChannel object with the given name exists, the error ERROR_ALARM_CHANNEL_NOT_FOUND is raised.

Compliance Requirement: Compliant systems SHALL implement this command if, and only if, AlarmChannel.create() is implemented.

Usage:

```
ReaderDevice.getAlarmChannel (name: string): AlarmChannel
```

Parameter(s):

name – Data Type: String. A name for the AlarmChannel object which should be retrieved from the reader.

1001 **Return Value(s):**

1002 Data Type: `AlarmChannel`. Reader will return the named `AlarmChannel` object.

1003

1004 **Possible Error Conditions:**

1005 `ERROR_COMMAND_NOT_SUPPORTED`

1006 `ERROR_ALARM_CHANNEL_NOT_FOUND`

1007 `ERROR_PARAMETER_MISSING`

1008 `ERROR_PARAMETER_INVALID_DATATYPE`

1009 `ERROR_PARAMETER_LENGTH_EXCEEDED`

1010 `ERROR_UNKNOWN`

1011

1012 **5.1.19 ReaderDevice.getAllAlarmChannels**

1013 Returns all `AlarmChannels` currently associated with the Reader. If no

1014 `AlarmChannel` are currently associated with this object, the command SHALL

1015 complete successfully and an empty list SHALL be returned.

1016

1017 **Compliance Requirement:** Compliant systems SHALL implement this command if, and

1018 only if, `AlarmChannel.create()` is implemented.

1019

1020 **Usage:**

1021 `ReaderDevice.getAllAlarmChannels (void): AlarmChannel[]`

1022

1023 **Parameter(s):**

1024 Data Type: `void`. This command takes no parameters.

1025

1026 **Return Value(s):**

1027 Data Type: `AlarmChannel[]`. Reader will return an array with all defined

1028 `AlarmChannel` objects for this Reader.

1029

1030 **Possible Error Conditions:**

1031 `ERROR_COMMAND_NOT_SUPPORTED`

1032 `ERROR_UNKNOWN`

1033

5.2 NotificationChannel

5.2.1 NotificationChannel.getLastNotificationAttempt

The Host queries the NotificationChannel object for the timestamp (TimeTicks) when the last attempt was made to send a notification to the given address.

Compliance Requirement: Compliant systems MAY implement this command.

Usage:

```
NotificationChannel.getLastNotificationAttempt (void) :  
timestamp
```

Parameter(s):

Data Type: void. This command takes no parameters.

Return Value(s):

Data Type: timestamp. NotificationChannel object will return the timestamp (TimeTicks) when the last attempt was made to send a notification to the given address.

Possible Error Conditions:

ERROR_COMMAND_NOT_SUPPORTED

ERROR_UNKNOWN

For a detailed error description see Reader Protocol chapter 8 Error Handling.

5.2.2 NotificationChannel.getLastSuccessfulNotification

The Host queries the NotificationChannel object for the timestamp (TimeTicks) when the last successful notification was send to the given address.

Compliance Requirement: Compliant systems MAY implement this command.

Usage:

1066 NotificationChannel.getLastSuccessfulNotification (void):
1067 timestamp

1068

1069 **Parameter(s):**

1070 Data Type: void. This command takes no parameters.

1071

1072 **Return Value(s):**

1073 Data Type: timestamp. NotificationChannel object will return the timestamp
1074 (TimeTicks) when the last successful notification was send to the given address.

1075

1076 **Possible Error Conditions:**

1077 ERROR_COMMAND_NOT_SUPPORTED

1078 ERROR_UNKNOWN

1079

1080 **5.2.3 NotificationChannel.getOperStatus**

1081 The Host queries the NotificationChannel for its OperationalStatus attribute.

1082

1083 **Compliance Requirement:** Compliant systems SHALL implement this command if, and
1084 only if, the system implements Notification Channels.

1085

1086 **Usage:**

1087 NotificationChannel.getOperStatus (void): OperationalStatus

1088

1089 **Parameter(s):**

1090 Data Type: void. This command takes no parameters.

1091

1092 **Return Value(s):**

1093 Data Type: OperationalStatus. NotificationChannel will return its
1094 OperationalStatus attribute.

1095

1096 **Possible Error Conditions:**

1097 ERROR_COMMAND_NOT_SUPPORTED

1098 ERROR_UNKNOWN

1099

1100 **5.2.4 NotificationChannel.setAdminStatus**
1101 The Host sets the NotificationChannels AdminStatus object of type AdministrativeStatus.
1102 The administrative state identifies if the NotificationChannel has been configured to be
1103 operational. For example: UP or DOWN.
1104
1105 **Compliance Requirement:** Compliant systems SHALL implement this command if the
1106 system implements Notification Channels.
1107
1108 **Usage:**
1109 NotificationChannel.setAdminStatus (administrativeStatus:
1110 AdministrativeStatus): void
1111
1112 **Parameter(s):**
1113 administrativeStatus – Data Type: AdministrativeStatus. The
1114 administrative state for the NotificationChannel.
1115
1116 **Return Value(s):**
1117 Data Type: void. This command will not return a value.
1118
1119 **Possible Error Conditions:**
1120
1121 ERROR_PARAMETER_MISSING
1122 ERROR_PARAMETER_INVALID_DATATYPE
1123 ERROR_UNKNOWN
1124
1125 **5.2.5 NotificationChannel.getAdminStatus**
1126 The Host queries the NotificationChannel for its AdminStatus object of type
1127 AdministrativeStatus.
1128
1129 **Compliance Requirement:** Compliant systems SHALL implement this command if, and
1130 only if, the system implements Notification Channels.
1131
1132 **Usage:**
1133 NotificationChannel.getAdminStatus (void):
1134 AdministrativeStatus

1135

1136 **Parameter(s):**

1137 Data Type: void. This command takes no parameters.

1138

1139 **Return Value(s):**

1140 Data Type: AdministrativeStatus. NotificationChannel will return its

1141 AdminStatus object of type AdministrativeStatus.

1142

1143 **Possible Error Conditions:**

1144 ERROR_COMMAND_NOT_SUPPORTED

1145 ERROR_UNKNOWN

1146

1147 **5.2.6 NotificationChannel.getOperStatusAlarmControl**

1148 The Host queries the NotificationChannel object for its

1149 OperStatusAlarmControl attribute of type

1150 TTOperationalStatusAlarmControl.. This attribute is the object that controls

1151 the conditions for generating alarms alerting a manager of changes in a

1152 NotificationChannel's operational status.

1153

1154 **Compliance Requirement:** Compliant systems SHALL implement this command if, and
1155 only if, the system implements Notification Channels and AlarmChannels.

1156

1157 **Usage:**

1158 NotificationChannel.getOperStatusAlarmControl (void):

1159 TTOperationalStatusAlarmControl

1160

1161 **Parameter(s):**

1162 Data Type: void. This command takes no parameters.

1163

1164 **Return Value(s):**

1165 Data Type: TTOperationalStatusAlarmControl. The NotificationChannel

1166 will return its OperStatusAlarmControl object of type TTOperationalStatusAlarmControl.

1167

1168 **Possible Error Conditions:**

1169 ERROR_COMMAND_NOT_SUPPORTED

1170 ERROR_UNKNOWN

1171

1172 **5.3 AlarmChannel**

1173 NOTE: AlarmChannel objects are to be treated as RP Notification Channels, which
1174 means that they use the same handshake and connection mechanisms. See [RP1] for
1175 details.

1176 **5.3.1 AlarmChannel.create**

1177 Create an AlarmChannel object with a given name. If an AlarmChannel object
1178 with the same name exists already, an error is returned. This is a static method.

1179

1180 The AlarmChannel SHALL implicitly be added to the list of all AlarmChannels
1181 kept by the ReaderDevice object.

1182

1183 **Compliance Requirement:** Compliant systems MAY implement this command. If it is
1184 not implemented, and Alarm Notifications are implemented, the system SHALL provide
1185 an alternate method of defining AlarmChannels.

1186

1187 **Usage:**

```
1188 static AlarmChannel.create(  
1189     name: string,  
1190     addr: address): AlarmChannel
```

1191

1192 **Parameter(s):**

1193 name - Data Type: string. The name of the AlarmChannel to be created.

1194 addr - Data Type: address. The (host) address to which the reader will send alarms.

1195

1196 **Return Value(s):**

1197 Data Type: AlarmChannel. The newly created object.

1198

1199 **Possible Error Conditions:**

1200 ERROR_COMMAND_NOT_SUPPORTED

1201 ERROR_OBJECT_EXISTS

1202 ERROR_UNKNOWN

1203 ERROR_PARAMETER_MISSING

1204 ERROR_PARAMETER_INVALID_DATATYPE
1205 ERROR_PARAMETER_INVALID_FORMAT
1206 ERROR_PARAMETER_LENGTH_EXCEEDED
1207 ERROR_AUTHORIZATION
1208 ERROR_TOO_MANY_ALARM_CHANNELS
1209

1210 **5.3.2 AlarmChannel.getName**

1211 Returns the name of the given AlarmChannel object.

1212

1213 **Compliance Requirement:** Compliant systems SHALL implement this command if
1214 AlarmChannels are implemented.

1215

1216 **Usage:**

1217 AlarmChannel.getName(void) : string

1218

1219 **Parameter(s):**

1220 Data Type: void. This command SHALL NOT take any parameters.

1221

1222 **Return Value(s):**

1223 Data Type: string. The name of the AlarmChannel object is returned.

1224

1225 **Possible Error Conditions:**

1226 ERROR_COMMAND_NOT_SUPPORTED

1227 ERROR_CHANNEL_NOT_FOUND

1228 ERROR_UNKNOWN

1229

1230 **5.3.3 AlarmChannel.getAddress**

1231 Returns the (host) address to which this AlarmChannel object sends its alarms.

1232

1233 **Compliance Requirement:** Compliant systems SHALL implement this command if
1234 AlarmChannels are implemented.

1235

1236 **Usage:**

1237 AlarmChannel.getAddress(void) : address

1238

1239 **Parameter(s):**

1240 Data Type: void. This command SHALL NOT take any parameters.

1241

1242 **Return Value(s):**

1243 Data Type: address. The reporting address for this AlarmChannel.

1244

1245 **Possible Error Conditions:**

1246 ERROR_CHANNEL_NOT_FOUND

1247 ERROR_COMMAND_NOT_SUPPORTED

1248 ERROR_UNKNOWN

1249

1250 **5.3.4 AlarmChannel.setAddress**

1251 Sets the (host) address to which this AlarmChannel object sends its alarms.

1252

1253 **Compliance Requirement:** Compliant systems SHALL implement this command if
1254 AlarmChannel.create is implemented. Compliant systems MAY implement this
1255 command if AlarmChannel is implemented by means of an alternate method without
1256 implementing NotificationChannel.create..

1257

1258 **Usage:**

1259 AlarmChannel.setAddress(addr: address) : void

1260

1261 **Parameter(s):**

1262 addr - Data Type: address. The reporting address for this AlarmChannel.

1263

1264 **Return Value(s):**

1265 Data Type: void. This command SHALL NOT return any value.

1266

1267 **Possible Error Conditions:**

1268 ERROR_COMMAND_NOT_SUPPORTED

1269 ERROR_CHANNEL_NOT_FOUND

1270 ERROR_UNKNOWN
1271 ERROR_PARAMETER_MISSING
1272 ERROR_PARAMETER_INVALID_DATATYPE
1273 ERROR_PARAMETER_INVALID_FORMAT
1274 ERROR_PARAMETER_LENGTH_EXCEEDED
1275 ERROR_AUTHORIZATION
1276

1277 **5.4 ReadPoint**

1278
1279
1280
1281
1282
1283
1284

1285 **5.4.1 ReadPoint.getClassName**

1286 This method returns the class name of the ReadPoint object. In the current specification,
1287 the only supported read point class is an "AntennaReadPoint", but other types may be
1288 introduced in future versions.

1289

1290 **Compliance Requirement:** Compliant systems SHALL implement this command.

1291 **Compliance Requirement:** Compliant systems SHALL implement this command.

1292

1293 **Usage:**

1294 `ReadPoint.getClassName(void): string`

1295

1296 **Parameter(s):**

1297 Data Type: void. This command takes no parameters.

1298

1299 **Return Value(s):**

1300 Data Type: string. The class name for the ReadPoint. The only supported return value is
1301 "AntennaReadPoint". Note that future versions of the specification may support other
1302 return values.

1303

1304 **Possible Error Conditions:**

1305 ERROR_UNKNOWN

1306 **5.4.2 ReadPoint.getDescription**

1307 Returns a description of this ReadPoint.

1308 The description is set via a call to setDescription and returned via this call. In the event
1309 there has been no previous call to setDescription then the returned string will have 0
1310 length.

1311

1312 **Compliance Requirement:** Compliant systems SHALL implement this command .

1313

1314 **Usage:**

1315 ReadPoint.getDescription (void): string

1316

1317 **Parameter(s):**

1318 Data Type: void. This command takes no parameters.

1319

1320 **Return Value(s):**

1321 Data Type: string. The readpoint descriptive name.

1322

1323 **Possible Error Conditions:**

1324 ERROR_UNKNOWN

1325

1326 **5.4.3 ReadPoint.setDescription**

1327 Set the current ReadPoint description.

1328 This may later be retrieved using ReadPoint.getDescription.

1329

1330 **Compliance Requirement:** Compliant systems SHALL implement this command.

1331 **Usage:**

1332 ReadPoint.setDescription(description: string): void

1333

1334 **Parameter(s):**

1335 description – Data Type: String. A description for this ReadPoint. The
1336 maximum allowable length for this string is vendor-dependent. Attempts to set a string
1337 of excessive length will result in a “length exceeded” error.

1338 **Return Value(s):**

1339 Data Type: void. This command will not return a value.

1340

1341 **Possible Error Conditions:**

1342 ERROR_PARAMETER_MISSING

1343 ERROR_PARAMETER_INVALID_DATATYPE

1344 ERROR_PARAMETER_INVALID_VALUE

1345 ERROR_PARAMETER_LENGTH_EXCEEDED

1346 ERROR_UNKNOWN

1347 ERROR_AUTHORIZATION

1348

1349 **5.4.4 ReadPoint.getAdminStatus**

1350 Returns the current ReadPoint administrative status.

1351 The administrative status represents the host’s *desired* status for this ReadPoint. This
1352 differs from the operational status which represents the *actual* ReadPoint status. The
1353 administrative status is set via a call to setAdminStatus and getAdminStatus simply
1354 returns the most-recently-set value, or the initial vendor-specific administrative state if
1355 setAdminStatus has not yet been called.

1356 **Compliance Requirement:** Compliant systems SHALL implement this command.

1357 **Usage:**

1358 ReadPoint.getAdminStatus (void): AdministrativeStatus

1359

1360 **Parameter(s):**

1361 Data Type: void. This command takes no parameters.

1362

1363 **Return Value(s):**

1364 Data Type: AdministrativeStatus.

1365

1366 **Possible Error Conditions:**

1367 ERROR_UNKNOWN

1368 For a detailed description of these error codes see the Reader Protocol Specification 1.1.

1369

1370 **5.4.5 ReadPoint.setAdminStatus**

1371 Set the current ReadPoint administrative status.

1372 The administrative status represents the host's *desired* status for this ReadPoint. This
1373 differs from the operational status which represents the *actual* ReadPoint status. Hosts
1374 may query the most-recently-requested status using getAdminStatus.

1375

1376 **Compliance Requirement:** Compliant systems SHALL implement this command.

1377 **Usage:**

1378 `ReadPoint.setAdminStatus (status: AdministrativeStatus):`
1379 `void`

1380

1381 **Parameter(s):**

1382 `status` - Data Type: `AdministrativeStatus`. The desired
1383 administrative status of the ReadPoint.

1384

1385 **Return Value(s):**

1386 Data Type: `void`. This command will not return a value.

1387

1388 **Possible Error Conditions:**

1389 `ERROR_PARAMETER_MISSING`

1390 `ERROR_PARAMETER_INVALID_DATATYPE`

1391 `ERROR_PARAMETER_INVALID_VALUE`

1392 `ERROR_UNKNOWN`

1393

1394 **5.4.6 ReadPoint.getOperStatus**

1395 Returns the ReadPoint's current operational status.

1396 The operational status represents the actual status of the ReadPoint. It may be polled
1397 using `getOperStatus` or monitored via alarms (see `GetOperStatusAlarmControl`).

1398

1399 **Compliance Requirement:** Compliant systems SHALL implement this command.

1400 **Usage:**

1401 `ReadPoint.getOperStatus (void): OperationalStatus`

1402

1403 **Parameter(s):**

1404 Data Type: void. This command takes no parameters.

1405

1406 **Return Value(s):**

1407 Data Type: OperationalStatus. The operational status of the
1408 ReadPoint. It may be "unknown", "up", "down" or "other".
1409 See the definition of the OperationalStatus enumeration for
1410 details.

1411

1412 **Possible Error Conditions:**

1413 ERROR_UNKNOWN

1414

1415 **5.4.7 ReadPoint.getOperStatusAlarmControl**

1416 The Host queries the ReadPoint object for its OperStatusAlarmControl
1417 attribute of type TOperationalStatusAlarmControl.. This attribute is the
1418 object that controls the conditions for generating alarms alerting a manager of changes in
1419 a ReadPoint's operational status.

1420 In addition to an alarm, the operational status may also be polled (see getOperStatus).

1421

1422 **Compliance Requirement:** Compliant systems SHALL implement this command if, and
1423 only if, AlarmChannels are implemented.

1424

1425 **Usage:**

1426 ReadPoint.getOperStatusAlarmControl (void) :
1427 TOperationalStatusAlarmControl

1428

1429 **Parameter(s):**

1430 Data Type: void. This command takes no parameters.

1431

1432 **Return Value(s):**

1433 Data Type: TOperationalStatusAlarmControl. An alarm control
1434 for monitoring the operational status of the ReadPoint.

1435

1436 **Possible Error Conditions:**

1437 ERROR_COMMAND_NOT_SUPPORTED

1438 ERROR_UNKNOWN

1439

1440 **5.5 AntennaReadPoint**

1441 AntennaReadPoint extends ReadPoint. It is used to track details about the Radio
1442 Frequency characteristics and RFID statistics for the reader device.

1443 **5.5.1 AntennaReadPoint.getIdentificationCount**

1444 Returns the number of the successful tags that have been identified across an
1445 AntennaReadPoint. In other words, the number of tags whose unique identifier has
1446 been successfully detected by the AntennaReadPoint.

1447 This count is automatically reset to 0 whenever the reader is restarted, and may be
1448 manually reset via a call to ReaderDevice.resetStatistics.

1449 Note that if a single tag's identifier is read multiple times, then the read count will
1450 increment on every successful read. It is the number of times *any* tag's identifier is read,
1451 not the number of *unique* tags which have been read.

1452

1453 **Compliance Requirement:** Compliant systems SHALL implement this command.

1454

1455 **Usage:**

1456 AntennaReadPoint.getIdentificationCount (void): integer

1457

1458 **Parameter(s):**

1459 Data Type: void. This command takes no parameters.

1460

1461 **Return Value(s):**

1462 Data Type: integer. The count of the successful tag identifiers read at this
1463 AntennaReadPoint.

1464

1465 **Possible Error Conditions:**

1466 ERROR_UNKNOWN

1467

1468 **5.5.2 AntennaReadPoint.getFailedIdentificationCount**

1469 Returns the number of the failed tag identification attempts at the
1470 AntennaReadPoint.

1471 This count only includes failures where the reader can unambiguously determine that a
1472 tag was in field that could not be identified. If the reader does not have the means to
1473 determine this, this counter may be left at 0.

1474 This count is automatically reset to 0 whenever the reader is restarted, and may be
1475 manually reset via a call to `ReaderDevice.resetStatistics`.

1476

1477 **Compliance Requirement:** Compliant systems MAY implement this command.

1478

1479 **Usage:**

1480 `AntennaReadPoint.getFailedIdentificationCount (void) :`
1481 `integer`

1482

1483 **Parameter(s):**

1484 Data Type: `void`. This command takes no parameters.

1485

1486 **Return Value(s):**

1487 Data Type: `integer`. The count of the failed attempts at reading the identifier for a
1488 tag at this antenna `ReadPoint`.

1489

1490 **Possible Error Conditions:**

1491 `ERROR_COMMAND_NOT_SUPPORTED`

1492 `ERROR_UNKNOWN`

1493

1494 **5.5.3 AntennaReadPoint.getMemReadCount**

1495 Returns the number of tag memory reads at the `AntennaReadPoint`. This should not
1496 be confused with the number of times the tag has been successfully queried or has
1497 communicated its unique identifier. Such statistics is reported by
1498 `AntennaReadPoint.getIdentificationCount`.

1499 This count is automatically reset to 0 whenever the reader is restarted, and may be
1500 manually reset via a call to `ReaderDevice.resetStatistics`.

1501

1502 **Compliance Requirement:** Compliant systems MAY implement this command.

1503

1504 **Usage:**

1505 `AntennaReadPoint.getMemReadCount (void) : integer`

1506
1507 **Parameter(s):**
1508 Data Type: void. This command takes no parameters.
1509
1510 **Return Value(s):**
1511 Data Type: integer. The count of the successful tag memory reads at this antenna
1512 ReadPoint.
1513
1514 **Possible Error Conditions:**
1515 ERROR_COMMAND_NOT_SUPPORTED
1516 ERROR_UNKNOWN

1517 **5.5.4 AntennaReadPoint.getFailedMemReadCount**

1518 Returns the number of the failed tag memory reads at the AntennaReadPoint.
1519 This count only includes failures where it attempts to read tag memory but does not
1520 complete successfully.
1521 This count is automatically reset to 0 whenever the reader is restarted, and may be
1522 manually reset via a call to ReaderDevice.resetStatistics.

1523
1524 **Compliance Requirement:** Compliant systems MAY implement this command.
1525

1526 **Usage:**

1527 AntennaReadPoint.getFailedMemReadCount (void): integer

1528

1529 **Parameter(s):**

1530 Data Type: void. This command takes no parameters.

1531

1532 **Return Value(s):**

1533 Data Type: integer. The count of the failed tag memory reads at this antenna
1534 readpoint.

1535

1536 **Possible Error Conditions:**

1537 ERROR_COMMAND_NOT_SUPPORTED

1538 ERROR_UNKNOWN

5.5.5 AntennaReadPoint.getFailedMemReadAlarmControl

The Host queries the `AntennaReadPoint` object for its `FailedMemReadAlarmControl` attribute of type `AlarmControl`. This alarm control object controls the generation of alarms triggered by failed read operations of memory banks across the `AntennaRead` point.

In addition to an alarm, the status of failed memory reads may also be polled (see `getMemReadCount`).

Compliance Requirement: Compliant systems MAY implement this command.

Usage:

```
AntennaReadPoint.getFailedMemReadAlarmControl  
(void):AlarmControl
```

Parameter(s):

Data Type: `void`. This command takes no parameters.

Return Value(s):

Data Type: `AlarmControl`. An alarm control for monitoring tag memory read failures.

Possible Error Conditions:

`ERROR_COMMAND_NOT_SUPPORTED`

`ERROR_UNKNOWN`

5.5.6 AntennaReadPoint.getWriteCount

Returns the number of successful tag writes at the `AntennaReadPoint`, including writes to any of the memory banks including the tag identifier.

This count is automatically reset to 0 whenever the reader is restarted, and may be manually reset via a call to `ReaderDevice.resetStatistics`.

Note that if a single tag is written multiple times, then the count will increment on every successful write. It is the number of times *any* tag has been written, not the number of *unique* tags which have been written.

Compliance Requirement: Compliant systems SHALL implement this command.

Usage:

1574 `AntennaReadPoint.getWriteCount (void): integer`
1575
1576 **Parameter(s):**
1577 Data Type: void. This command takes no parameters.
1578
1579 **Return Value(s):**
1580 Data Type: integer. The count of the successful writes at this
1581 `AntennaReadPoint`. If Write functionality is not supported by the Reader it must
1582 return 0 every time.
1583
1584 **Possible Error Conditions:**
1585 `ERROR_UNKNOWN`
1586 For a detailed description of these error codes see the Reader Protocol Specification 1.1.
1587

1588 **5.5.7 AntennaReadPoint.getFailedWriteCount**

1589 Returns the number of the failed attempts to write tags at the `AntennaReadPoint`,
1590 including failed writes to any of the memory banks including the tag identifier.
1591 This is the number of times the reader attempted to write a tag, but the operation failed to
1592 complete successfully.
1593 This count is automatically reset to 0 whenever the reader is restarted, and may be
1594 manually reset via a call to `ReaderDevice.resetStatistics`.

1595
1596 **Compliance Requirement:** Compliant systems MAY implement this command.
1597

1598 **Usage:**

1599 `AntennaReadPoint.getFailedWriteCount (void): integer`
1600

1601 **Parameter(s):**

1602 Data Type: void. This command takes no parameters.
1603

1604 **Return Value(s):**

1605 Data Type: integer. The count of the failed writes at this `AntennaReadPoint`.
1606

1607 **Possible Error Conditions:**

1608 ERROR_COMMAND_NOT_SUPPORTED

1609 ERROR_UNKNOWN

1610

1611 **5.5.8 AntennaReadPoint.getFailedWriteAlarmControl**

1612 The Host queries the AntennaReadPoint object for its
1613 FailedWriteAlarmControl attribute of type AlarmControl. This alarm control
1614 object controls the generation of alarms triggered by failed write operations across the
1615 AntennaRead point, including failed writes to any of the memory banks including the tag
1616 identifier.

1617 In addition to an alarm, the status of writes may also be polled (see getWriteCount and
1618 getFailedWriteCount).

1619

1620 **Compliance Requirement:** Compliant systems MAY implement this command.

1621

1622 **Usage:**

1623 AntennaReadPoint.getFailedWriteAlarmControl
1624 (void):AlarmControl

1625

1626 **Parameter(s):**

1627 Data Type: void. This command takes no parameters.

1628

1629 **Return Value(s):**

1630 Data Type: AlarmControl. An alarm control for monitoring the number of failed
1631 writes.

1632

1633 **Possible Error Conditions:**

1634 ERROR_COMMAND_NOT_SUPPORTED

1635 ERROR_UNKNOWN

1636 For a detailed description of these error codes see the Reader Protocol Specification 1.1.

1637 **5.5.9 AntennaReadPoint.getKillCount**

1638 Returns the number of tags successfully killed at the AntennaReadPoint.

1639 This count is automatically reset to 0 whenever the reader is restarted, and may be
1640 manually reset via a call to ReaderDevice.resetStatistics.

1641

1642 **Compliance Requirement:** Compliant systems SHALL implement this command.
1643
1644 **Usage:**
1645 `AntennaReadPoint.getKillCount (void): integer`
1646
1647 **Parameter(s):**
1648 Data Type: void. This command takes no parameters.
1649
1650 **Return Value(s):**
1651 Data Type: integer. The count of the successful tag kills at this antenna ReadPoint.
1652
1653 **Possible Error Conditions:**
1654 `ERROR_UNKNOWN`
1655

1656 **5.5.10 AntennaReadPoint.getFailedKillCount**

1657 Returns the number of the failed tag kills at the AntennaReadPoint.
1658 This count only includes failures where it attempts to kill a tag but the kill does not
1659 complete successfully.
1660 This count is automatically reset to 0 whenever the reader is restarted, and may be
1661 manually reset via a call to `ReaderDevice.resetStatistics`.

1662
1663 **Compliance Requirement:** Compliant systems MAY implement this command.

1664
1665 **Usage:**
1666 `AntennaReadPoint.getFailedKillCount (void): integer`
1667

1668 **Parameter(s):**
1669 Data Type: void. This command takes no parameters.
1670

1671 **Return Value(s):**
1672 Data Type: integer. The count of the failed tag kills at this antenna readpoint.
1673

1674 **Possible Error Conditions:**

1675 ERROR_COMMAND_NOT_SUPPORTED

1676 ERROR_UNKNOWN

1677 **5.5.11 AntennaReadPoint.getFailedKillAlarmControl**

1678 Return the AntennaReadPoint's failed kill alarm control.

1679 In addition to an alarm, the status of kills may also be polled (see getKillCount and
1680 getFailedKillCount).

1681

1682 **Compliance Requirement:** Compliant systems MAY implement this command.

1683

1684 **Usage:**

1685 AntennaReadPoint.getFailedKillAlarmControl

1686 (void):AlarmControl

1687

1688 **Parameter(s):**

1689 Data Type: void. This command takes no parameters.

1690

1691 **Return Value(s):**

1692 Data Type: AlarmControl. An alarm control for monitoring tag kill failures.

1693

1694 **Possible Error Conditions:**

1695 ERROR_COMMAND_NOT_SUPPORTED

1696 ERROR_UNKNOWN

1697 **5.5.12 AntennaReadPoint.getEraseCount**

1698 Returns the number of tags successfully erased at the AntennaReadPoint.

1699 This count is automatically reset to 0 whenever the reader is restarted, and may be
1700 manually reset via a call to ReaderDevice.resetStatistics.

1701

1702 **Compliance Requirement:** Compliant systems SHALL implement this command.

1703

1704 **Usage:**

1705 AntennaReadPoint.getEraseCount (void): integer

1706

1707 **Parameter(s):**

1708 Data Type: void. This command takes no parameters.

1709

1710 **Return Value(s):**

1711 Data Type: integer. The count of the successful tag erasures at this antenna
1712 ReadPoint.

1713

1714 **Possible Error Conditions:**

1715 ERROR_UNKNOWN

1716

1717 **5.5.13 AntennaReadPoint.getFailedEraseCount**

1718 Returns the number of the failed tag erasures at the AntennaReadPoint.

1719 This count only includes failures where it attempts to erase a tag but the erasure does not
1720 complete successfully.

1721 This count is automatically reset to 0 whenever the reader is restarted, and may be
1722 manually reset via a call to ReaderDevice.resetStatistics.

1723

1724 **Compliance Requirement:** Compliant systems MAY implement this command.

1725

1726 **Usage:**

1727 AntennaReadPoint.getFailedEraseCount (void): integer

1728

1729 **Parameter(s):**

1730 Data Type: void. This command takes no parameters.

1731

1732 **Return Value(s):**

1733 Data Type: integer. The count of the failed tag erasures at this antenna readpoint.

1734

1735 **Possible Error Conditions:**

1736 ERROR_COMMAND_NOT_SUPPORTED

1737 ERROR_UNKNOWN

1738 **5.5.14 AntennaReadPoint.getFailedEraseAlarmControl**

1739 Return the AntennaReadPoint's failed erase alarm control.

1740 In addition to an alarm, the status of erasures may also be polled (see `getEraseCount` and
1741 `getFailedEraseCount`).

1742

1743 **Compliance Requirement:** Compliant systems MAY implement this command.

1744

1745 **Usage:**

1746 `AntennaReadPoint.getFailedEraseAlarmControl`
1747 `(void):AlarmControl`

1748

1749 **Parameter(s):**

1750 Data Type: `void`. This command takes no parameters.

1751

1752 **Return Value(s):**

1753 Data Type: `AlarmControl`. An alarm control for monitoring tag erasure failures.

1754

1755 **Possible Error Conditions:**

1756 `ERROR_COMMAND_NOT_SUPPORTED`

1757 `ERROR_UNKNOWN`

1758 **5.5.15 AntennaReadPoint.getLockCount**

1759 Returns the number of tags successfully locked at the `AntennaReadPoint`.

1760 This count is automatically reset to 0 whenever the reader is restarted, and may be
1761 manually reset via a call to `ReaderDevice.resetStatistics`.

1762

1763 **Compliance Requirement:** Compliant systems MAY implement this command.

1764

1765 **Usage:**

1766 `AntennaReadPoint.getLockCount (void): integer`

1767

1768 **Parameter(s):**

1769 Data Type: `void`. This command takes no parameters.

1770

1771 **Return Value(s):**

1772 Data Type: `integer`. The count of tags successfully locked at this antenna `ReadPoint`.

1773

1774 **Possible Error Conditions:**

1775 ERROR_COMMAND_NOT_SUPPORTED

1776 ERROR_UNKNOWN

1777 **5.5.16 AntennaReadPoint.getFailedLockCount**

1778 Returns the number of the failed tag locks at the AntennaReadPoint.

1779 This count only includes failures where it attempts to lock a tag but the lock does not
1780 complete successfully.

1781 This count is automatically reset to 0 whenever the reader is restarted, and may be
1782 manually reset via a call to ReaderDevice.resetStatistics.

1783

1784 **Compliance Requirement:** Compliant systems MAY implement this command.

1785

1786 **Usage:**

1787 AntennaReadPoint.getFailedLockCount (void): integer

1788

1789 **Parameter(s):**

1790 Data Type: void. This command takes no parameters.

1791

1792 **Return Value(s):**

1793 Data Type: integer. The count of the failed tag locks at this antenna readpoint.

1794

1795 **Possible Error Conditions:**

1796 ERROR_COMMAND_NOT_SUPPORTED

1797 ERROR_UNKNOWN

1798 AntennaReadPoint.getFailedLockAlarmControl

1799

1800

1801

1802

1803

1804

1805

1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823

1824 **5.5.16.1**

1825 Return the AntennaReadPoint's failed lock alarm control.

1826 In addition to an alarm, the status of locks may also be polled (see getLockCount and
1827 getFailedLockCount).

1828

1829 **Compliance Requirement:** Compliant systems MAY implement this command.

1830

1831 **Usage:**

1832 `AntennaReadPoint.getFailedLockAlarmControl`
1833 `(void):AlarmControl`

1834

1835 **Parameter(s):**

1836 Data Type: void. This command takes no parameters.

1837

1838 **Return Value(s):**

1839 Data Type: `AlarmControl`. An alarm control for monitoring tag lock failures.

1840

1841 **Possible Error Conditions:**

1842 `ERROR_COMMAND_NOT_SUPPORTED`

1843 `ERROR_UNKNOWN`

1844 **5.5.17 AntennaReadPoint.getTimeEnergized**

1845 Returns the number of milliseconds the `AntennaReadPoint` has been
1846 energized in order to communicate with tags.

1847 **Compliance Requirement:** Compliant systems MAY implement this command.

1848

1849 **Usage:**

1850 `AntennaReadPoint.getTimeEnergized(void) : integer`

1851

1852 **Parameter(s):**

1853 Data Type: `void`. This command takes no parameters.

1854

1855 **Return Value(s):**

1856 Data Type: `integer`. The number of milliseconds the `AntennaReadPoint` has
1857 been energized attempting communication with tags. This count is
1858 automatically reset to 0 whenever the reader is restarted, and may be manually reset via a
1859 call to `ReaderDevice.resetStatistics`.

1860

1861 **Possible Error Conditions:**

1862 `ERROR_COMMAND_NOT_SUPPORTED`

1863 `ERROR_UNKNOWN`

1864 `AntennaReadPoint.getPowerLevel`

1865

1866

1867

1868

1869

1870

1871

1872

1873 **5.5.17.1**
1874 Returns the current transmit power level of the `AntennaReadPoint`. The units of
1875 measurement are vendor specific.
1876
1877 **Compliance Requirement:** Compliant systems MAY implement this command.
1878
1879 **Usage:**
1880 `AntennaReadPoint.getPowerLevel(void) : integer`
1881
1882 **Parameter(s):**
1883 Data Type: `void`. This command takes no parameters.
1884
1885 **Return Value(s):**
1886 Data Type: `integer`. The current transmit power level of the `AntennaReadPoint`.
1887 The meaning of this value is device-dependent and users should consult documentation
1888 provided by the reader manufacturer.
1889
1890 **Possible Error Conditions:**
1891 `ERROR_COMMAND_NOT_SUPPORTED`
1892 `ERROR_UNKNOWN`
1893
1894 **5.5.18 `AntennaReadPoint.getNoiseLevel`**
1895 The Host sends a `getNoiseLevel` message to query the Reader for the current noise
1896 level observed at the `AntennaReadPoint`. The units of measurement are vendor
1897 specific.
1898
1899 **Compliance Requirement:** Compliant systems MAY implement this command.
1900
1901 **Usage:**
1902 `AntennaReadPoint.getNoiseLevel(void) : integer`
1903
1904 **Parameter(s):**
1905 Data Type: `void`. This command takes no parameters.
1906

1907 **Return Value(s):**
1908 Data Type: `integer`. The current noise level observed at the `AntennaReadPoint`.
1909 The meaning of this value is device-dependent and users should consult the
1910 documentation provided by the reader manufacturer for the meaning. The only constraint
1911 is that larger numbers imply higher levels of noise.

1912

1913 **Possible Error Conditions:**

1914 `ERROR_COMMAND_NOT_SUPPORTED`

1915 `ERROR_UNKNOWN`

1916

1917 **5.6 Source Object**

1918 **5.6.1 Source.getUnknownToGlimpsedCount**

1919 The Host queries the Reader for the number of times a transition from state Unknown to
1920 state Glimpsed have been detected for the particular source.

1921

1922 **Compliance Requirement:** Compliant systems MAY implement this command.

1923

1924 **Usage:**

1925 `Source.getUnknownToGlimpsedCount (void): integer`

1926

1927 **Parameter(s):**

1928 Data Type: `void`. This command takes no parameters.

1929

1930 **Return Value(s):**

1931 Data Type: `integer`. The number times the particular transition has been detected.

1932

1933 **Possible Error Conditions:**

1934 `ERROR_SOURCE_NOT_FOUND`

1935 `ERROR_COMMAND_NOT_SUPPORTED`

1936

1937 **5.6.2 Source.getGlimpsedToUnknownCount**

1938 The Host queries the Reader for the number of times a transition from state Glimpsed to
1939 state Unknown have been detected for the particular source.

1940
1941 **Compliance Requirement:** Compliant systems MAY implement this command.
1942
1943 **Usage:**
1944 `Source.getGlimpsedToUnknownCount (void) : integer`
1945
1946 **Parameter(s):**
1947 Data Type: `void`. This command takes no parameters.
1948
1949 **Return Value(s):**
1950 Data Type: `integer`. The number times the particular transition has been detected.
1951
1952 **Possible Error Conditions:**
1953 `ERROR_SOURCE_NOT_FOUND`
1954 `ERROR_COMMAND_NOT_SUPPORTED`
1955

1956 **5.6.3 Source.getGlimpsedToObservedCount**

1957 The Host queries the Reader for the number of times a transition from state Glimpsed to
1958 state Observed have been detected for the particular source.

1959
1960 **Compliance Requirement:** Compliant systems MAY implement this command.
1961

1962 **Usage:**
1963 `Source.getGlimpsedToObservedCount (void) : integer`
1964

1965 **Parameter(s):**
1966 Data Type: `void`. This command takes no parameters.
1967

1968 **Return Value(s):**
1969 Data Type: `integer`. The number times the particular transition has been detected.
1970

1971 **Possible Error Conditions:**
1972 `ERROR_SOURCE_NOT_FOUND`

1973 ERROR_COMMAND_NOT_SUPPORTED

1974

1975 **5.6.4 Source.getObservedToLostCount**

1976 The Host queries the Reader for the number of times a transition from state Observed to
1977 state Lost have been detected for the particular source.

1978

1979 **Compliance Requirement:** Compliant systems MAY implement this command.

1980

1981 **Usage:**

1982 `Source.getObservedToLostCount (void): integer`

1983

1984 **Parameter(s):**

1985 Data Type: `void`. This command takes no parameters.

1986

1987 **Return Value(s):**

1988 Data Type: `integer`. The number times the particular transition has been detected.

1989

1990 **Possible Error Conditions:**

1991 ERROR_SOURCE_NOT_FOUND

1992 ERROR_COMMAND_NOT_SUPPORTED

1993

1994 **5.6.5 Source.getLostToGlimpsedCount**

1995 The Host queries the Reader for the number of times a transition from state Lost to state
1996 Observed have been detected for the particular source.

1997

1998 **Compliance Requirement:** Compliant systems MAY implement this command.

1999

2000 **Usage:**

2001 `Source.getLostToGlimpsedCount (void): integer`

2002

2003 **Parameter(s):**

2004 Data Type: `void`. This command takes no parameters.

2005
2006 **Return Value(s):**
2007 Data Type: `integer`. The number times the particular transition has been detected.
2008

2009 **Possible Error Conditions:**
2010 `ERROR_SOURCE_NOT_FOUND`
2011 `ERROR_COMMAND_NOT_SUPPORTED`
2012

2013 **5.6.6 Source.getLostToUnknownCount**

2014 The Host queries the Reader for the number of times a transition from state Lost to state
2015 Unknown have been detected for the particular source.

2016
2017 **Compliance Requirement:** Compliant systems MAY implement this command.
2018

2019 **Usage:**
2020 `Source.getLostToUnknownCount (void): integer`
2021

2022 **Parameter(s):**
2023 Data Type: `void`. This command takes no parameters.
2024

2025 **Return Value(s):**
2026 Data Type: `integer`. The number times the particular transition has been detected.
2027

2028 **Possible Error Conditions:**
2029 `ERROR_SOURCE_NOT_FOUND`
2030 `ERROR_COMMAND_NOT_SUPPORTED`
2031

2032 **5.6.7 Source.getOperStatus**

2033 The Host queries the Reader for the operational status of this particular Source. The
2034 operational status reported is one of the values in the `OperationalStatus` enumeration, e.g.
2035 UP, DOWN, UNKNOWN or OTHER. The operational status represents the actual status
2036 of the Source. It may be polled using `getOperStatus` or monitored via alarms (see
2037 `GetOperStatusAlarmControl`).

2038
2039
2040 **Compliance Requirement:** Compliant systems SHALL implement this command.
2041 **Usage:**
2042 `Source.getOperStatus (void) : OperationalStatus`
2043
2044 **Parameter(s):**
2045 Data Type: `void`. This command takes no parameters.
2046
2047 **Return Value(s):**
2048 Data Type: `OperationalStatus`. Reader will return the current operational status of
2049 this Source.
2050
2051 **Possible Error Conditions:**
2052 `ERROR_COMMAND_NOT_SUPPORTED`
2053 `ERROR_SOURCE_NOT_FOUND`
2054

2055 **5.6.8 Source.getAdminStatus**

2056 The Host queries the Reader for the administrative status of a particular Source. The
2057 administrative status reported is one of the values in the `AdministrativeStatus`
2058 enumeration, e.g. UP or DOWN.

2059 The administrative status represents the host's *desired* status for this Source. This differs
2060 from the operational status which represents the *actual* Source status. The administrative
2061 status is set via a call to `setAdminStatus` and `getAdminStatus` simply returns the most-
2062 recently-set value.

2063 **Compliance Requirement:** Compliant systems SHALL implement this command.

2064 **Usage:**

2065 `Source.getAdminStatus (void) : AdministrativeStatus`

2066

2067 **Parameter(s):**

2068 Data Type: `void`. This command takes no parameters.

2069

2070 **Return Value(s):**

2071 Data Type: `AdministrativeStatus`. Reader will return the current administrative
2072 status of this Source.

2073

2074 **Possible Error Conditions:**

2075 ERROR_SOURCE_NOT_FOUND

2076 ERROR_COMMAND_NOT_SUPPORTED

2077

2078 **5.6.9 Source.setAdminStatus**

2079 This operation is used to set the administrative status of a particular Source. The

2080 administrative status assigned is one of the values in the AdministrativeStatus

2081 enumeration, e.g. UP or DOWN

2082 The administrative status represents the host's *desired* status for this Source. This differs
2083 from the operational status which represents the *actual* status. Hosts may query the most-
2084 recently-requested status using getAdminStatus.

2085

2086 **Compliance Requirement:** Compliant systems SHALL implement this command.

2087

2088 **Usage:**

2089 Source.setAdminStatus (administrativeStatus

2090 :AdministrativeStatus): void

2091

2092 **Parameter(s):**

2093 administrativeStatus – Data Type: AdministrativeStatus. This is the
2094 new administrative status to be set.

2095

2096 **Return Value(s):**

2097 Data Type: void. This command will not return a value.

2098

2099 **Possible Error Conditions:**

2100 ERROR_COMMAND_NOT_SUPPORTED

2101 ERROR_INVALID_ADMIN_STATUS

2102

2103 **5.6.10 Source.getOperStatusAlarmControl**

2104 The Host queries the Source object for its OperStatusAlarmControl attribute of

2105 type TTOperationalStatusAlarmControl. This attribute is the object that

2106 controls the conditions for generating alarms alerting a manager of changes in a

2107 Source's operational status.

2108 In addition to an alarm, the operational status may also be polled (see `getOperStatus`).

2109

2110 **Compliance Requirement:** Compliant systems SHALL implement this command if
2111 `AlarmChannel` is implemented.

2112

2113 **Usage:**

2114 `Source.getOperStatusAlarm (void) :`

2115 `TTOperationalStatusAlarmControl`

2116

2117 **Parameter(s):**

2118 Data Type: `void`. This command takes no parameters.

2119

2120 **Return Value(s):**

2121 Data Type: `TTOperationalStatusAlarmControl`. An alarm control for
2122 monitoring the operational status of the Source.

2123

2124 **Possible Error Conditions:**

2125 `ERROR_COMMAND_NOT_FOUND`

2126 `ERROR_SOURCE_NOT_FOUND`

2127

2128 **5.7 Trigger Object**

2129 **5.7.1 Trigger.getFireCount**

2130 The Host queries the Reader for the number of times a particular trigger has fired.

2131

2132 **Compliance Requirement:** Compliant systems MAY implement this command.

2133

2134 **Usage:**

2135 `Trigger.getFireCount (void) : integer`

2136

2137 **Parameter(s):**

2138 Data Type: `void`. This command takes no parameters.

2139

2140 **Return Value(s):**

2141 Data Type: `integer`. Reader will return the number of times the trigger has fired.

2142

2143 **Possible Error Conditions:**

2144 `ERROR_TRIGGER_NOT_FOUND`

2145 `ERROR_COMMAND_NOT_SUPPORTED`

2146

2147 **5.8 IOPort Object**

2148 **5.8.1 IOPort.getName**

2149 The Host queries the Reader for the name of a particular IO-port. Since the IO-port
2150 names are set by the manufacturer, there is no `setName()` command.

2151

2152 **Compliance Requirement:** Compliant systems SHALL implement this command if
2153 IOPorts are exposed.

2154

2155 **Usage:**

2156 `IOPort.getName(void): string`

2157

2158 **Parameter(s):**

2159 Data Type: `void`. This command takes no parameters.

2160

2161 **Return Value(s):**

2162 Data Type: `string`. Reader will return the name of this IO-port.

2163

2164 **Possible Error Conditions:**

2165 `ERROR_IOPORT_NOT_FOUND`

2166 `ERROR_COMMAND_NOT_SUPPORTED`

2167

2168 **5.8.2 IOPort.getDescription**

2169 This operation provides a textual description of the IO-port. This is typically used to
2170 denote the equipment connected via this IO-port.

2171 **Compliance Requirement:** Compliant systems SHALL implement this command if
2172 IOPorts are exposed.

2173

2174 **Usage:**
2175 `IOPort.getDescription (void): string`
2176
2177 **Parameter(s):**
2178 Data Type: `void`. This command takes no parameters.
2179
2180 **Return Value(s):**
2181 Data Type: `string`. Reader will return the description of the IO-port in a free form
2182 String. If no description has been set the operation returns a String of length zero (an
2183 empty string).
2184
2185 **Possible Error Conditions:**
2186 `ERROR_IOPORT_NOT_FOUND`
2187 `ERROR_COMMAND_NOT_SUPPORTED`
2188

2189 **5.8.3 IOPort.setDescription**

2190 This operation is used to associate a textual description with an IO-port. This is typically
2191 used to denote the equipment connected via this IO-port.

2192 **Compliance Requirement:** Compliant systems SHALL implement this command if
2193 IOPorts are exposed.

2194
2195 **Usage:**
2196 `IOPort.setDescription (description : string): void`
2197

2198 **Parameter(s):**
2199 `description` – Data Type: `string`. The textual description.
2200

2201 **Return Value(s):**
2202 Data Type: `void`. This operation does not provide a return value.
2203

2204 **Possible Error Conditions:**
2205 `ERROR_IOPORT_NOT_FOUND`
2206 `ERROR_PARAMETER_LENGTH_EXCEEDED`
2207 `ERROR_COMMAND_NOT_SUPPORTED`

2208

2209 **5.8.4 IOPort.getOperStatus**

2210 The Host queries the Reader for the operational status of this particular IO-port. The
2211 operational status reported is one of the values in the OperationalStatus enumeration, e.g.
2212 UP, DOWN, UNKNOWN or OTHER. The operational status represents the actual status
2213 of the IOPort. It may be polled using getOperStatus or monitored via alarms (see
2214 getOperStatusAlarmControl).

2215

2216 **Compliance Requirement:** Compliant systems SHALL implement this command if
2217 IOPorts are exposed.

2218

2219 **Usage:**

2220 `IOPort.getOperStatus (void) : OperationalStatus`

2221

2222 **Parameter(s):**

2223 Data Type: `void`. This command takes no parameters.

2224

2225 **Return Value(s):**

2226 Data Type: `OperationalStatus`. Reader will return the current operational status of
2227 this IO-port.

2228

2229 **Possible Error Conditions:**

2230 `ERROR_IOPORT_NOT_FOUND`

2231 `ERROR_COMMAND_NOT_SUPPORTED`

2232

2233 **5.8.5 IOPort.getAdminStatus**

2234 The Host queries the Reader for the administrative status of a particular IO-port. The
2235 administrative status reported is one of the values in the AdministrativeStatus
2236 enumeration, e.g. UP or DOWN.

2237 The administrative status represents the host's *desired* status for this IOPort. This differs
2238 from the operational status which represents the *actual* IOPort status. The administrative
2239 status is set via a call to `setAdminStaus` and `getAdminStatus` simply returns the most-
2240 recently-set value.

2241

2242 **Compliance Requirement:** Compliant systems SHALL implement this command if
2243 IOPorts.

2244

2245 **Usage:**

2246 `IOPort.getAdminStatus (void) : AdministrativeStatus`

2247

2248 **Parameter(s):**

2249 Data Type: `void`. This command takes no parameters.

2250

2251 **Return Value(s):**

2252 Data Type: `AdministrativeStatus`. Reader will return the current administrative

2253 status of this IO-port.

2254

2255 **Possible Error Conditions:**

2256 `ERROR_IOPORT_NOT_FOUND`

2257 `ERROR_COMMAND_NOT_SUPPORTED`

2258

2259 **5.8.6 IOPort.setAdminStatus**

2260 This operation is used to set the administrative status of a particular IO-port. The

2261 administrative status assigned is one of the values in the `AdministrativeStatus`

2262 enumeration, e.g. UP or DOWN

2263 The administrative status represents the host's *desired* status for this IOPort. This differs

2264 from the operational status which represents the *actual* IOPort status. Hosts may query

2265 the most-recently-requested status using `getAdminStatus`.

2266

2267 **Compliance Requirement:** Compliant systems SHALL implement this command if

2268 IOPorts are exposed.

2269

2270 **Usage:**

2271 `IOPort.setAdminStatus (administrativeStatus :`

2272 `AdministrativeStatus) : void`

2273

2274 **Parameter(s):**

2275 `administrativeStatus` – Data Type: `AdministrativeStatus`.. This is the

2276 new administrative status to be set.

2277

2278 **Return Value(s):**

2279 Data Type: `void`. This operation does not provide a return value.

2280

2281 **Possible Error Conditions:**

2282 `ERROR_COMMAND_NOT_SUPPORTED`

2283 `ERROR_IOPORT_NOT_FOUND`

2284 `ERROR_INVALID_ADMIN_STATUS`

2285

2286 **5.8.7 IOPort.getOperStatusAlarmControl**

2287 The Host queries the `IOPort` object for its `OperStatusAlarmControl` attribute of
2288 type `TTOperationalStatusAlarmControl`. This attribute is the object that
2289 controls the conditions for generating alarms alerting a manager of changes in an
2290 `IOPort`'s operational status.

2291 In addition to an alarm, the operational status may also be polled (see `getOperStatus`).

2292

2293 **Compliance Requirement:** Compliant systems SHALL implement this command if
2294 `AlarmChannel` is implemented..

2295

2296 **Usage:**

2297 `IOPort.getOperStatusAlarmControl (void) :`

2298 `TTOperationalStatusAlarmControl`

2299

2300 **Parameter(s):**

2301 Data Type: `void`. This command takes no parameters.

2302

2303 **Return Value(s):**

2304 Data Type: `TTOperationalStatusAlarmControl`. An alarm control for
2305 monitoring the operational status of the `IOPort`.

2306

2307 **Possible Error Conditions:**

2308 `ERROR_COMMAND_NOT_SUPPORTED`

2309 `ERROR_IOPORT_NOT_FOUND`

2310

2311 `AlarmControl` Object `AlarmControl` is the base class for all classes responsible for
2312 controlling the generation of `Alarm` messages, and consists of four main attributes.

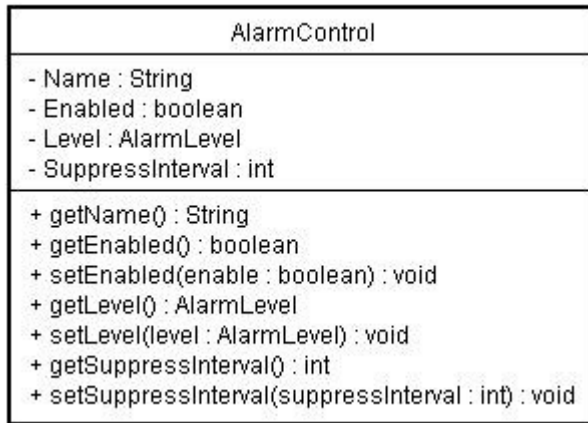


Figure 15 AlarmControl UML

The Name attribute (of type String) is a unique identifier of an AlarmControl object.

The Enabled attribute (of type Boolean) controls whether the Alarm is enabled. If set to false, Alarm generation will be inhibited for the specific Alarm this object controls.

The Level attribute (of type AlarmLevel) specifies the Alarm level (defined by the AlarmLevel enumeration) assigned to alarms whose generation this object controls.

The SuppressInterval attribute (of type integer) specifies the minimum number of seconds that SHALL elapse before the next Alarm of the same type is generated. Each object can only generate one of these Alarms per SuppressInterval. Setting the SuppressInterval to 0 results in an Alarm generated for every Alarm condition encountered. Setting SuppressInterval to a value greater than zero reduces the number of Alarms raised, minimizing the network traffic and load on the health monitoring applications. The ideal value will depend on the monitored environment and how frequently the health monitoring application needs to be interrupted to process Alarm conditions. Every Alarm contains a SuppressCount indicating the number of times the Alarm generation has been suppressed during SuppressInterval. This count is reset to 0 after the successful generation of the Alarm.

For example, consider two AntennaReadPoint instances named AntennaReadPoint1 and AntennaReadPoint2. The FailedKillAlarmControl and FailedWriteAlarmControl have both been enabled with SuppressInterval set to 15 seconds. The following table lists the action taken when a number of Alarm conditions are encountered at different times.

Time	Object	Alarm Condition	Action
09:00:00	AntennaReadPoint1	Kill Failure	Generate Alarm
09:00:01	AntennaReadPoint2	Kill Failure	Generate Alarm
09:00:05	AntennaReadPoint1	Kill Failure	Suppress Alarm
09:00:06	AntennaReadPoint1	Write Failure	Generate Alarm
09:00:08	AntennaReadPoint1	Kill Failure	Suppress Alarm
09:00:16	AntennaReadPoint1	Kill Failure	Generate Alarm

2342

2343

2344 Defined subclasses of `AlarmControl` are:

- 2345 • `EdgeTriggeredAlarmControl`
- 2346 •
- 2347 • `TTOperationalStatusAlarmControl`

2348 Future revisions of this specification may define new `AlarmControls` in addition to
 2349 `EdgeTriggeredAlarmControl` and `TTOperationStatusAlarmControl`

2350

2351 **5.8.8 AlarmControl.getName**

2352 Queries the reader for the current value of the `AlarmControl`'s `Name` attribute.

2353

2354 **Compliance Requirement:** Compliant systems SHALL implement this command if
 2355 `AlarmChannel` is implemented.

2356

2357 **Usage:**

2358 `AlarmControl.getName (void): string`

2359

2360 **Parameter(s):**

2361 Data Type: `void`. This command takes no parameters.

2362

2363 **Return Value(s):**

2364 Data Type: `string`. The `AlarmControl`'s unique name.

2365

2366 **Possible Error Conditions:**

2367 None.

2368

2369 **5.8.9 AlarmControl.setEnabled**

2370 Queries the reader for the current value of the AlarmControl's Enabled attribute.

2371

2372 **Compliance Requirement:** Compliant systems SHALL implement this command if
2373 AlarmChannel is implemented.

2374

2375 **Usage:**

2376 AlarmControl.setEnabled (void): boolean

2377

2378 **Parameter(s):**

2379 Data Type: void. This command takes no parameters.

2380

2381 **Return Value(s):**

2382 Data Type: boolean. Indicates whether the AlarmControl is enabled.

2383

2384 **Possible Error Conditions:**

2385 None.

2386

2387 **5.8.10 AlarmControl.setEnabled**

2388 Enables or disables alarm generation.

2389

2390 **Compliance Requirement:** Compliant systems SHALL implement this command if
2391 AlarmChannel is implemented.

2392

2393 **Usage:**

2394 AlarmControl.setEnabled (enable: boolean): void

2395

2396 **Parameter(s):**

2397 enable – Data Type: boolean. A boolean that enables or disables
2398 the AlarmControl.

2399

2400 **Return Value(s):**
2401 Data Type: void.
2402
2403 **Possible Error Conditions:**
2404 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2405 alarm controls.
2406

2407 **5.8.11 AlarmControl.getLevel**

2408 Queries the reader for the current value of the AlarmControl's Level attribute.
2409 **Compliance Requirement:** Compliant systems SHALL implement this command if
2410 AlarmChannel is implemented.

2411

2412 **Usage:**

2413 AlarmControl.getLevel (void): AlarmLevel

2414

2415 **Parameter(s):**

2416 Data Type: void. This command takes no parameters.

2417

2418 **Return Value(s):**

2419 Data Type: AlarmLevel.

2420

2421 **Possible Error Conditions:**

2422 None.

2423

2424 **5.8.12 AlarmControl.setLevel**

2425 Sets the Level attribute.

2426 **Compliance Requirement:** Compliant systems SHALL implement this command if
2427 AlarmChannel is implemented.

2428

2429 **Usage:**

2430 AlarmControl.setLevel (alarmLevel: AlarmLevel): void

2431

2432 **Parameter(s):**

2433 alarmLevel – Data Type: AlarmLevel. The desired Alarm Level value.

2434

2435 **Return Value(s):**

2436 Data Type: void.

2437

2438 **Possible Error Conditions:**

2439 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2440 alarm controls.

2441

2442 **5.8.13 AlarmControl.getSuppressInterval**

2443 Queries the reader for the current value of the AlarmControl's SuppressInterval
2444 attribute.

2445 **Compliance Requirement:** Compliant systems SHALL implement this command if
2446 AlarmChannel is implemented.

2447

2448 **Usage:**

2449 AlarmControl.getSuppressInterval (void): integer

2450

2451 **Parameter(s):**

2452 Data Type: void. This command takes no parameters.

2453

2454 **Return Value(s):**

2455 Data Type: integer. The value of the AlarmControl
2456 SuppressInterval value in msec.

2457

2458 **Possible Error Conditions:**

2459 None.

2460

2461 **5.8.14 AlarmControl.setSuppressInterval**

2462 Sets the SuppressInterval attribute.

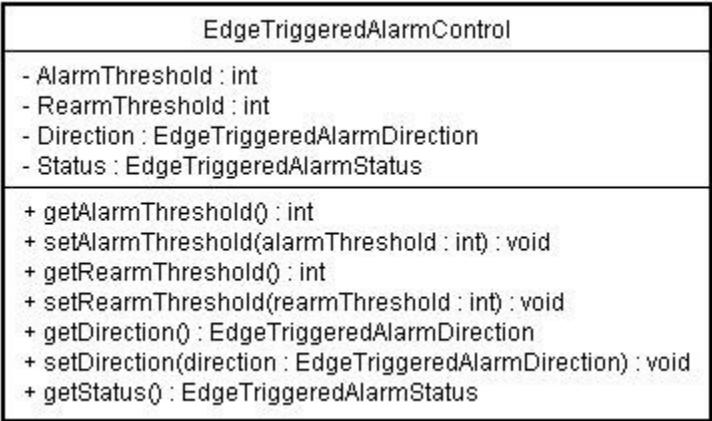
2463 **Compliance Requirement:** Compliant systems SHALL implement this command if
2464 AlarmChannel is implemented.

2465

2466 **Usage:**
2467 AlarmControl.setSuppressInterval (suppressInterval:
2468 integer): void
2469
2470 **Parameter(s):**
2471 suppressInterval – Data Type: integer. The desired Alarm
2472 SuppressInterval value in msec.
2473
2474 **Return Value(s):**
2475 Data Type: void: This command will not return a value.
2476
2477 **Possible Error Conditions:**
2478 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2479 alarm controls.
2480
2481

2482 **5.9 EdgeTriggeredAlarmControl**

2483 This class extends AlarmControl to control alarms generated when a monitored,
2484 integer-valued, model element first crosses a threshold value (the AlarmThreshold).
2485 This type of alarm is called “edge triggered”. The monitored value must cross a
2486 potentially different threshold (the RearmThreshold) before it can be triggered again.

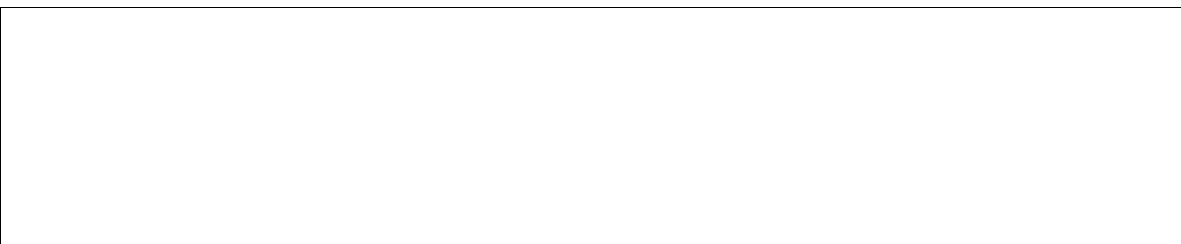


2487
2488 **Figure 16 EdgeTriggeredAlarmControl UML**

2489
2490
2491

2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515

2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528



The `Direction` attribute (of enumerated type `EdgeTriggeredAlarmDirection`) indicates whether the alarm will be triggered on an `EdgeTriggeredAlarmDirection.RISING` or `EdgeTriggeredAlarmDirection.FALLING` passage through the `AlarmThreshold`. The `Status` attribute (of enumerated type `EdgeTriggeredAlarmStatus`) identifies the current state of the alarm, i.e., `EdgeTriggeredAlarmStatus.ARMED` or `EdgeTriggeredAlarmStatus.FIRED`.

If alarm control's direction is `RISING`, an alarm is triggered when the monitored value first becomes greater than the `AlarmThreshold`; once triggered, the alarm enters the `SUSPENDED` state, remaining in that state until the monitored value moves below (less than) the `RearmThreshold`, at which time the alarm re-enters the `ARMED` state.

If the alarm control's direction is `FALLING`, an alarm is triggered when the monitored value first becomes less than the `AlarmThreshold`; once triggered, the alarm enters the `SUSPENDED` state, remaining in that state until the monitored value moves above (greater than) the `RearmThreshold`, at which time the alarm re-enters the `ARMED` state.

5.9.1 `EdgeTriggeredAlarmControl.getAlarmThreshold`

Queries the reader for the current value of the `EdgeTriggeredAlarmControl`'s `AlarmThreshold` attribute.

Compliance Requirement: Compliant systems SHALL implement this command.

Usage:

`EdgeTriggeredAlarmControl.getAlarmThreshold (void) : integer`

Parameter(s):

Data Type: `void`. This command takes no parameters.

Return Value(s):

2529 Data Type: integer.

2530

2531 **Possible Error Conditions:**

2532 None.

2533 **5.9.2 EdgeTriggeredAlarmControl.setAlarmThreshold**

2534 Command sent to the reader to set the current value of the
2535 EdgeTriggeredAlarmControl's AlarmThreshold attribute.

2536

2537 **Compliance Requirement:** Compliant systems SHALL implement this command.

2538

2539 **Usage:**

2540 EdgeTriggeredAlarmControl.setAlarmThreshold (alarmThreshold:
2541 integer): void

2542

2543 **Parameter(s):**

2544 alarmThreshold – Data Type: integer. The desired value of AlarmThreshold.

2545

2546 **Return Value(s):**

2547 Data Type: void.

2548

2549 **Possible Error Conditions:**

2550 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2551 alarm controls.

2552

2553 **5.9.3 EdgeTriggeredAlarmControl.getRearmThreshold**

2554 Queries the reader for the current value of the EdgeTriggeredAlarmControl's
2555 RearmThreshold attribute.

2556

2557 **Compliance Requirement:** Compliant systems SHALL implement this command.

2558

2559 **Usage:**

2560 EdgeTriggeredAlarmControl.getRearmThreshold (void): integer

2561

2562 **Parameter(s):**

2563 Data Type: void. This command takes no parameters.

2564

2565 **Return Value(s):**

2566 Data Type: integer.

2567

2568 **Possible Error Conditions:**

2569 None.

2570 **5.9.4 EdgeTriggeredAlarmControl.setRearmThreshold**

2571 Command sent to the reader to set the current value of the
2572 EdgeTriggeredAlarmControl's RearmThreshold attribute.

2573

2574 **Compliance Requirement:** Compliant systems SHALL implement this command.

2575

2576 **Usage:**

2577 EdgeTriggeredAlarmControl.setRearmThreshold (alarmThreshold:
2578 integer): void

2579

2580 **Parameter(s):**

2581 alarmThreshold – Data Type: integer. The desired value of RearmThreshold.

2582

2583 **Return Value(s):**

2584 Data Type: void.

2585
2586 **Possible Error Conditions:**
2587 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2588 alarm controls.
2589

2590 **5.9.5 EdgeTriggeredAlarmControl.getDirection**

2591 Queries the reader for the current value of the EdgeTriggeredAlarmControl's
2592 Direction attribute.

2593

2594 **Compliance Requirement:** Compliant systems SHALL implement this command.

2595

2596 **Usage:**

2597 EdgeTriggeredAlarmControl.getDirection (void) :
2598 EdgeTriggeredAlarmDirection

2599

2600 **Parameter(s):**

2601 Data Type: void. This command takes no parameters.

2602

2603 **Return Value(s):**

2604 Data Type: EdgeTriggeredAlarmDirection. The return data type is an
2605 enumerated type with possible values RISING and FALLING.

2606

2607 **Possible Error Conditions:**

2608 None.

2609 **5.9.6 EdgeTriggeredAlarmControl.setDirection**

2610 Command sent to the reader to set the current value of the
2611 EdgeTriggeredAlarmControl's Direction attribute.

2612

2613 **Compliance Requirement:** Compliant systems SHALL implement this command.

2614

2615 **Usage:**

2616 EdgeTriggeredAlarmControl.setDirection(direction:
2617 EdgeTriggeredAlarmDirection): void

2618

2619 **Parameter(s):**
2620 direction: – Data Type: `EdgeTriggeredAlarmDirection`. The desired value of
2621 Direction.
2622
2623 **Return Value(s):**
2624 Data Type: `void`.
2625
2626 **Possible Error Conditions:**
2627 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2628 alarm controls.
2629
2630

2631 **5.9.7 EdgeTriggeredAlarmControl.getStatus**

2632 Queries the reader for the current value of the `EdgeTriggeredAlarmControl`'s
2633 Status attribute.

2634

2635 **Compliance Requirement:** Compliant systems SHALL implement this command.

2636

2637 **Usage:**

2638 `EdgeTriggeredAlarmControl.getStatus (void) :`
2639 `EdgeTriggeredAlarmStatus`

2640

2641 **Parameter(s):**

2642 Data Type: `void`. This command takes no parameters.

2643

2644 **Return Value(s):**

2645 Data Type: `EdgeTriggeredAlarmStatus`. The return data type is
2646 an enumerated type with possible values ARMED OR FIRED.

2647

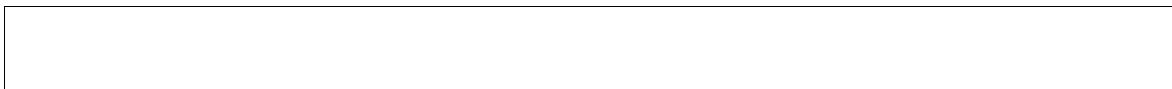
2648 **Possible Error Conditions:**

2649 None.

2650

2651

2652



2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672

--

5.10 TTOperationalStatusAlarmControl

This class extends `AlarmControl` to control alarms generated when a monitored model element of type `OperationalStatus` transitions to a new value. This type of alarm is called “transition triggered” (abbreviated “TT”).

2673
2674
2675
2676
2677
2678
2679
2680

--

TTOperationalStatusAlarmControl
- TriggerFromState : OperationalStatus - TriggerToState : OperationalStatus
+ getTriggerFromState() : OperationalStatus + setTriggerFromState(triggerFromState : OperationalStatus) : void + getTriggerToState() : OperationalStatus + setTriggerToState(triggerToState : OperationalStatus) : void

Figure 17 TTOperationalStatusAlarmControl UML

The TriggerFromState attribute (of enumerated type OperationalStatus) indicates the value of the monitored, OperationalStatus-valued, model element prior to the state transition which triggers the alarm. The TriggerToState attribute (of enumerated type OperationalStatus) indicates the value of the monitored, OperationalStatus-valued, model element immediately after the state transition which triggers the alarm.

5.10.1 TTOperationalStatusAlarmControl.getTriggerFromState

Queries the reader for the current value of the TTOperationalStatusAlarmControl's TriggerFromState attribute.

Compliance Requirement: Compliant systems SHALL implement this command.

Usage:

`TTOperationalStatusAlarmControl.getTriggerFromState (void) : OperationalStatus`

Parameter(s):

Data Type: void. This command takes no parameters.

Return Value(s):

Data Type: OperationalStatus. The return data type is an enumerated type with possible values UNKNOWN, OTHER, UP, DOWN and ANY.

Possible Error Conditions:

None.

2709 **5.10.2 TTOperationalStatusAlarmControl.setTriggerFromSta**
2710 **te**

2711 Command sent to the reader to set the current value of the
2712 TTOperationalStatusAlarmControl's TriggerFromState attribute.

2713

2714 **Compliance Requirement:** Compliant systems SHALL implement this command.

2715

2716 **Usage:**

2717 TTOperationalStatusAlarmControl.setTriggerFromState
2718 (triggerFromState: OperationalStatus): void

2719

2720 **Parameter(s):**

2721 operationalStatus – Data Type: OperationalStatus. The desired value of
2722 TriggerFromState.

2723

2724 **Return Value(s):**

2725 Data Type: void.

2726

2727 **Possible Error Conditions:**

2728 ERROR_AUTHORIZATION. A management application may not be authorized to alter
2729 alarm controls.

2730

2731 **5.10.3 TTOperationalStatusAlarmControl.getTriggerToState**

2732 Queries the reader for the current value of the
2733 TTOperationalStatusAlarmControl's TriggerToState attribute.

2734

2735 **Compliance Requirement:** Compliant systems SHALL implement this command.

2736

2737 **Usage:**

2738 TTOperationalStatusAlarmControl.getTriggerToState (void):
2739 OperationalStatus

2740

2741 **Parameter(s):**

2742 Data Type: void. This command takes no parameters.

2743

2744 **Return Value(s):**

2745 Data Type: `OperationalStatus`. The return data type is an enumerated type with
2746 possible values UNKNOWN, OTHER, UP, DOWN and ANY.

2747 **Possible Error Conditions:**

2748 None.

2749 **5.10.4 TOperationalStatusAlarmControl.setTriggerToState**

2750 Command sent to the reader to set the current value of the
2751 TOperationalStatusAlarmControl's TriggerToState attribute.

2752

2753 **Compliance Requirement:** Compliant systems SHALL implement this command.

2754

2755 **Usage:**

2756 `TOperationalStatusAlarmControl.setTriggerToState`
2757 `(triggerToState: OperationalStatus): void`

2758

2759 **Parameter(s):**

2760 `operationalStatus` – Data Type: `OperationalStatus`. The desired value of
2761 `TriggerToState`.

2762

2763 **Return Value(s):**

2764 Data Type: `void`.

2765

2766 **Possible Error Conditions:**

2767 `ERROR_AUTHORIZATION`. A management application may not be authorized to alter
2768 alarm controls.

2769

2770

2771

2772

2773

2774

2775

2776

2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799

6 Reader Layer – Alarm Notifications

6.1 Alarm Objects

The AlarmControl class, and its subclasses, specify those elements of a Reader's object model that control alarm generation. The alarms themselves are messages that are sent to management systems and host applications whose responsibility it is to monitor Reader health and operations. The Reader object model has been expanded to include class definitions that specify, in an abstract manner, the contents of these alarm messages. Note that these Alarm objects do not specify persistent data objects maintained by the readers. These objects are instantiated in the course of a reader's generation or a management system's receipt and processing of alarms; once an alarm message is transmitted or processed, its corresponding information object disappears.

In the object descriptions below, the UML for each object is presented with its elements and operations or methods. Each object element has an associated getter. Since Alarms can only be generated by the reader device and can not be modified by the consumer, no setters are provided. Getters are used during message processing.

6.2 Alarm

Alarm is the base of all the classes within the object model that define the contents of alarm messages.

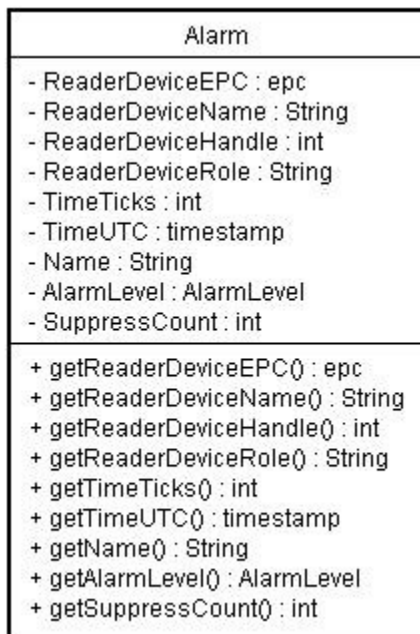


Figure 18 Alarm UML

Each Alarm MAY carry a ReaderDeviceEPC attribute (of type epc) specifying the EPC of the reader that generated the alarm. The reader EPC is ReaderDevice.EPC element of the Object model.

2831 Each Alarm SHALL carry a `ReaderDeviceName` attribute (of type `String`)
 2832 specifying the name of the reader that generated the alarm. The reader name is
 2833 `ReaderDevice.Name` element of the Object model.

2834 Each Alarm MAY carry a `ReaderDeviceHandle` attribute (of type `integer`)
 2835 specifying the handle of the reader that generated the alarm. The reader handle is
 2836 `ReaderDevice.Handle` element of the Object model.

2837 Each Alarm MAY carry a `ReaderDeviceRole` attribute (of type `String`)
 2838 specifying the Role of the reader that generated the alarm. The reader Role is
 2839 `ReaderDevice.Role` element of the object model.

2840 Each Alarm SHALL also carry a `TimeTicks` attribute (of type `integer`) recording the
 2841 number of ticks when the alarm was generated. It's value is the value of the
 2842 `ReaderDevice.TimeTicks` element of the Object model at the time the Alarm was
 2843 generated.

2844 Each Alarm MAY also carry a `TimeUTC` attribute (of type `timestamp`) recording when
 2845 the alarm was generated. It's value is the value of the `ReaderDevice.TimeUTC`
 2846 element of the object model at the time the Alarm was generated.

2847 Each Alarm SHALL carry a `Name` attribute (of type `String`) identifying the type of
 2848 Alarm, e.g., “FreeMemoryAlarm”, “TagListFullAlarm”, “ReadPointOperStatusAlarm”.
 2849 It SHALL matched on of the Alarm types specified in this section in this
 2850 specification. Each Alarm SHALL carry an `AlarmLevel` attribute (of type
 2851 `AlarmLevel`), indicating the severity level assigned to the alarm.

2852 Each Alarm SHALL carry a `SuppressCount` attribute (of type `integer`), indicating the
 2853 number of times the generation of this Alarm has been suppressed. This attribute is
 2854 incremented every time an alarm condition is encountered while the
 2855 `SuppressInterval` is in effect, preventing the reader from generating the Alarm.
 2856 This value is reset to 0 after the Alarm is generated. Suppression is controlled by the
 2857 `SuppressInterval` attribute of `AlarmControl`.

2858 Expanding on the example presented in the `AlarmControl` section, a
 2859 `FailedLockAlarm` is generated at 09:00:00. At 09:00:05, the alarm condition is
 2860 encountered again while the `SuppressInterval` has not yet been reached. At this
 2861 time, the `SuppressCount` for `AntennaReadPoint1.FailedKillAlarmControl` is
 2862 incremented by one and the alarm generation is suppressed. The same occurs at
 2863 09:00:08, incrementing `SuppressCount` by 1 again while suppressing the Alarm. At
 2864 09:00:16, the Alarm is generated containing a `SuppressCount` equal to 2, indicating
 2865 the Alarm was not sent on 2 prior occasions during the interval from 09:00:00 (first
 2866 alarm) until now. The `SuppressCount` for this object will then be reset to 0 after the
 2867 successful generation of the Alarm.

2868

Time	Object	Alarm Condition	Action	Suppress Count
------	--------	-----------------	--------	----------------

09:00:00	AntennaReadPoint1	Kill Failure	Generate Alarm	0
09:00:01	AntennaReadPoint2	Kill Failure	Generate Alarm	0
09:00:05	AntennaReadPoint1	Kill Failure	Suppress Alarm	Increment
09:00:06	AntennaReadPoint1	Write Failure	Generate Alarm	0
09:00:08	AntennaReadPoint1	Kill Failure	Suppress Alarm	Increment
09:00:16	AntennaReadPoint1	Kill Failure	Generate Alarm	2

2869

2870 **6.2.1 Alarm.getReaderDeviceEPC**

2871 Queries the Alarm for the value of the ReaderDeviceEPC attribute.

2872

2873 **Compliance Requirement:** If this alarm is implemented, compliant systems MAY
2874 implement this command.

2875

2876 **Usage:**

2877 Alarm.getReaderDeviceEPC (void): epc

2878

2879 **Parameter(s):**

2880 Data Type: void. This command takes no parameters.

2881

2882 **Return Value(s):**

2883 Data Type: epc. The EPC for the reader device.

2884

2885 **Possible Error Conditions:**

2886 None.

2887 **6.2.2 Alarm.getReaderDeviceName**

2888 Queries the Alarm for the value of the ReaderDeviceName attribute.

2889

2890 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
2891 implement this command.

2892

2893 **Usage:**

2894 Alarm.getReaderDeviceName (void): string

2895

2896 **Parameter(s):**

2897 Data Type: `void`. This command takes no parameters.

2898

2899 **Return Value(s):**

2900 Data Type: `string`. The name for the reader device.

2901

2902 **Possible Error Conditions:**

2903 None.

2904 **6.2.3 Alarm.getReaderDeviceHandle**

2905 Queries the Alarm for the value of the `ReaderDeviceHandle` attribute.

2906

2907 **Compliance Requirement:** Compliant systems MAY implement this command.

2908

2909 **Usage:**

2910 `Alarm.getReaderDeviceHandle (void) : integer`

2911

2912 **Parameter(s):**

2913 Data Type: `void`. This command takes no parameters.

2914

2915 **Return Value(s):**

2916 Data Type: `integer`. The handle for the reader device.

2917

2918 **Possible Error Conditions:**

2919 None.

2920 **6.2.4 Alarm.getReaderDeviceRole**

2921 Queries the Alarm for the value of the `ReaderDeviceRole` attribute.

2922

2923 **Compliance Requirement:** Compliant systems MAY implement this command.

2924

2925 **Usage:**

2926 `Alarm.getReaderDeviceRole (void) : string`

2927

2928 **Parameter(s):**

2929 Data Type: `void`. This command takes no parameters.

2930

2931 **Return Value(s):**

2932 Data Type: `string`. The role for the reader device.

2933

2934 **Possible Error Conditions:**

2935 None.

2936 **6.2.5 Alarm.getTimeTicks**

2937 Queries the Alarm for the value of the `TimeTicks` attribute.

2938

2939 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
2940 implement this command.

2941

2942 **Usage:**

2943 `Alarm.getTimeTicks (void) : integer`

2944

2945 **Parameter(s):**

2946 Data Type: `void`. This command takes no parameters.

2947

2948 **Return Value(s):**

2949 Data Type: `integer`. The `TimeTicks` for the reader device.

2950

2951 **Possible Error Conditions:**

2952 None.

2953 **6.2.6 Alarm.getTimeUTC**

2954 Queries the Alarm for the value of the `TimeUTC` attribute.

2955

2956 **Compliance Requirement:** Compliant systems MAY implement this command.

2957

2958 **Usage:**

2959 Alarm.getTimeUTC (void): timestamp

2960

2961 **Parameter(s):**

2962 Data Type: void. This command takes no parameters.

2963

2964 **Return Value(s):**

2965 Data Type: timestamp. The UTC time for the reader device.

2966

2967 **Possible Error Conditions:**

2968 None.

2969 **6.2.7 Alarm.getName**

2970 Queries the Alarm for the value of the Name attribute.

2971

2972 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
2973 implement this command.

2974

2975 **Usage:**

2976 Alarm.getName (void): string

2977

2978 **Parameter(s):**

2979 Data Type: void. This command takes no parameters.

2980

2981 **Return Value(s):**

2982 Data Type: string. The name for the alarm.

2983

2984 **Possible Error Conditions:**

2985 None.

2986 **6.2.8 Alarm.getAlarmLevel**

2987 Queries the Alarm for the value of the AlarmLevel attribute.

2988

2989 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
2990 implement this command.

2991

2992 **Usage:**

2993 `Alarm.getAlarmLevel (void): AlarmLevel`

2994

2995 **Parameter(s):**

2996 Data Type: `void`. This command takes no parameters.

2997

2998 **Return Value(s):**

2999 Data Type: `AlarmLevel`. The alarm severity level.

3000

3001 **Possible Error Conditions:**

3002 None.

3003 **6.2.9 Alarm.getSuppressCount**

3004 Queries the Alarm for the value of the `SupressCount` attribute.

3005

3006 **Compliance Requirement:** Compliant systems SHALL implement this command.

3007

3008 **Usage:**

3009 `Alarm.getSuppressCount (void): integer`

3010

3011 **Parameter(s):**

3012 Data Type: `void`. This command takes no parameters.

3013

3014 **Return Value(s):**

3015 Data Type: `integer`. The number of times since last alarm
3016 generation that this alarm has been suppressed from being
3017 generated.

3018

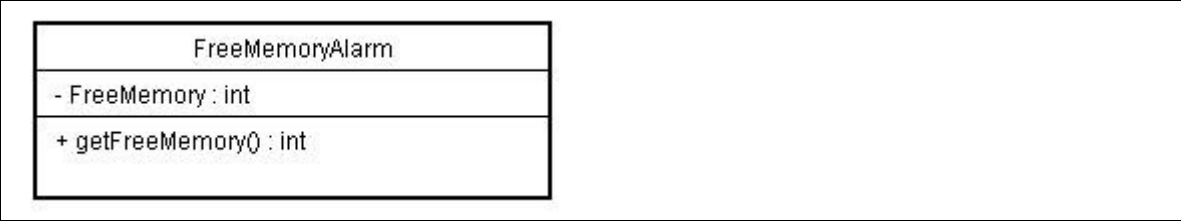
3019 **Possible Error Conditions:**

3020 None.

3021

3022 **6.3 FreeMemoryAlarm**

3023 FreeMemoryAlarm extends the Alarm class. Its receipt signals the movement of a
3024 reader device's free memory (represented in the abstract model by
3025 ReaderDevice.FreeMemory) below a specified threshold value. The abstract model's
3026 ReaderDevice.FreeMemoryAlarmControl object controls the triggering of alarms
3027 of this type.



3028
3029
3030 **Figure 19 FreeMemoryAlarm UML**

3031
3032 The FreeMemory attribute SHALL carry the value of ReaderDevice.FreeMemory
3033 when the alarm was triggered.

3034 **Compliance Requirement:** Compliant systems MAY implement this alarm.

3035 **6.3.1 FreeMemoryAlarm.getFreeMemory**

3036 Queries the FreeMemoryAlarm for the value of the FreeMemory attribute.

3037
3038 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3039 implement this command.

3040
3041 **Usage:**

3042 FreeMemoryAlarm.getFreeMemory (void) : integer

3043
3044 **Parameter(s):**

3045 Data Type: void. This command takes no parameters.

3046
3047 **Return Value(s):**

3048 Data Type: integer. The free memory for the reader device.

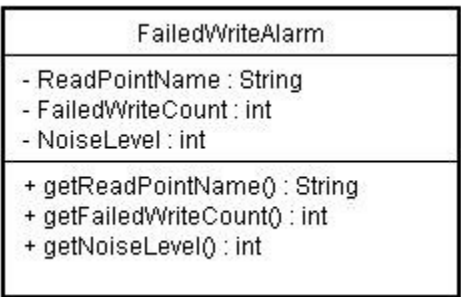
3049
3050 **Possible Error Conditions:**

3051 None.

3052

3053 **6.4 FailedWriteAlarm**

3054 FailedWriteAlarm extends the Alarm class. Its receipt signals a tag write or tag block
3055 write failure. The abstract model's
3056 AntennaReadPoint.FailedWriteAlarmControl data element controls the
3057 triggering of alarms of this type.



3058

3059

3060

3061

3062

3063

3064 **Figure 20 FailedWriteAlarm UML**

3065

3066 The ReadPointName attribute identifies the read point (an AntennaReadPoint) over
3067 which the write failure occurred. This SHALL be the value of that read point's
3068 ReadPoint.Name element.

3069 The FailedWriteCount attribute SHALL carry the value of
3070 AntennaReadPoint.FailedWriteCount element after the write failure occurred.

3071 The NoiseLevel attribute MAY carry the value of the
3072 AntennaReadPoint.NoiseLevel element when the write failure occurred.

3073 **Compliance Requirement:** Compliant systems MAY implement this alarm.

3074 **6.4.1 FailedWriteAlarm.getReadPointName**

3075 Queries the FailedWriteAlarm for the value of the ReadPointName attribute.

3076

3077 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3078 implement this command.

3079

3080 **Usage:**
3081 `FailedWriteAlarm.getReadPointName (void): string`

3082

3083 **Parameter(s):**

3084 Data Type: `void`. This command takes no parameters.

3085

3086 **Return Value(s):**

3087 Data Type: `string`. The name of the `ReadPoint`.

3088

3089 **Possible Error Conditions:**

3090 None.

3091 **6.4.2 FailedWriteAlarm.getFailedWriteCount**

3092 Queries the `FailedWriteAlarm` for the value of the `FailedWriteCount` attribute.

3093

3094 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3095 implement this command.

3096

3097 **Usage:**

3098 `FailedWriteAlarm.getFailedWriteCount (void): integer`

3099

3100 **Parameter(s):**

3101 Data Type: `void`. This command takes no parameters.

3102

3103 **Return Value(s):**

3104 Data Type: `integer`. The `FailedWriteCount` for the `ReadPoint`.

3105

3106 **Possible Error Conditions:**

3107 None.

3108 **6.4.3 FailedWriteAlarm.getNoiseLevel**

3109 Queries the `FailedWriteAlarm` for the value of the `NoiseLevel` attribute.

3110

3111 **Compliance Requirement:** Compliant systems MAY implement this command.

3112

3113 **Usage:**

3114 `FailedWriteAlarm.getNoiseLevel (void) : integer`

3115

3116 **Parameter(s):**

3117 Data Type: `void`. This command takes no parameters.

3118

3119 **Return Value(s):**

3120 Data Type: `integer`. The `NoiseLevel` for the `ReadPoint`.

3121

3122 **Possible Error Conditions:**

3123 None.

3124

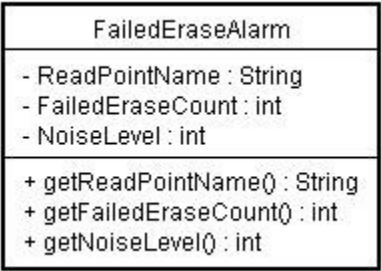
3125 **6.5 FailedEraseAlarm**

3126 `FailedEraseAlarm` extends the `Alarm` class. Its receipt signals a tag erase or tag block

3127 erase failure. The abstract model's

3128 `AntennaReadPoint.FailedEraseAlarmControl` data element controls the

3129 triggering of alarms of this type.



3130

3131 **Figure 21 FailedEraseAlarm UML**

3132

3133 The `ReadPointName` attribute identifies the read point (an `AntennaReadPoint`) over

3134 which the erase failure occurred. This SHALL be the value of that read point's

3135 `ReadPoint.Name` element.

3136 The `FailedEraseCount` attribute SHALL carry the value of

3137 `AntennaReadPoint.FailedEraseCount` element after the erase failure occurred.

3138 The `NoiseLevel` attribute MAY carry the value of the

3139 `AntennaReadPoint.NoiseLevel` element when the erase failure occurred.

3140 Note that the erase failure could be attributable to the use of an incorrect tag erase
3141 password.

3142 Compliance Requirement: Compliant systems MAY implement this alarm.

3143 **6.5.1 FailedEraseAlarm.getReadPointName**

3144 Queries the FailedEraseAlarm for the value of the `ReadPointName` attribute.

3145

3146 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3147 implement this command.

3148

3149 **Usage:**

3150 `FailedEraseAlarm.getReadPointName (void): string`

3151

3152 **Parameter(s):**

3153 Data Type: `void`. This command takes no parameters.

3154

3155 **Return Value(s):**

3156 Data Type: `string`. The name of the `ReadPoint`.

3157

3158 **Possible Error Conditions:**

3159 None.

3160 **6.5.2 FailedEraseAlarm.getFailedEraseCount**

3161 Queries the FailedEraseAlarm for the value of the `FailedEraseCount` attribute.

3162

3163 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3164 implement this command.

3165

3166 **Usage:**

3167 `FailedEraseAlarm.getFailedEraseCount (void): integer`

3168

3169 **Parameter(s):**

3170 Data Type: `void`. This command takes no parameters.

3171

3172 **Return Value(s):**

3173 Data Type: integer. The FailedEraseCount for the ReadPoint.

3174

3175 **Possible Error Conditions:**

3176 None.

3177 **6.5.3 FailedEraseAlarm.getNoiseLevel**

3178 Queries the FailedEraseAlarm for the value of the NoiseLevel attribute.

3179

3180 **Compliance Requirement:** Compliant systems MAY implement this command.

3181

3182 **Usage:**

3183 FailedEraseAlarm.getNoiseLevel (void) : integer

3184

3185 **Parameter(s):**

3186 Data Type: void. This command takes no parameters.

3187

3188 **Return Value(s):**

3189 Data Type: integer. The NoiseLevel for the ReadPoint.

3190

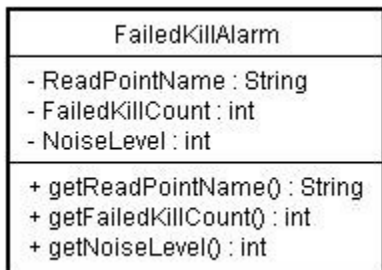
3191 **Possible Error Conditions:**

3192 None.

3193

3194 **6.6 FailedKillAlarm**

3195 FailedKillAlarm extends the Alarm class. Its receipt signals a tag kill failure. The
3196 abstract model's AntennaReadPoint.FailedKillAlarmControl data element
3197 controls the triggering of alarms of this type.



3198

3199

3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214

3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231



Figure 22 FailedKillAlarm UML

The `ReadPointName` attribute identifies the read point (an `AntennaReadPoint`) over which the kill failure occurred. This SHALL be the value of that read point's `ReadPoint.Name` element.

The `FailedKillCount` attribute SHALL carry the value of `AntennaReadPoint.FailedKillCount` element after the kill failure occurred.

The `NoiseLevel` attribute MAY carry the value of the `AntennaReadPoint.NoiseLevel` element when the kill failure occurred.

Note that the kill failure could be attributable to the use of an incorrect tag kill password.

Compliance Requirement: Compliant systems MAY implement this alarm.

6.6.1 FailedKillAlarm.getReadPointName

Queries the FailedKillAlarm for the value of the `ReadPointName` attribute.

Compliance Requirement: If this alarm is implemented, compliant systems SHALL implement this command.

Usage:

`FailedKillAlarm.getReadPointName (void): string`

Parameter(s):

Data Type: `void`. This command takes no parameters.

Return Value(s):

Data Type: `string`. The name of the `ReadPoint`.

Possible Error Conditions:

None.

3232 **6.6.2 FailedKillAlarm.getFailedKillCount**

3233 Queries the FailedKillAlarm for the value of the FailedKillCount attribute.

3235 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3236 implement this command.

3238 **Usage:**

3239 FailedKillAlarm.getFailedKillCount (void): integer

3241 **Parameter(s):**

3242 Data Type: void. This command takes no parameters.

3244 **Return Value(s):**

3245 Data Type: integer. The FailedKillCount for the ReadPoint.

3247 **Possible Error Conditions:**

3248 None.

3249 **6.6.3 FailedKillAlarm.getNoiseLevel**

3250 Queries the FailedKillAlarm for the value of the NoiseLevel attribute.

3252 **Compliance Requirement:** Compliant systems MAY implement this command.

3254 **Usage:**

3255 FailedKillAlarm.getNoiseLevel (void): integer

3257 **Parameter(s):**

3258 Data Type: void. This command takes no parameters.

3260 **Return Value(s):**

3261 Data Type: integer. The NoiseLevel for the ReadPoint.

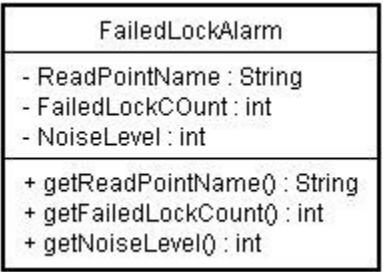
3263 **Possible Error Conditions:**

3264 None.

3265

3266 **6.7 FailedLockAlarm**

3267 FailedLockAlarm extends the Alarm class. Its receipt signals a tag lock failure. The
3268 abstract model's AntennaReadPoint.FailedLockAlarmControl data element
3269 controls the triggering of alarms of this type.



3270

3271

3272

3273

3274

3275

3276

3277 **Figure 23 FailedKillAlarm UML**

3278

3279 The ReadPointName attribute identifies the read point (an AntennaReadPoint) over
3280 which the lock failure occurred. This SHALL be the value of that read point's
3281 ReadPoint.Name element.

3282 The FailedLockCount attribute SHALL carry the value of
3283 AntennaReadPoint.FailedLockCount element after the read failure occurred.

3284 The NoiseLevel attribute MAY carry the value of the
3285 AntennaReadPoint.NoiseLevel element when the lock failure occurred.

3286 Note that the lock failure could be attributable to the use of an incorrect tag lock
3287 password.

3288 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3289 implement this alarm.

3290 **6.7.1 FailedLockAlarm.getReadPointName**

3291 Queries the FailedLockAlarm for the value of the ReadPointName attribute.

3292

3293 **Compliance Requirement:** Compliant systems MAY implement this command.

3294

3295 **Usage:**

3296 `FailedLockAlarm.getReadPointName (void): string`

3297

3298 **Parameter(s):**

3299 Data Type: `void`. This command takes no parameters.

3300

3301 **Return Value(s):**

3302 Data Type: `string`. The name of the `ReadPoint`.

3303

3304 **Possible Error Conditions:**

3305 None.

3306 **6.7.2 FailedLockAlarm.getFailedLockCount**

3307 Queries the `FailedLockAlarm` for the value of the `FailedLockCount` attribute.

3308

3309 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3310 implement this command.

3311

3312 **Usage:**

3313 `FailedLockAlarm.getFailedLockCount (void): integer`

3314

3315 **Parameter(s):**

3316 Data Type: `void`. This command takes no parameters.

3317

3318 **Return Value(s):**

3319 Data Type: `integer`. The `FailedLockCount` for the `ReadPoint`.

3320

3321 **Possible Error Conditions:**

3322 None.

3323 **6.7.3 FailedLockAlarm.getNoiseLevel**

3324 Queries the `FailedLockAlarm` for the value of the `NoiseLevel` attribute.

3325

Compliance Requirement: Compliant systems MAY implement this command.

Usage:

`FailedLockAlarm.getNoiseLevel (void): integer`

Parameter(s):

Data Type: `void`. This command takes no parameters.

Return Value(s):

Data Type: `integer`. The `NoiseLevel` for the `ReadPoint`.

Possible Error Conditions:

None.

6.8 FailedMemReadAlarm

`FailedMemReadAlarm` extends the `Alarm` class. Its receipt signals a tag user-memory read failure. The abstract model's `AntennaReadPoint.FailedMemReadAlarmControl` data element controls the triggering of alarms of this type.

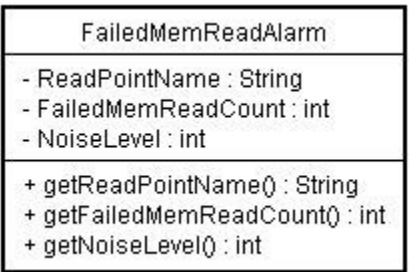


Figure 24 FailedMemReadAlarm UML

3354 The `ReadPointName` attribute identifies the read point (an `AntennaReadPoint`) over
3355 which the memory read failure occurred. This SHALL be the value of that read point's
3356 `ReadPoint.Name` element.

3357 The `FailedMemReadCount` attribute SHALL carry the value of
3358 `AntennaReadPoint.FailedMemReadCount` element after the read failure occurred.

3359 The `NoiseLevel` attribute MAY carry the value of the
3360 `AntennaReadPoint.NoiseLevel` element when the memory read failure occurred.

3361 Note that the read failure could be attributable to the use of an incorrect tag memory read
3362 password.

3363 **Compliance Requirement:** Compliant systems MAY implement this alarm.

3364 **6.8.1 FailedMemReadAlarm.getReadPointName**

3365 Queries the FailedMemReadAlarm for the value of the `ReadPointName` attribute.

3366

3367 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3368 implement this command.

3369

3370 **Usage:**

3371 `FailedMemReadAlarm.getReadPointName (void): string`

3372

3373 **Parameter(s):**

3374 Data Type: `void`. This command takes no parameters.

3375

3376 **Return Value(s):**

3377 Data Type: `string`. The name of the `ReadPoint`.

3378

3379 **Possible Error Conditions:**

3380 None.

3381 **6.8.2 FailedMemReadAlarm.getFailedMemReadCount**

3382 Queries the FailedMemReadAlarm for the value of the `FailedMemReadCount` attribute.

3383

3384 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3385 implement this command.

3386

3387 **Usage:**

3388 `FailedMemReadAlarm.getFailedMemReadCount (void): integer`
3389
3390 **Parameter(s):**
3391 Data Type: `void`. This command takes no parameters.
3392
3393 **Return Value(s):**
3394 Data Type: `integer`. The `FailedMemReadCount` for the `ReadPoint`.
3395
3396 **Possible Error Conditions:**
3397 None.

3398 **6.8.3 FailedMemReadAlarm.getNoiseLevel**

3399 Queries the `FailedMemReadAlarm` for the value of the `NoiseLevel` attribute.

3400

3401 **Compliance Requirement:** Compliant systems MAY implement this command.

3402

3403 **Usage:**

3404 `FailedMemReadAlarm.getNoiseLevel (void): integer`

3405

3406 **Parameter(s):**

3407 Data Type: `void`. This command takes no parameters.

3408

3409 **Return Value(s):**

3410 Data Type: `integer`. The `NoiseLevel` for the `ReadPoint`.

3411

3412 **Possible Error Conditions:**

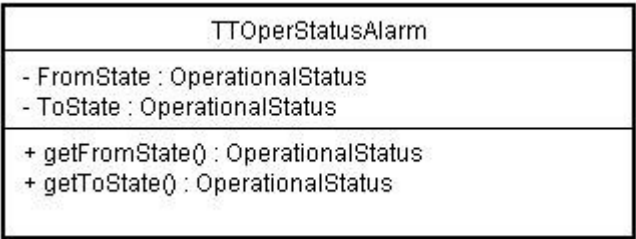
3413 None.

3414

3415

3416 **6.9 TTOperStatusAlarm**

3417 TTOperStatusAlarm extends the Alarm class, and is the base class for all Transition
3418 Triggered Operational Status Alarms. The base class identifies the particular transition
3419 between Operational Status states that triggered the alarm.



3420

3421 **Figure 25 TTOperStatusAlarm UML**

3422

3423 The FromState attribute SHALL identify the originating OperationalStatus before
3424 the Alarm is generated.

3425 The ToState attribute SHALL identify the OperationalStatus at the time the
3426 Alarm is generated.

3427 Valid values for OperationalStatus are defined in 7.1.2
3428 OperationalStatus.

3429

3430

3431

3432 **6.9.1 TTOperStatusAlarm.getFromState**

3433 Queries the TTOperStatusAlarm for the value of the OperationalStatus attribute
3434 before the transition.

3435

3436 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3437 implement this command.

3438

3439 **Usage:**

3440 TTOperStatusAlarm.getFromState (void) : OperationalStatus

3441

3442 **Parameter(s):**

3443 Data Type: void. This command takes no parameters.

3444

3445 **Return Value(s):**

3446 Data Type: `OperationalStatus`. The `OperationalStatus` before the

3447 transition.

3448

3449 **Possible Error Conditions:**

3450 None.

3451 **6.9.2 TTOperStatusAlarm.getToState**

3452 Queries the `TTOperStatusAlarm` for the value of the `OperationalStatus` attribute

3453 after the transition.

3454

3455 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL

3456 implement this command.

3457

3458 **Usage:**

3459 `TTOperStatusAlarm.getToState (void) : OperationalStatus`

3460

3461 **Parameter(s):**

3462 Data Type: `void`. This command takes no parameters.

3463

3464 **Return Value(s):**

3465 Data Type: `OperationalStatus`. The `OperationalStatus` after the

3466 transition.

3467

3468 **Possible Error Conditions:**

3469 None.

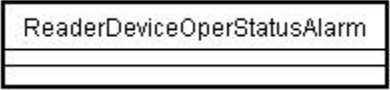
3470 **6.10 ReaderDeviceOperStatusAlarm**

3471 `ReaderDeviceOperStatusAlarm` extends the `TTOperStatusAlarm` class. Its

3472 receipt signals a change in the operational status of a Reader. The abstract model's

3473 `ReaderDevice.OperStatusAlarmControl` data element controls the triggering of

3474 alarms of this type.



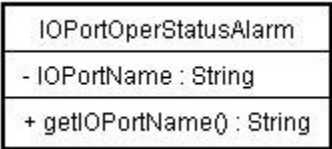
3475

3476 **Figure 26 ReaderDeviceStatusAlarm UML**

3477
3478 **Compliance Requirement:** Compliant systems SHALL implement this alarm if
3479 AlarmChannel is implemented.

3481 **6.11 IOPortOperStatusAlarm**

3482 IOPortOperStatusAlarm extends the TToperStatusAlarm class. Its receipt signals
3483 a change in the operational status of a Reader's IO Port. The abstract model's
3484 IOPort.OperStatusAlarmControl data element controls the triggering of alarms of
3485 this type.



3486
3487
3488 **Figure 27 IOPortOperStatusAlarm UML**

3489
3490 The IOPortName message attribute SHALL identify the name of the IO Port that
3491 experienced the alarm-triggering state transition, i.e., the value of the respective
3492 IOPort.Name model element.

3493
3494 **Compliance Requirement:** Compliant systems SHALL implement this alarm if
3495 AlarmChannel and IOPort are implemented.

3496 **6.11.1 IOPortOperStatusAlarm.getIOPortName**

3497 Queries the IOPortStatusAlarm for the value of the IOPortName attribute.

3498
3499 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3500 implement this command.

3501
3502 **Usage:**

3503 IOPortOperStatus.getIOPortName (void) : string

3504
3505 **Parameter(s):**

3506 Data Type: void. This command takes no parameters.

3507

3508 **Return Value(s):**

3509 Data Type: string. The name of the IOPort.

3510

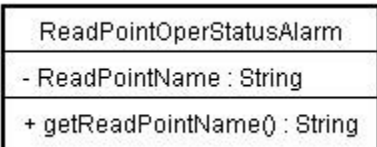
3511 **Possible Error Conditions:**

3512 None.

3513

3514 **6.12 ReadPointOperStatusAlarm**

3515 ReadPointOperStatusAlarm extends the TTOperStatusAlarm class. Its receipt
3516 signals a change in the operational status of one of a Reader's Read Points. The abstract
3517 model's ReadPoint.OperStatusAlarmControl data element controls the triggering
3518 of alarms of this type.



3519

3520

3521 **Figure 28 ReadPointOperStatusAlarm UML**

3522

3523 The ReadPointName message attribute SHALL identify the name of the Read Point that
3524 experienced the alarm-triggering state transition, i.e., the value of the respective
3525 ReadPoint.Name model element.

3526 **Compliance Requirement:** Compliant systems SHALL implement this alarm if
3527 AlarmChannel is implemented.

3528 **6.12.1 ReadPointOperStatusAlarm.getReadPointName**

3529 Queries the ReadPointOperStatusAlarm for the value of the ReadPointName attribute.

3530

3531 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3532 implement this command.

3533

3534 **Usage:**

3535 ReadPointOperStatusAlarm.getReadPointName (void) : string

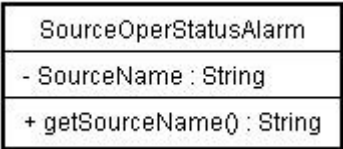
3536

3537 **Parameter(s):**

3538 Data Type: void. This command takes no parameters.
3539
3540 **Return Value(s):**
3541 Data Type: string. The name of the ReadPoint.
3542
3543 **Possible Error Conditions:**
3544 None.
3545

3546 **6.13 SourceOperStatusAlarm**

3547 SourceOperStatusAlarm extends the TTOperStatusAlarm class. Its receipt signals
3548 a change in the operational status of a logical source of EPC data on a Reader. The
3549 abstract model's SourceOperStatusAlarmControl data element controls the
3550 triggering of alarms of this type.



3551
3552
3553 **Figure 29 SourceOperStatusAlarm UML**

3554
3555 The SourceName message attribute SHALL identify the name of the logical source that
3556 experienced the alarm-triggering state transition, i.e., the value of the respective
3557 SourceName model element.
3558 **Compliance Requirement:** Compliant systems SHALL implement this alarm if
3559 AlarmChannel is implemented.

3560 **6.13.1 SourceOperStatusAlarm.getSourceName**

3561 Queries the SourceOperStatusAlarm for the value of the SourceName attribute.

3562
3563 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3564 implement this command.

3565
3566 **Usage:**
3567 SourceOperStatusAlarm.getSourceName (void) : string

3568

3569 **Parameter(s):**
3570 Data Type: void. This command takes no parameters.
3571
3572 **Return Value(s):**
3573 Data Type: string. The name of the Source.
3574
3575 **Possible Error Conditions:**
3576 None.
3577

3578 **6.14 NotificationChannelOperStatusAlarm**

3579 NotificationChannelOperStatusAlarm extends the TTOperStatusAlarm class.
3580 Its receipt signals a change in the operational status of one of a Reader's notification
3581 channels. The abstract model's NotificationChannel.OperStatusAlarmControl
3582 data element controls the triggering of alarms of this type.



3583
3584 **Figure 30 NotificationChannelOperStatusAlarm UML**
3585



3587 The NotificationChannelName message attribute SHALL identify
3588 the name of the notification channel that experienced the alarm-triggering state
3589 transition, i.e., the value of the respective NotificationChannel.Name model
3590 element.

3591 **Compliance Requirement:** Compliant systems SHALL implement this alarm if
3592 AlarmChannel and NotificationChannel are implemented.

3593 **6.14.1 NotificationChannelOperStatusAlarm.getNotificationC**
3594 **hannelName**

3595 Queries the NotificationChannelOperStatusAlarm for the value of the
3596 NotificationChannelName attribute.

3597
3598 **Compliance Requirement:** If this alarm is implemented, compliant systems SHALL
3599 implement this command.

3600

3601 **Usage:**

3602 `NotificationChannelOperStatusAlarm.getNotificationChannelNa`
3603 `me(void): string`

3604

3605 **Parameter(s):**

3606 Data Type: `void`. This command takes no parameters.

3607

3608 **Return Value(s):**

3609 Data Type: `string`. The name of the `NotificationChannel`.

3610

3611 **Possible Error Conditions:**

3612 None.

3613

3614 7 Enumerated types

3615 7.1.1 AdministrativeStatus

3616 The administrative status represents the host's desired state for an object.

Value	Description
UP	The host desires that the operational status be “up”
DOWN	The host desires that the operational status be “down”

3617

3618 7.1.2 OperationalStatus

3619 The operational status represents the actual state for an object.

Value	Description
UNKNOWN	The reader lacks the ability to determine the operational status of the object. The real status may be “up”, “down”, or “other”.
UP	The object is operational. For aggregate objects (i.e. objects that encompass other objects, as a source may encompass multiple read points) this means that all objects that are administratively up are either operationally up or operationally unknown.

DOWN	The object is not operational. For aggregate objects (i.e. objects that encompass other objects, as a source may encompass multiple read points) this means that all objects that are administratively up are either operationally down or operationally unknown.
OTHER	The object is neither fully up, nor fully down. For aggregate objects (i.e. objects that encompass other objects, as a source may encompass multiple read points) this includes the case where all sub-objects are administratively down and the aggregate object is administratively up.
ANY	A wildcard operator. The comparison “ANY”==S is true for all status enumerations, S. This is never returned by the reader, but is useful to hosts describing events. For example a host may desire an alert when the reader state changed from UP to ANY.

3620

3621 7.1.3 EdgeTriggeredAlarmDirection

3622 Controls the direction in which an edge-triggered alarm should fire.

Value	Description
RISING	Trigger on the rising edge
FALLING	Trigger on the falling edge

3623

3624 7.1.4 EdgeTriggeredAlarmStatus

3625 Describes the status of an edge-triggered alarm. See the Alarm object description for an
3626 operational model.

Value	Description
ARMED	Alarm will fire when the monitored value crosses the alarm threshold
FIRED	Alarm has fired; monitored value has not yet crossed re-arm threshold

3627

--

3628

3629 7.1.5 AlarmLevel

3630 The level of a reported alarm. When the host requests an alarm it defines the alarm level
3631 (see AlarmControl.setLevel). This additional information is then returned by the reader
3632 along with each occurrence of that alarm. It is expected that some management
3633 systems may choose to filter based on the alarm level. The alarm levels are modeled
3634 after The syslog Protocol IETF draft [SYSLOG].

Value	Description
-------	-------------

EMERGENCY	System is unusable
ALERT	Action must be taken immediately
CRITICAL	Critical conditions
ERROR	Error conditions
WARNING	Warning conditions
NOTICE	Normal but significant conditions
INFORMATIONAL	Informational messages
DEBUG	Debug level messages

3635

--

3636

3637

3638 **8 Error Handling**

3639 **8.1 Error Conditions**

3640 There are several types of error conditions:

- 3641 • Communication errors, either when a host issues a command or when the reader
3642 tries to send a notification (e.g., tag list report) to the host
- 3643 • Command execution errors that occur as response to commands sent from the host
3644 to the reader.

3645 Vendor extensions

3646 This section lists the errors that are unique to the Reader Management Specification.
3647 Refer to the Reader Protocol Specification 1.1 for the definition of errors inherited from
3648 the the Reader Protocol Specification 1.1.

3649 **8.2 Communication Errors**

3650 **8.2.1 Communication Host-to-Reader**

3651 These kinds of communication errors occur when the host tries to issue a command to the
3652 reader but the reader is not responding. These conditions are reported by the MTB-
3653 specific Transport Layer.

3654 **8.2.2 Communication Reader-to-Host**

3655 This specification does not define how the reader reacts when it isn't able to connect or
3656 exchange data over a notification channel; the exact behavior is implementation

3657 dependent. Notifications could be dropped, sent over a secondary channel, queued for
3658 later retry, raise a trap in a management console, and/or many other behaviors.

3659 8.3 Command Errors

3660 All commands support a standard `error` structure, for use in the event of errors. This
3661 contains standard attributes (some required, some optional) as well as support for vendor
3662 extensions.

3663 All standard error conditions defined can be reported with additional data about the error
3664 (formatted and transmitted as per the MTB in use):

- 3665 • SHALL: The **error code** and/or **error name**. At least one of these attributes SHALL
3666 be included in the `Error` structure. The exact contents depend on the binding, and
3667 SHALL be constant at the binding level, e.g. MTB X can be defined as always
3668 including the **error code** (but not the **error name**)
- 3669 • SHALL (where applicable): Information about the **error cause**. For example, for
3670 input parameter errors, information on which parameter failed needs to be included.
3671 Depending on the binding, this can be either the name or the index of the parameter.
3672 Vendor extensions are allowed here.
- 3673 • SHALL (where applicable): A **vendor name or identifier** for the responses to
3674 vendor-specific commands SHALL be given.
- 3675 • MAY: A **descriptive text string**, mainly used for logging purposes. The language for
3676 the text is implementation-dependent and MAY be configurable.

3677 The error code is a 16-bit (2-byte) integer. The following table defines the mapping of
3678 error conditions / error codes unique to Reader Management. Refer to Reader Protocol
3679 Specification 1.1 for errors imported from it.

3680

Error Code (Hex)	Error Name	Description
0001	ERROR_UNKNOWN	Refer to Reader Protocol Specification 1.1
0002	ERROR_COMMAND_NOT_SUPPORTED	Refer to Reader Protocol Specification 1.1
0003	ERROR_PARAMEER_INVALID_FORMAT	Refer to Reader Protocol Specification 1.1
0004	ERROR_PARAMETER_MISSING	Refer to Reader Protocol Specification 1.1

Error Code (Hex)	Error Name	Description
0009	ERROR_PARAMETER_LENGTH_EXCEEDED	Refer to Reader Protocol Specification 1.1
000D	ERROR_TRIGGER_NOT_FOUND	Refer to Reader Protocol Specification 1.1
000E	ERROR_READPOINT_NOT_FOUND	Refer to Reader Protocol Specification 1.1
0010	ERROR_SOURCE_NOT_FOUND	Raised when the given Source parameter is not known
1000	ERROR_IOPORT_NOT_FOUND	Raised when the given IOPort parameter is not known
1001	ERROR_ALARM_CHANNEL_NOT_FOUND	Raised when the given AlarmChannel parameter is not known
1002	ERROR_INVALID_ADMIN_STATUS	Raised when the specified Administrative status is not allowed for the given state of the object
1003	ERROR_TOO_MANY_ALARM_CHANNELS	Raised when the command failed due to an internal resource restriction. For instance, not enough memory, maximum number of objects reached, etc.
1004	ERROR_AUTHORIZATION	Raised when the command failed due to lack of rights to perform the operation on the object.

Table 1: Basic Reader Management Error Codes

3681

3682

3683 **9 Vendor Extensions**

3684 Vendors SHALL use the error conditions defined above for the commands of the
3685 standard command set. Vendors MAY add additional information on the exact cause of
3686 the error.

3687 Vendors can extend the standard Reader Management command set. These extension
3688 commands MAY use the same general error conditions as defined above, however they
3689 MAY also define additional error conditions. These vendor-defined error conditions
3690 SHALL use names starting with <VENDOR>_ERROR..., for example
3691 ACME_ERROR_WHATSIT_NOT_ADDED_NO_FLABOOZLE if Acme Corp. needed an
3692 error condition indicating there was no WHATSIT associated with a FLABOOZLE.

3693

3694 Vendor extensions may be:

- 3695 • Additional commands defined for the following objects:

3696 AlarmChannel
3697 AlarmControl
3698 AntennaReadPoint
3699 EdgeTriggeredAlarmControl
3700 IOPort
3701
3702 NotificationChannel
3703 ReaderDevice
3704 ReadPoint
3705 Source
3706
3707 Trigger
3708 TTOperationalStatusAlarmControl
3709

- 3710 • Additional attributes defined for the following objects:

3711 Alarm
3712
3713 FailedKillAlarm
3714 FailedLockAlarm
3715 FailedMemReadAlarm
3716
3717 FailedWriteAlarm
3718 FreeMemoryAlarm
3719 IOPortOperStatusAlarm
3720 NotificationChannelOperStatusAlarm
3721 OperStatusAlarm
3722 ReaderDeviceOperStatusAlarm
3723 ReadPointOperStatusAlarm
3724 SourceOperStatusAlarm

3725

3726 • Commands to additional objects.

3727

3728 • Additional types of Alarm (i.e. additional subclasses of Alarm).

3729

3730 The exact syntax for expressing vendor extensions is detailed for each Message/Transport
3731 Binding (MTB).

3732

3733

3734 **10 Message/Transport Bindings (MTBs)**

3735 The previous section defined an Abstract Command Set independent of message format
3736 or communication transport. This section defines the actual mapping of the Command Set
3737 to the specific Message Formats defined by the Reader Protocol Version 1.0 and SNMP.
3738 Each message format is independent of the other. A compliant application SHALL
3739 implement at least one of the Message Formats and MAY implement more than one of
3740 the message formats. If a particular Message Format is implemented, the compliant
3741 application SHALL implement all of the operations labeled SHALL. If an operation is
3742 not supported, the Reader and/ or Host SHALL reply with ERROR_NOT_SUPPORTED.

3743 Currently, two message formats are specified, XML, and SNMP. The XML message
3744 format can be used over any of the 3 transports (Serial, TCP, and HTTP) defined by the
3745 Reader Protocol Specification 1.1. In the future, additional message formats and
3746 transports may be defined.

3747 **10.1 Address Notation**

3748 All transports support a URI naming convention when an address is required by the
3749 command set, e.g. `NotificationChannel.setAddress()`. This uses the
3750 following notation except where explicitly stated otherwise:

3751 `<transport>://<locator>`

3752 For example:

3753 `tcp://foo.bar.com:2000`

3754 `serial://com1:3`

3755 <http://myreaderisawebserver:7183>

3756

3757 Note 1: in the case of a serial connection, where multiple channels have to be multiplexed
3758 over a single connection, the 'port' is a number that identifies the channel.

3759 Within the SNMP MIB, addresses are defined in compliance with RFC 3291/RFC 4001,
3760 the INET-ADDRESS MIB. This allows for addressing via IPv4, IPv6, or DNS Fully
3761 Qualified Domain Name.

3762 **10.2 Vendor Extension Details**

3763 Refer to the Reader Protocol 1.1 Vendor Extensions for the syntax used to define new
3764 Commands, Errors, and Attributes using the XML Message format.

3765 Vendor Extensions for the SNMP binding are implemented through the use of a vendor
3766 specific MIB. The Vendor specific MIB may reference the EPCglobal Reader
3767 Management MIB.

3768 The management application needs to know a priori the vendor extensions
3769 applicable to a reader. The XML schema provided by the reader vendor defines the

3770 operations (standard and extensions). SNMP provides for extensions by defining a
3771 separate MIB applicable to the reader model/vendor.

3772

3773 10.3 XML Message Format

3774 Refer to the Reader Protocol Version 1.1 Specification for the XML Binding rules,
3775 Object identification Scheme and Data Types. The following sections provide the Reader
3776 Management specific schema definitions.

3777 10.3.1 Command XML Message Encoding (Host-To-Reader)

```
3778 <?xml version="1.0" encoding="UTF-8"?>
3779 <!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Gerhard Gangl (7iD) -->
3780 <!-- edited with XMLSpy v2006 U (http://www.altova.com) by munusamy (none) -->
3781 <xsd:schema targetNamespace="urn:epcglobal:rm:xsd:1" elementFormDefault="unqualified"
3782 attributeFormDefault="unqualified" version="1.0" xmlns:rm="urn:epcglobal:rm:xsd:1"
3783 xmlns:epcglobal="urn:epcglobal:xsd:1" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3784   <xsd:annotation>
3785     <xsd:documentation xml:lang="en">
3786       <epcglobal:copyright>Copyright&#169;2005-2006 Epcglobal Inc., All Rights Reserved.</epcglobal:copyright>
3787       <epcglobal:disclaimer>EPCglobal Inc., its members, officers, directors, employees, or
3788         agents shall not be liable for any injury, loss, damages, financial or otherwise,
3789         arising from, related to, or caused by the use of this document. The use of said
3790         document shall constitute your express consent to the foregoing exculpation.</epcglobal:disclaimer>
3791       <epcglobal:specification>Reader Management (RM) version 1.0</epcglobal:specification>
3792     </xsd:documentation>
3793   </xsd:annotation>
3794   <xsd:include schemaLocation="RmCommon.xsd"/>
3795   <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="..EpcGlobal.xsd"/>
3796   <!-- Reader Management Command -->
3797   <xsd:element name="command">
3798     <xsd:annotation>
3799       <xsd:documentation xml:lang="en"> This element defines a reader management command.
3800       The commands are grouped by the object they belong to. To specify the object a command relates to,
3801       the name of the object is used. Therefore, all commands that return the name of the object
3802       become useless, since the name must be known before it can be queried. All the names of a
3803       particular object type are global irrespective of where they belong to. Hence names need to
3804       be unique within their object type. All the commands are encoded by its straight forward
3805       conversion from UML notation. Each command is wrapped in its object type and contains its
3806       parameters as child elements. </xsd:documentation>
3807     </xsd:annotation>
3808     <xsd:complexType>
3809       <xsd:complexContent>
3810         <xsd:extension base="epcglobal:Document">
3811           <xsd:sequence>
3812             <xsd:element name="id" type="xsd:string">
3813               <xsd:annotation>
3814                 <xsd:documentation xml:lang="en">The id of the command</xsd:documentation>
3815               </xsd:annotation>
3816             </xsd:element>
3817             <xsd:element name="targetName" type="xsd:string" minOccurs="0">
3818               <xsd:annotation>
```

```

3819         <xsd:documentation xml:lang="en">The name of the object on which this command should be
3820         executed. It can be left out for static methods/ or commands targeted to ReaderDevice object</xsd:documentation>
3821     </xsd:annotation>
3822 </xsd:element>
3823 <xsd:choice>
3824     <!-- Reader Management (Target) Object Types -->
3825     <!-- elements inside are ordered in alphabetical order -->
3826     <xsd:element name="alarmChannel" type="rm:AlarmChannelCommand">
3827         <xsd:annotation>
3828             <xsd:documentation xml:lang="en">An Alarm channel</xsd:documentation>
3829         </xsd:annotation>
3830     </xsd:element>
3831     <xsd:element name="alarmControl" type="rm:AlarmControlCommand">
3832         <xsd:annotation>
3833             <xsd:documentation xml:lang="en">AlarmControl is the base class for all of classes
3834             within the RMP object model responsible for controlling the generation of alarm messages within an EPCglobal
3835             compliant Reader.</xsd:documentation>
3836         </xsd:annotation>
3837     </xsd:element>
3838     <xsd:element name="antennaReadPoint" type="rm:AntennaReadPointCommand">
3839         <xsd:annotation>
3840             <xsd:documentation xml:lang="en">Extension of Read point object for
3841             antenna.</xsd:documentation>
3842         </xsd:annotation>
3843     </xsd:element>
3844     <xsd:element name="edgeTriggeredAlarmControl" type="rm:EdgeTriggeredAlarmControlCommand">
3845         <xsd:annotation>
3846             <xsd:documentation xml:lang="en">This class extends AlarmControl to control alarms
3847             generated when a monitored, integer-valued, model element first crosses a threshold value (the
3848             AlarmThreshold).</xsd:documentation>
3849         </xsd:annotation>
3850     </xsd:element>
3851     <xsd:element name="iOPort" type="rm:IOPortCommand">
3852         <xsd:annotation>
3853             <xsd:documentation xml:lang="en">The hardware element that provides external input
3854             and output lines to connect to other components outside the reader device.</xsd:documentation>
3855         </xsd:annotation>
3856     </xsd:element>
3857     <xsd:element name="notificationChannel" type="rm:NotificationChannelCommand">
3858         <xsd:annotation>
3859             <xsd:documentation xml:lang="en">The notification channel carries messages
3860             issued asynchronously by the Reader to the Host.</xsd:documentation>
3861         </xsd:annotation>
3862     </xsd:element>
3863     <xsd:element name="readerDevice" type="rm:ReaderDeviceCommand">
3864         <xsd:annotation>
3865             <xsd:documentation xml:lang="en">A Reader</xsd:documentation>
3866         </xsd:annotation>
3867     </xsd:element>
3868     <xsd:element name="readPoint" type="rm:ReadPointCommand">
3869         <xsd:annotation>
3870             <xsd:documentation xml:lang="en">A ReadPoint</xsd:documentation>
3871         </xsd:annotation>
3872     </xsd:element>
3873     <xsd:element name="source" type="rm:SourceCommand">
3874         <xsd:annotation>

```

```

3875         <xsd:documentation xml:lang="en">A Read source</xsd:documentation>
3876     </xsd:annotation>
3877 </xsd:element>
3878 <xsd:element name="trigger" type="rm:TriggerCommand">
3879     <xsd:annotation>
3880         <xsd:documentation xml:lang="en">A Read/Notify Trigger</xsd:documentation>
3881     </xsd:annotation>
3882 </xsd:element>
3883 <xsd:element name="tTOperationalStatusAlarmControl"
3884 type="rm:TTOperationalStatusAlarmControlCommand">
3885     <xsd:annotation>
3886         <xsd:documentation xml:lang="en">This class extends AlarmControl to control alarms
3887 generated when a monitored model element of type OperationalStatus transitions to a new
3888 value.</xsd:documentation>
3889     </xsd:annotation>
3890 </xsd:element>
3891 <xsd:any namespace="##any" processContents="lax">
3892     <xsd:annotation>
3893         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
3894     </xsd:annotation>
3895 </xsd:any>
3896 </xsd:choice>
3897 </xsd:sequence>
3898 </xsd:extension>
3899 </xsd:complexContent>
3900 </xsd:complexType>
3901 </xsd:element>
3902 <!-- Reader Management (Target) Object Types -->
3903 <!-- types are ordered in alphabetical order -->
3904 <!-- inside each type commands are listed in the order in which it is defined in the RM specification. -->
3905 <xsd:complexType name="AlarmChannelCommand">
3906     <xsd:choice>
3907         <xsd:annotation>
3908             <xsd:documentation xml:lang="en"> Alarm channel object commands.</xsd:documentation>
3909         </xsd:annotation>
3910         <xsd:element name="create">
3911             <xsd:annotation>
3912                 <xsd:documentation xml:lang="en">static method to create an Alarm channel. </xsd:documentation>
3913             </xsd:annotation>
3914             <xsd:complexType>
3915                 <xsd:sequence>
3916                     <xsd:element name="name" type="xsd:string">
3917                         <xsd:annotation>
3918                             <xsd:documentation xml:lang="en"> Alarm channel name </xsd:documentation>
3919                         </xsd:annotation>
3920                     </xsd:element>
3921                     <xsd:element name="addr" type="rm:AddressParamType">
3922                         <xsd:annotation>
3923                             <xsd:documentation xml:lang="en"> The (host) address of where alarms will be sent
3924 to.</xsd:documentation>
3925                         </xsd:annotation>
3926                     </xsd:element>
3927                 </xsd:sequence>
3928             </xsd:complexType>
3929         </xsd:element>
3930         <xsd:element name="getName">

```

```

3931     <xsd:annotation>
3932         <xsd:documentation xml:lang="en">get the alarm channel name</xsd:documentation>
3933     </xsd:annotation>
3934     <xsd:complexType/>
3935 </xsd:element>
3936 <xsd:element name="getAddress">
3937     <xsd:annotation>
3938         <xsd:documentation xml:lang="en">get the (host) address to which this AlarmChannel object sends its
3939 alarms.</xsd:documentation>
3940     </xsd:annotation>
3941     <xsd:complexType/>
3942 </xsd:element>
3943 <xsd:element name="setAddress">
3944     <xsd:annotation>
3945         <xsd:documentation xml:lang="en">set the (host) address to which this AlarmChannel object sends its
3946 alarms.</xsd:documentation>
3947     </xsd:annotation>
3948     <xsd:complexType>
3949         <xsd:sequence>
3950             <xsd:element name="addr" type="rm:AddressParamType">
3951                 <xsd:annotation>
3952                     <xsd:documentation xml:lang="en"> address to set.</xsd:documentation>
3953                 </xsd:annotation>
3954             </xsd:element>
3955         </xsd:sequence>
3956     </xsd:complexType>
3957 </xsd:element>
3958 <xsd:any namespace="##any" processContents="lax">
3959     <xsd:annotation>
3960         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
3961     </xsd:annotation>
3962 </xsd:any>
3963 </xsd:choice>
3964 </xsd:complexType>
3965 <xsd:complexType name="AlarmControlCommand">
3966     <xsd:choice>
3967         <xsd:annotation>
3968             <xsd:documentation xml:lang="en"> Alarm control object commands. </xsd:documentation>
3969         </xsd:annotation>
3970         <xsd:element name="getName">
3971             <xsd:annotation>
3972                 <xsd:documentation xml:lang="en">get the alarm control name</xsd:documentation>
3973             </xsd:annotation>
3974             <xsd:complexType/>
3975         </xsd:element>
3976         <xsd:element name="getEnabled">
3977             <xsd:annotation>
3978                 <xsd:documentation xml:lang="en">query the reader for the current value of the AlarmControlâ€™s
3979 Enabled attribute.</xsd:documentation>
3980             </xsd:annotation>
3981             <xsd:complexType/>
3982         </xsd:element>
3983         <xsd:element name="setEnabled">
3984             <xsd:annotation>
3985                 <xsd:documentation xml:lang="en">enable or disable alarm generation.</xsd:documentation>
3986             </xsd:annotation>

```

```

3987     <xsd:complexType>
3988     <xsd:sequence>
3989         <xsd:element name="enable" type="xsd:boolean">
3990             <xsd:annotation>
3991                 <xsd:documentation xml:lang="en">whether to enable the alarm generation.</xsd:documentation>
3992             </xsd:annotation>
3993         </xsd:element>
3994     </xsd:sequence>
3995 </xsd:complexType>
3996 </xsd:element>
3997 <xsd:element name="getLevel">
3998     <xsd:annotation>
3999         <xsd:documentation xml:lang="en">query the reader for the current value of the AlarmControl™s Level
4000 attribute.</xsd:documentation>
4001     </xsd:annotation>
4002 </xsd:complexType/>
4003 </xsd:element>
4004 <xsd:element name="setLevel">
4005     <xsd:annotation>
4006         <xsd:documentation xml:lang="en">set the Level attribute.</xsd:documentation>
4007     </xsd:annotation>
4008 </xsd:complexType>
4009     <xsd:sequence>
4010         <xsd:element name="alarmLevel" type="rm:AlarmLevelParamType">
4011             <xsd:annotation>
4012                 <xsd:documentation xml:lang="en">Alarm level to set</xsd:documentation>
4013             </xsd:annotation>
4014         </xsd:element>
4015     </xsd:sequence>
4016 </xsd:complexType>
4017 </xsd:element>
4018 <xsd:element name="getSuppressInterval">
4019     <xsd:annotation>
4020         <xsd:documentation xml:lang="en">query the reader for the current value of the AlarmControl™s
4021 SuppressInterval attribute.</xsd:documentation>
4022     </xsd:annotation>
4023 </xsd:complexType/>
4024 </xsd:element>
4025 <xsd:element name="setSuppressInterval">
4026     <xsd:annotation>
4027         <xsd:documentation xml:lang="en">set the SuppressInterval attribute.</xsd:documentation>
4028     </xsd:annotation>
4029 </xsd:complexType>
4030     <xsd:sequence>
4031         <xsd:element name="suppressInterval" type="xsd:int">
4032             <xsd:annotation>
4033                 <xsd:documentation xml:lang="en">SuppressInterval to set in seconds</xsd:documentation>
4034             </xsd:annotation>
4035         </xsd:element>
4036     </xsd:sequence>
4037 </xsd:complexType>
4038 </xsd:element>
4039 <xsd:any namespace="##any" processContents="lax">
4040     <xsd:annotation>
4041         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4042     </xsd:annotation>

```

```

4043     </xsd:any>
4044   </xsd:choice>
4045 </xsd:complexType>
4046 <xsd:complexType name="AntennaReadPointCommand">
4047   <xsd:choice>
4048     <xsd:annotation>
4049       <xsd:documentation xml:lang="en"> Antenna read point object commands </xsd:documentation>
4050     </xsd:annotation>
4051     <xsd:element name="getIdentificationCount">
4052       <xsd:annotation>
4053         <xsd:documentation xml:lang="en">get the number of the successful tags that have been
4054           identified across an AntennaReadPoint. </xsd:documentation>
4055       </xsd:annotation>
4056     </xsd:complexType>
4057   </xsd:element>
4058   <xsd:element name="getFailedIdentificationCount">
4059     <xsd:annotation>
4060       <xsd:documentation xml:lang="en">get the number of the failed tag identification attempts at the
4061 AntennaReadPoint. </xsd:documentation>
4062     </xsd:annotation>
4063   </xsd:complexType>
4064 </xsd:element>
4065   <xsd:element name="getMemReadCount">
4066     <xsd:annotation>
4067       <xsd:documentation xml:lang="en">get the number of tag memory reads at the AntennaReadPoint.
4068 </xsd:documentation>
4069     </xsd:annotation>
4070   </xsd:complexType>
4071 </xsd:element>
4072   <xsd:element name="getFailedMemReadCount">
4073     <xsd:annotation>
4074       <xsd:documentation xml:lang="en">get the number of the failed tag memory reads at the
4075 AntennaReadPoint. </xsd:documentation>
4076     </xsd:annotation>
4077   </xsd:complexType>
4078 </xsd:element>
4079   <xsd:element name="getFailedMemReadAlarmControl">
4080     <xsd:annotation>
4081       <xsd:documentation xml:lang="en">get the AntennaReadPoint's failed memory read alarm
4082 control.</xsd:documentation>
4083     </xsd:annotation>
4084   </xsd:complexType>
4085 </xsd:element>
4086   <xsd:element name="getWriteCount">
4087     <xsd:annotation>
4088       <xsd:documentation xml:lang="en">get the number of successful tag writes at the AntennaReadPoint.
4089 </xsd:documentation>
4090     </xsd:annotation>
4091   </xsd:complexType>
4092 </xsd:element>
4093   <xsd:element name="getFailedWriteCount">
4094     <xsd:annotation>
4095       <xsd:documentation xml:lang="en">get the number of the failed attempts to write tags at the
4096 AntennaReadPoint. </xsd:documentation>
4097     </xsd:annotation>
4098   </xsd:complexType>

```

```

4099     </xsd:element>
4100     <xsd:element name="getFailedWriteAlarmControl">
4101         <xsd:annotation>
4102             <xsd:documentation xml:lang="en">get the AntennaReadPoint's failed write alarm
4103 control.</xsd:documentation>
4104         </xsd:annotation>
4105         <xsd:complexType/>
4106     </xsd:element>
4107     <xsd:element name="getKillCount">
4108         <xsd:annotation>
4109             <xsd:documentation xml:lang="en">get the number of tags successfully killed at the AntennaReadPoint.
4110 </xsd:documentation>
4111         </xsd:annotation>
4112         <xsd:complexType/>
4113     </xsd:element>
4114     <xsd:element name="getFailedKillCount">
4115         <xsd:annotation>
4116             <xsd:documentation xml:lang="en">get the number of the failed tag kills at the AntennaReadPoint.
4117 </xsd:documentation>
4118         </xsd:annotation>
4119         <xsd:complexType/>
4120     </xsd:element>
4121     <xsd:element name="getFailedKillAlarmControl">
4122         <xsd:annotation>
4123             <xsd:documentation xml:lang="en">get the AntennaReadPoint's failed kill alarm
4124 control.</xsd:documentation>
4125         </xsd:annotation>
4126         <xsd:complexType/>
4127     </xsd:element>
4128     <xsd:element name="getEraseCount">
4129         <xsd:annotation>
4130             <xsd:documentation xml:lang="en">get the number of tags successfully erased at the
4131 AntennaReadPoint.</xsd:documentation>
4132         </xsd:annotation>
4133         <xsd:complexType/>
4134     </xsd:element>
4135     <xsd:element name="getFailedEraseCount">
4136         <xsd:annotation>
4137             <xsd:documentation xml:lang="en">get the number of the failed tag erasures at the AntennaReadPoint.
4138 </xsd:documentation>
4139         </xsd:annotation>
4140         <xsd:complexType/>
4141     </xsd:element>
4142     <xsd:element name="getFailedEraseAlarmControl">
4143         <xsd:annotation>
4144             <xsd:documentation xml:lang="en">get the AntennaReadPoint's failed erase alarm
4145 control.</xsd:documentation>
4146         </xsd:annotation>
4147         <xsd:complexType/>
4148     </xsd:element>
4149     <xsd:element name="getLockCount">
4150         <xsd:annotation>
4151             <xsd:documentation xml:lang="en">get the number of tags successfully locked at the AntennaReadPoint.
4152 </xsd:documentation>
4153         </xsd:annotation>
4154         <xsd:complexType/>

```

```

4155     </xsd:element>
4156     <xsd:element name="getFailedLockCount">
4157         <xsd:annotation>
4158             <xsd:documentation xml:lang="en">get the number of the failed tag locks at the AntennaReadPoint.
4159 </xsd:documentation>
4160         </xsd:annotation>
4161         <xsd:complexType/>
4162     </xsd:element>
4163     <xsd:element name="getFailedLockAlarmControl">
4164         <xsd:annotation>
4165             <xsd:documentation xml:lang="en">get the AntennaReadPoint's failed lock alarm
4166 control.</xsd:documentation>
4167         </xsd:annotation>
4168         <xsd:complexType/>
4169     </xsd:element>
4170     <xsd:element name="getTimeEnergized">
4171         <xsd:annotation>
4172             <xsd:documentation xml:lang="en">get the number of milliseconds the AntennaReadPoint has been
4173 energized in order to communicate with tags.</xsd:documentation>
4174         </xsd:annotation>
4175         <xsd:complexType/>
4176     </xsd:element>
4177     <xsd:element name="getPowerLevel">
4178         <xsd:annotation>
4179             <xsd:documentation xml:lang="en">get the current transmit power level.</xsd:documentation>
4180         </xsd:annotation>
4181         <xsd:complexType/>
4182     </xsd:element>
4183     <xsd:element name="getNoiseLevel">
4184         <xsd:annotation>
4185             <xsd:documentation xml:lang="en">get the current noise level.</xsd:documentation>
4186         </xsd:annotation>
4187         <xsd:complexType/>
4188     </xsd:element>
4189     <xsd:any namespace="##any" processContents="lax">
4190         <xsd:annotation>
4191             <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4192         </xsd:annotation>
4193     </xsd:any>
4194 </xsd:choice>
4195 </xsd:complexType>
4196 <xsd:complexType name="EdgeTriggeredAlarmControlCommand">
4197     <xsd:choice>
4198         <xsd:annotation>
4199             <xsd:documentation xml:lang="en"> Edge triggered alarm control object commands. </xsd:documentation>
4200         </xsd:annotation>
4201         <xsd:element name="getAlarmThreshold">
4202             <xsd:annotation>
4203                 <xsd:documentation xml:lang="en">query the reader for the current value of the
4204 EdgeTriggeredAlarmControl's AlarmThreshold attribute.</xsd:documentation>
4205             </xsd:annotation>
4206             <xsd:complexType/>
4207         </xsd:element>
4208         <xsd:element name="setAlarmThreshold">
4209             <xsd:annotation>

```

```

4210      <xsd:documentation xml:lang="en">set the current value of the EdgeTriggeredAlarmControlâ€™s
4211 AlarmThreshold attribute.</xsd:documentation>
4212    </xsd:annotation>
4213    <xsd:complexType>
4214      <xsd:sequence>
4215        <xsd:element name="alarmThreshold" type="xsd:int">
4216          <xsd:annotation>
4217            <xsd:documentation xml:lang="en">threshold to set</xsd:documentation>
4218          </xsd:annotation>
4219        </xsd:element>
4220      </xsd:sequence>
4221    </xsd:complexType>
4222  </xsd:element>
4223  <xsd:element name="getRearmThreshold">
4224    <xsd:annotation>
4225      <xsd:documentation xml:lang="en">query the reader for the current value of the
4226 EdgeTriggeredAlarmControlâ€™s RearmThreshold attribute.</xsd:documentation>
4227    </xsd:annotation>
4228    <xsd:complexType/>
4229  </xsd:element>
4230  <xsd:element name="setRearmThreshold">
4231    <xsd:annotation>
4232      <xsd:documentation xml:lang="en">set the current value of the RearmThreshold
4233 attribute.</xsd:documentation>
4234    </xsd:annotation>
4235    <xsd:complexType>
4236      <xsd:sequence>
4237        <xsd:element name="rearmThreshold" type="xsd:int">
4238          <xsd:annotation>
4239            <xsd:documentation xml:lang="en">threshold to set</xsd:documentation>
4240          </xsd:annotation>
4241        </xsd:element>
4242      </xsd:sequence>
4243    </xsd:complexType>
4244  </xsd:element>
4245  <xsd:element name="getDirection">
4246    <xsd:annotation>
4247      <xsd:documentation xml:lang="en">query the reader for the current value of the
4248 EdgeTriggeredAlarmControlâ€™s Direction attribute.</xsd:documentation>
4249    </xsd:annotation>
4250    <xsd:complexType/>
4251  </xsd:element>
4252  <xsd:element name="setDirection">
4253    <xsd:annotation>
4254      <xsd:documentation xml:lang="en">set the current value of the EdgeTriggeredAlarmControlâ€™s Direction
4255 attribute.</xsd:documentation>
4256    </xsd:annotation>
4257    <xsd:complexType>
4258      <xsd:sequence>
4259        <xsd:element name="direction" type="rm:EdgeTriggeredAlarmDirectionParamType">
4260          <xsd:annotation>
4261            <xsd:documentation xml:lang="en">direction to set</xsd:documentation>
4262          </xsd:annotation>
4263        </xsd:element>
4264      </xsd:sequence>
4265    </xsd:complexType>

```

```

4266     </xsd:element>
4267     <xsd:element name="getStatus">
4268         <xsd:annotation>
4269             <xsd:documentation xml:lang="en">query the reader for the current value of the
4270 EdgeTriggeredAlarmControlâ€™s Status attribute.</xsd:documentation>
4271         </xsd:annotation>
4272         <xsd:complexType/>
4273     </xsd:element>
4274     <xsd:any namespace="##any" processContents="lax">
4275         <xsd:annotation>
4276             <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4277         </xsd:annotation>
4278     </xsd:any>
4279 </xsd:choice>
4280 </xsd:complexType>
4281 <xsd:complexType name="IOPortCommand">
4282     <xsd:choice>
4283         <xsd:annotation>
4284             <xsd:documentation xml:lang="en">IO port object commands. </xsd:documentation>
4285         </xsd:annotation>
4286         <xsd:element name="getName">
4287             <xsd:annotation>
4288                 <xsd:documentation xml:lang="en">get the IO port name</xsd:documentation>
4289             </xsd:annotation>
4290             <xsd:complexType/>
4291         </xsd:element>
4292         <xsd:element name="getDescription">
4293             <xsd:annotation>
4294                 <xsd:documentation xml:lang="en">get a textual description of the IO-port.</xsd:documentation>
4295             </xsd:annotation>
4296             <xsd:complexType/>
4297         </xsd:element>
4298         <xsd:element name="setDescription">
4299             <xsd:annotation>
4300                 <xsd:documentation xml:lang="en">set (associate) a textual description with an IO-
4301 port.</xsd:documentation>
4302             </xsd:annotation>
4303             <xsd:complexType>
4304                 <xsd:sequence>
4305                     <xsd:element name="description" type="xsd:string">
4306                         <xsd:annotation>
4307                             <xsd:documentation xml:lang="en">description to set</xsd:documentation>
4308                         </xsd:annotation>
4309                     </xsd:element>
4310                 </xsd:sequence>
4311             </xsd:complexType>
4312         </xsd:element>
4313         <xsd:element name="getOperStatus">
4314             <xsd:annotation>
4315                 <xsd:documentation xml:lang="en">query for the operational status of an IO-port.</xsd:documentation>
4316             </xsd:annotation>
4317             <xsd:complexType/>
4318         </xsd:element>
4319         <xsd:element name="getAdminStatus">
4320             <xsd:annotation>
4321                 <xsd:documentation xml:lang="en">query for the administrative status of an IO-port.</xsd:documentation>

```

```

4322     </xsd:annotation>
4323     <xsd:complexType/>
4324 </xsd:element>
4325 <xsd:element name="setAdminStatus">
4326     <xsd:annotation>
4327         <xsd:documentation xml:lang="en">set the administrative status of an IO-port.</xsd:documentation>
4328     </xsd:annotation>
4329     <xsd:complexType>
4330         <xsd:sequence>
4331             <xsd:element name="administrativeStatus" type="rm:AdministrativeStatusParamType">
4332                 <xsd:annotation>
4333                     <xsd:documentation xml:lang="en">Administrative status to set</xsd:documentation>
4334                 </xsd:annotation>
4335             </xsd:element>
4336         </xsd:sequence>
4337     </xsd:complexType>
4338 </xsd:element>
4339 <xsd:element name="getOperStatusAlarmControl">
4340     <xsd:annotation>
4341         <xsd:documentation xml:lang="en">get the IOPort's operational status alarm
4342 control.</xsd:documentation>
4343     </xsd:annotation>
4344     <xsd:complexType/>
4345 </xsd:element>
4346 <xsd:any namespace="##any" processContents="lax">
4347     <xsd:annotation>
4348         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4349     </xsd:annotation>
4350 </xsd:any>
4351 </xsd:choice>
4352 </xsd:complexType>
4353 <xsd:complexType name="NotificationChannelCommand">
4354     <xsd:choice>
4355         <xsd:annotation>
4356             <xsd:documentation xml:lang="en">Notification channel object commands</xsd:documentation>
4357         </xsd:annotation>
4358         <xsd:element name="getLastNotificationAttempt">
4359             <xsd:annotation>
4360                 <xsd:documentation xml:lang="en">get the timestamp (TimeTicks) when the last attempt was made to
4361 send a notification to the given address.</xsd:documentation>
4362             </xsd:annotation>
4363             <xsd:complexType/>
4364         </xsd:element>
4365         <xsd:element name="getLastSuccessfulNotification">
4366             <xsd:annotation>
4367                 <xsd:documentation xml:lang="en">get the timestamp (TimeTicks) when the last successful notification
4368 was send to the given address.</xsd:documentation>
4369             </xsd:annotation>
4370             <xsd:complexType/>
4371         </xsd:element>
4372         <xsd:element name="getOperStatus">
4373             <xsd:annotation>
4374                 <xsd:documentation xml:lang="en">query the NotificationChannel for its OperationalStatus
4375 object.</xsd:documentation>
4376             </xsd:annotation>
4377             <xsd:complexType/>

```

```

4378     </xsd:element>
4379     <xsd:element name="setAdminStatus">
4380         <xsd:annotation>
4381             <xsd:documentation xml:lang="en">set the administrative status of Notification
4382 channel.</xsd:documentation>
4383         </xsd:annotation>
4384         <xsd:complexType>
4385             <xsd:sequence>
4386                 <xsd:element name="administrativeStatus" type="rm:AdministrativeStatusParamType">
4387                     <xsd:annotation>
4388                         <xsd:documentation xml:lang="en"> administrative status to set. </xsd:documentation>
4389                     </xsd:annotation>
4390                 </xsd:element>
4391             </xsd:sequence>
4392         </xsd:complexType>
4393     </xsd:element>
4394     <xsd:element name="getAdminStatus">
4395         <xsd:annotation>
4396             <xsd:documentation xml:lang="en">query the NotificationChannel for its administrative
4397 status.</xsd:documentation>
4398         </xsd:annotation>
4399         <xsd:complexType/>
4400     </xsd:element>
4401     <xsd:element name="getOperStatusAlarmControl">
4402         <xsd:annotation>
4403             <xsd:documentation xml:lang="en">query the notification channel for its operational status alarm
4404 control.</xsd:documentation>
4405         </xsd:annotation>
4406         <xsd:complexType/>
4407     </xsd:element>
4408     <xsd:any namespace="##any" processContents="lax">
4409         <xsd:annotation>
4410             <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4411         </xsd:annotation>
4412     </xsd:any>
4413 </xsd:choice>
4414 </xsd:complexType>
4415 <xsd:complexType name="ReaderDeviceCommand">
4416     <xsd:annotation>
4417         <xsd:documentation xml:lang="en">Reader Device object commands. </xsd:documentation>
4418     </xsd:annotation>
4419     <xsd:choice>
4420         <xsd:element name="getDescription">
4421             <xsd:annotation>
4422                 <xsd:documentation xml:lang="en">query the Reader for its user defined
4423 description.</xsd:documentation>
4424             </xsd:annotation>
4425             <xsd:complexType/>
4426         </xsd:element>
4427         <xsd:element name="setDescription">
4428             <xsd:annotation>
4429                 <xsd:documentation xml:lang="en">set the reader's user-defined description.</xsd:documentation>
4430             </xsd:annotation>
4431             <xsd:complexType>
4432                 <xsd:sequence>
4433                     <xsd:element name="description" type="xsd:string">

```

```

4434         <xsd:annotation>
4435             <xsd:documentation xml:lang="en"> description to set</xsd:documentation>
4436         </xsd:annotation>
4437     </xsd:element>
4438 </xsd:sequence>
4439 </xsd:complexType>
4440 </xsd:element>
4441 <xsd:element name="getLocationDescription">
4442     <xsd:annotation>
4443         <xsd:documentation xml:lang="en">query the reader for its user defined location
4444 description.</xsd:documentation>
4445     </xsd:annotation>
4446     <xsd:complexType/>
4447 </xsd:element>
4448 <xsd:element name="setLocationDescription">
4449     <xsd:annotation>
4450         <xsd:documentation xml:lang="en">set the reader's user-defined location
4451 description.</xsd:documentation>
4452     </xsd:annotation>
4453     <xsd:complexType>
4454         <xsd:sequence>
4455             <xsd:element name="locationDescription" type="xsd:string">
4456                 <xsd:annotation>
4457                     <xsd:documentation xml:lang="en"> location description to set. </xsd:documentation>
4458                 </xsd:annotation>
4459             </xsd:element>
4460         </xsd:sequence>
4461     </xsd:complexType>
4462 </xsd:element>
4463 <xsd:element name="getContact">
4464     <xsd:annotation>
4465         <xsd:documentation xml:lang="en">query the reader for its user-defined contact
4466 description.</xsd:documentation>
4467     </xsd:annotation>
4468     <xsd:complexType/>
4469 </xsd:element>
4470 <xsd:element name="setContact">
4471     <xsd:annotation>
4472         <xsd:documentation xml:lang="en">set the reader's user-defined contact
4473 description.</xsd:documentation>
4474     </xsd:annotation>
4475     <xsd:complexType>
4476         <xsd:sequence>
4477             <xsd:element name="contact" type="xsd:string">
4478                 <xsd:annotation>
4479                     <xsd:documentation xml:lang="en"> contact to set</xsd:documentation>
4480                 </xsd:annotation>
4481             </xsd:element>
4482         </xsd:sequence>
4483     </xsd:complexType>
4484 </xsd:element>
4485 <xsd:element name="getSerialNumber">
4486     <xsd:annotation>
4487         <xsd:documentation xml:lang="en">query the reader for its serial number.</xsd:documentation>
4488     </xsd:annotation>
4489     <xsd:complexType/>

```

```

4490     </xsd:element>
4491     <xsd:element name="getOperStatus">
4492         <xsd:annotation>
4493             <xsd:documentation xml:lang="en">query the reader for its OperationalStatus
4494 object.</xsd:documentation>
4495         </xsd:annotation>
4496         <xsd:complexType/>
4497     </xsd:element>
4498     <xsd:element name="getOperStatusAlarmControl">
4499         <xsd:annotation>
4500             <xsd:documentation xml:lang="en">query the reader for its operational status alarm
4501 control.</xsd:documentation>
4502         </xsd:annotation>
4503         <xsd:complexType/>
4504     </xsd:element>
4505     <xsd:element name="getFreeMemory">
4506         <xsd:annotation>
4507             <xsd:documentation xml:lang="en">query the reader for its available free memory.</xsd:documentation>
4508         </xsd:annotation>
4509         <xsd:complexType/>
4510     </xsd:element>
4511     <xsd:element name="getFreeMemoryAlarmControl">
4512         <xsd:annotation>
4513             <xsd:documentation xml:lang="en">query the reader for its FreeMemoryAlarmControl
4514 object.</xsd:documentation>
4515         </xsd:annotation>
4516         <xsd:complexType/>
4517     </xsd:element>
4518     <xsd:element name="getNTPServers">
4519         <xsd:annotation>
4520             <xsd:documentation xml:lang="en">query the reader for a list of NTP servers used by it to synchronize its
4521 current UTC clock (TimeUTC).</xsd:documentation>
4522         </xsd:annotation>
4523         <xsd:complexType/>
4524     </xsd:element>
4525     <xsd:element name="getDHCPServer">
4526         <xsd:annotation>
4527             <xsd:documentation xml:lang="en">query the reader for the DHCP server currently used by the device for
4528 DHCP requests.</xsd:documentation>
4529         </xsd:annotation>
4530         <xsd:complexType/>
4531     </xsd:element>
4532     <xsd:element name="getIOPort">
4533         <xsd:annotation>
4534             <xsd:documentation xml:lang="en">get the IO Port with the specified name currently associated with this
4535 Reader.</xsd:documentation>
4536         </xsd:annotation>
4537         <xsd:complexType>
4538             <xsd:sequence>
4539                 <xsd:element name="name" type="xsd:string">
4540                     <xsd:annotation>
4541                         <xsd:documentation xml:lang="en">IO port name </xsd:documentation>
4542                     </xsd:annotation>
4543                 </xsd:element>
4544             </xsd:sequence>
4545         </xsd:complexType>

```

```

4546     </xsd:element>
4547     <xsd:element name="getAllIOPorts">
4548         <xsd:annotation>
4549             <xsd:documentation xml:lang="en">query the reader for all its IOPort objects.</xsd:documentation>
4550         </xsd:annotation>
4551         <xsd:complexType/>
4552     </xsd:element>
4553     <xsd:element name="resetStatistics">
4554         <xsd:annotation>
4555             <xsd:documentation xml:lang="en">reset reader's entire internal statistic counters to
4556 zero.</xsd:documentation>
4557         </xsd:annotation>
4558         <xsd:complexType/>
4559     </xsd:element>
4560     <xsd:element name="removeAlarmChannels">
4561         <xsd:annotation>
4562             <xsd:documentation xml:lang="en">remove the specified AlarmChannels from the list of AlarmChannels
4563 currently associated with the reader.</xsd:documentation>
4564         </xsd:annotation>
4565         <xsd:complexType>
4566             <xsd:sequence>
4567                 <xsd:element name="channels" type="rm:AlarmChannelListParamType">
4568                     <xsd:annotation>
4569                         <xsd:documentation xml:lang="en"> alarm channels to remove. </xsd:documentation>
4570                     </xsd:annotation>
4571                 </xsd:element>
4572             </xsd:sequence>
4573         </xsd:complexType>
4574     </xsd:element>
4575     <xsd:element name="removeAllAlarmChannels">
4576         <xsd:annotation>
4577             <xsd:documentation xml:lang="en">remove all AlarmChannels currently associated with the
4578 reader.</xsd:documentation>
4579         </xsd:annotation>
4580         <xsd:complexType/>
4581     </xsd:element>
4582     <xsd:element name="getAlarmChannel">
4583         <xsd:annotation>
4584             <xsd:documentation xml:lang="en">get the AlarmChannel with the specified name currently associated
4585 with the reader.</xsd:documentation>
4586         </xsd:annotation>
4587         <xsd:complexType>
4588             <xsd:sequence>
4589                 <xsd:element name="name" type="xsd:string">
4590                     <xsd:annotation>
4591                         <xsd:documentation xml:lang="en"> Alarm channel name </xsd:documentation>
4592                     </xsd:annotation>
4593                 </xsd:element>
4594             </xsd:sequence>
4595         </xsd:complexType>
4596     </xsd:element>
4597     <xsd:element name="getAllAlarmChannels">
4598         <xsd:annotation>
4599             <xsd:documentation xml:lang="en">get all AlarmChannels currently associated with the
4600 reader.</xsd:documentation>
4601         </xsd:annotation>

```

```

4602     <xsd:complexType/>
4603 </xsd:element>
4604 <xsd:any namespace="##any" processContents="lax">
4605     <xsd:annotation>
4606         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4607     </xsd:annotation>
4608 </xsd:any>
4609 </xsd:choice>
4610 </xsd:complexType>
4611 <xsd:complexType name="ReadPointCommand">
4612     <xsd:annotation>
4613         <xsd:documentation xml:lang="en"> Read point object commands.</xsd:documentation>
4614     </xsd:annotation>
4615 <xsd:choice>
4616     <xsd:element name="getClassName">
4617         <xsd:annotation>
4618             <xsd:documentation xml:lang="en">get the class name of the ReadPoint object. In the current
4619 specification, the only supported read point class is an "AntennaReadPoint".</xsd:documentation>
4620         </xsd:annotation>
4621         <xsd:complexType/>
4622     </xsd:element>
4623     <xsd:element name="getDescription">
4624         <xsd:annotation>
4625             <xsd:documentation xml:lang="en">get the description of the ReadPoint.</xsd:documentation>
4626         </xsd:annotation>
4627         <xsd:complexType/>
4628     </xsd:element>
4629     <xsd:element name="setDescription">
4630         <xsd:annotation>
4631             <xsd:documentation xml:lang="en">set the ReadPoint description.</xsd:documentation>
4632         </xsd:annotation>
4633         <xsd:complexType>
4634             <xsd:sequence>
4635                 <xsd:element name="description" type="xsd:string">
4636                     <xsd:annotation>
4637                         <xsd:documentation xml:lang="en"> description to set.</xsd:documentation>
4638                     </xsd:annotation>
4639                 </xsd:element>
4640             </xsd:sequence>
4641         </xsd:complexType>
4642     </xsd:element>
4643     <xsd:element name="getAdminStatus">
4644         <xsd:annotation>
4645             <xsd:documentation xml:lang="en">get the current ReadPoint administrative status.</xsd:documentation>
4646         </xsd:annotation>
4647         <xsd:complexType/>
4648     </xsd:element>
4649     <xsd:element name="setAdminStatus">
4650         <xsd:annotation>
4651             <xsd:documentation xml:lang="en">set the current ReadPoint administrative status.</xsd:documentation>
4652         </xsd:annotation>
4653         <xsd:complexType>
4654             <xsd:sequence>
4655                 <xsd:element name="status" type="rm:AdministrativeStatusParamType">
4656                     <xsd:annotation>
4657                         <xsd:documentation xml:lang="en"> administrative status to set. </xsd:documentation>

```

```

4658         </xsd:annotation>
4659     </xsd:element>
4660 </xsd:sequence>
4661 </xsd:complexType>
4662 </xsd:element>
4663 <xsd:element name="getOperStatus">
4664     <xsd:annotation>
4665         <xsd:documentation xml:lang="en">get the ReadPoint's current operational status.</xsd:documentation>
4666     </xsd:annotation>
4667     <xsd:complexType/>
4668 </xsd:element>
4669 <xsd:element name="getOperStatusAlarmControl">
4670     <xsd:annotation>
4671         <xsd:documentation xml:lang="en">get the ReadPoint's operational status alarm control.
4672 </xsd:documentation>
4673     </xsd:annotation>
4674     <xsd:complexType/>
4675 </xsd:element>
4676 <xsd:any namespace="##any" processContents="lax">
4677     <xsd:annotation>
4678         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4679     </xsd:annotation>
4680 </xsd:any>
4681 </xsd:choice>
4682 </xsd:complexType>
4683 <xsd:complexType name="SourceCommand">
4684     <xsd:choice>
4685         <xsd:annotation>
4686             <xsd:documentation xml:lang="en">Source object commands. </xsd:documentation>
4687         </xsd:annotation>
4688         <xsd:element name="getUnknownToGlimpsedCount">
4689             <xsd:annotation>
4690                 <xsd:documentation xml:lang="en">query the reader for the number of times a transition from state
4691 Unknown to state Glimpsed have been detected for the particular source.</xsd:documentation>
4692             </xsd:annotation>
4693             <xsd:complexType/>
4694 </xsd:element>
4695             <xsd:element name="getGlimpsedToUnknownCount">
4696                 <xsd:annotation>
4697                     <xsd:documentation xml:lang="en">query the reader for the number of times a transition from state
4698 Glimpsed to state Unknown have been detected for the particular source.</xsd:documentation>
4699                 </xsd:annotation>
4700                 <xsd:complexType/>
4701 </xsd:element>
4702             <xsd:element name="getGlimpsedToObservedCount">
4703                 <xsd:annotation>
4704                     <xsd:documentation xml:lang="en">query the reader for the number of times a transition from state
4705 Glimpsed to state Observed have been detected for the particular source.</xsd:documentation>
4706                 </xsd:annotation>
4707                 <xsd:complexType/>
4708 </xsd:element>
4709             <xsd:element name="getObservedToLostCount">
4710                 <xsd:annotation>
4711                     <xsd:documentation xml:lang="en">query the reader for the number of times a transition from state
4712 Observed to state Lost have been detected for the particular source.</xsd:documentation>
4713                 </xsd:annotation>

```

```

4714     <xsd:complexType/>
4715   </xsd:element>
4716   <xsd:element name="getLostToGlimpsedCount">
4717     <xsd:annotation>
4718       <xsd:documentation xml:lang="en">query the reader for the number of times a transition from state Lost to
4719 state Glimpsed have been detected for the particular source.</xsd:documentation>
4720     </xsd:annotation>
4721     <xsd:complexType/>
4722   </xsd:element>
4723   <xsd:element name="getLostToUnknownCount">
4724     <xsd:annotation>
4725       <xsd:documentation xml:lang="en">query the reader for the number of times a transition from state Lost to
4726 state Unknown have been detected for the particular source.</xsd:documentation>
4727     </xsd:annotation>
4728     <xsd:complexType/>
4729   </xsd:element>
4730   <xsd:element name="getOperStatus">
4731     <xsd:annotation>
4732       <xsd:documentation xml:lang="en">query the reader for the operational status of this particular
4733 Source.</xsd:documentation>
4734     </xsd:annotation>
4735     <xsd:complexType/>
4736   </xsd:element>
4737   <xsd:element name="getAdminStatus">
4738     <xsd:annotation>
4739       <xsd:documentation xml:lang="en">query the reader for the administrative status of this particular
4740 Source.</xsd:documentation>
4741     </xsd:annotation>
4742     <xsd:complexType/>
4743   </xsd:element>
4744   <xsd:element name="setAdminStatus">
4745     <xsd:annotation>
4746       <xsd:documentation xml:lang="en">set the administrative status of this particular
4747 Source.</xsd:documentation>
4748     </xsd:annotation>
4749     <xsd:complexType>
4750       <xsd:sequence>
4751         <xsd:element name="administrativeStatus" type="rm:AdministrativeStatusParamType">
4752           <xsd:annotation>
4753             <xsd:documentation xml:lang="en">administrative status to set.</xsd:documentation>
4754           </xsd:annotation>
4755         </xsd:element>
4756       </xsd:sequence>
4757     </xsd:complexType>
4758   </xsd:element>
4759   <xsd:element name="getOperStatusAlarmControl">
4760     <xsd:annotation>
4761       <xsd:documentation xml:lang="en">get this particular source's operational status alarm control.
4762 </xsd:documentation>
4763     </xsd:annotation>
4764     <xsd:complexType/>
4765   </xsd:element>
4766   <xsd:any namespace="##any" processContents="lax">
4767     <xsd:annotation>
4768       <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4769     </xsd:annotation>

```

```

4770     </xsd:any>
4771   </xsd:choice>
4772 </xsd:complexType>
4773 <xsd:complexType name="TriggerCommand">
4774   <xsd:choice>
4775     <xsd:annotation>
4776       <xsd:documentation xml:lang="en"> Trigger object commands. </xsd:documentation>
4777     </xsd:annotation>
4778     <xsd:element name="getFireCount">
4779       <xsd:annotation>
4780         <xsd:documentation xml:lang="en">query the reader for the number of times a particular trigger has
4781 fired.</xsd:documentation>
4782       </xsd:annotation>
4783     </xsd:complexType/>
4784   </xsd:element>
4785   <xsd:any namespace="##any" processContents="lax">
4786     <xsd:annotation>
4787       <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4788     </xsd:annotation>
4789   </xsd:any>
4790 </xsd:choice>
4791 </xsd:complexType>
4792 <xsd:complexType name="TTOperationalStatusAlarmControlCommand">
4793   <xsd:choice>
4794     <xsd:annotation>
4795       <xsd:documentation xml:lang="en"> Transition triggered alarm control object commands.
4796 </xsd:documentation>
4797     </xsd:annotation>
4798     <xsd:element name="getTriggerFromState">
4799       <xsd:annotation>
4800         <xsd:documentation xml:lang="en">query the reader for the current value of the
4801 TTOperationalStatusAlarmControlâ€™s TriggerFromState attribute.</xsd:documentation>
4802       </xsd:annotation>
4803     </xsd:complexType/>
4804   </xsd:element>
4805   <xsd:element name="setTriggerFromState">
4806     <xsd:annotation>
4807       <xsd:documentation xml:lang="en">set the current value of the TTOperationalStatusAlarmControlâ€™s
4808 TriggerFromState attribute.</xsd:documentation>
4809     </xsd:annotation>
4810   </xsd:complexType>
4811     <xsd:sequence>
4812       <xsd:element name="triggerFromState" type="rm:OperationalStatusParamType">
4813         <xsd:annotation>
4814           <xsd:documentation xml:lang="en">trigger from state to set</xsd:documentation>
4815         </xsd:annotation>
4816       </xsd:element>
4817     </xsd:sequence>
4818   </xsd:complexType>
4819 </xsd:element>
4820   <xsd:element name="getTriggerToState">
4821     <xsd:annotation>
4822       <xsd:documentation xml:lang="en">query the reader for the current value of the
4823 TTOperationalStatusAlarmControlâ€™s TriggerToState attribute.</xsd:documentation>
4824     </xsd:annotation>
4825   </xsd:complexType/>

```

```

4826     </xsd:element>
4827     <xsd:element name="setTriggerToState">
4828       <xsd:annotation>
4829         <xsd:documentation xml:lang="en">set the current value of the TTOperatonalStatusAlarmControlâ™s
4830 TriggerToState attribute.</xsd:documentation>
4831       </xsd:annotation>
4832       <xsd:complexType>
4833         <xsd:sequence>
4834           <xsd:element name="triggerToState" type="rm:OperationalStatusParamType">
4835             <xsd:annotation>
4836               <xsd:documentation xml:lang="en">trigger to state to set</xsd:documentation>
4837             </xsd:annotation>
4838           </xsd:element>
4839         </xsd:sequence>
4840       </xsd:complexType>
4841     </xsd:element>
4842     <xsd:any namespace="##any" processContents="lax">
4843       <xsd:annotation>
4844         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4845       </xsd:annotation>
4846     </xsd:any>
4847   </xsd:choice>
4848 </xsd:complexType>
4849 </xsd:schema>

```

10.3.2 Reply XML Message Encoding (Reader-To-Host)

```

4852 <?xml version="1.0" encoding="UTF-8"?>
4853 <!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Gerhard Gangl (7iD) -->
4854 <xsd:schema targetNamespace="urn:epcglobal:rm:xsd:1" elementFormDefault="unqualified"
4855 attributeFormDefault="unqualified" version="1.0" xmlns:rm="urn:epcglobal:rm:xsd:1"
4856 xmlns:epcglobal="urn:epcglobal:xsd:1" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
4857   <xsd:annotation>
4858     <xsd:documentation xml:lang="en">
4859       <epcglobal:copyright>Copyright&#169;2005-2006 Epcglobal Inc., All Rights Reserved.</epcglobal:copyright>
4860       <epcglobal:disclaimer>EPCglobal Inc., its members, officers, directors, employees, or
4861         agents shall not be liable for any injury, loss, damages, financial or otherwise,
4862         arising from, related to, or caused by the use of this document. The use of said
4863         document shall constitute your express consent to the foregoing exculpation.</epcglobal:disclaimer>
4864       <epcglobal:specification>Reader Management (RM) version 1.0</epcglobal:specification>
4865     </xsd:documentation>
4866   </xsd:annotation>
4867   <xsd:include schemaLocation="RmCommon.xsd"/>
4868   <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EpcGlobal.xsd"/>
4869   <!-- Reader Management Reply -->
4870   <xsd:element name="reply">
4871     <xsd:annotation>
4872       <xsd:documentation xml:lang="en"> This element defines a reader management reply. Just like commands,
4873       replies are grouped by the object they belong to. To specify the object a reply relates to, the name of the object is
4874       used. All the replies contain a single return value inside the reply. return value is enclosed by the element
4875       "returnValue". In case of a single return value it is directly added to this      element. In case of an array of return
4876       values they are enclosed within element
4877       "list" with each value in turn enclosed in element "value". Each reply is wrapped in
4878       its object type. </xsd:documentation>

```

```

4879     </xsd:annotation>
4880     <xsd:complexType>
4881         <xsd:complexContent>
4882             <xsd:extension base="epcglobal:Document">
4883                 <xsd:sequence>
4884                     <xsd:element name="id" type="xsd:string">
4885                         <xsd:annotation>
4886                             <xsd:documentation xml:lang="en">The id of the command for which this is the
4887 reply. </xsd:documentation>
4888                         </xsd:annotation>
4889                     </xsd:element>
4890                     <xsd:element name="resultCode" type="xsd:int">
4891                         <xsd:annotation>
4892                             <xsd:documentation xml:lang="en">the error code for the command - 0 for
4893 success. </xsd:documentation>
4894                         </xsd:annotation>
4895                     </xsd:element>
4896                     <xsd:choice>
4897                         <!-- Error information -->
4898                         <xsd:element name="error" type="rm:ErrorType">
4899                             <xsd:annotation>
4900                                 <xsd:documentation xml:lang="en">Information about the error. </xsd:documentation>
4901                             </xsd:annotation>
4902                         </xsd:element>
4903                         <!-- Reader Management (Target) Object Types -->
4904                         <!-- elements inside are ordered in alphabetical order -->
4905                         <xsd:element name="alarmChannel" type="rm:AlarmChannelReply">
4906                             <xsd:annotation>
4907                                 <xsd:documentation xml:lang="en">An Alarm channel </xsd:documentation>
4908                             </xsd:annotation>
4909                         </xsd:element>
4910                         <xsd:element name="alarmControl" type="rm:AlarmControlReply">
4911                             <xsd:annotation>
4912                                 <xsd:documentation xml:lang="en">AlarmControl is the base class for all of classes
4913 within the RMP object model responsible for controlling the generation of alarm messages within an EPCglobal
4914 compliant Reader. </xsd:documentation>
4915                             </xsd:annotation>
4916                         </xsd:element>
4917                         <xsd:element name="antennaReadPoint" type="rm:AntennaReadPointReply">
4918                             <xsd:annotation>
4919                                 <xsd:documentation xml:lang="en">Extension of Read point object for
4920 antenna. </xsd:documentation>
4921                             </xsd:annotation>
4922                         </xsd:element>
4923                         <xsd:element name="edgeTriggeredAlarmControl" type="rm:EdgeTriggeredAlarmControlReply">
4924                             <xsd:annotation>
4925                                 <xsd:documentation xml:lang="en">This class extends AlarmControl to control alarms
4926 generated when a monitored, integer-valued, model element first crosses a threshold value (the
4927 AlarmThreshold). </xsd:documentation>
4928                             </xsd:annotation>
4929                         </xsd:element>
4930                         <xsd:element name="iOPort" type="rm:IOPortReply">
4931                             <xsd:annotation>
4932                                 <xsd:documentation xml:lang="en">The hardware element that provides external input
4933 and output lines to connect to other components outside the reader device. </xsd:documentation>
4934                             </xsd:annotation>

```

```

4935         </xsd:element>
4936         <xsd:element name="notificationChannel" type="rm:NotificationChannelReply">
4937             <xsd:annotation>
4938                 <xsd:documentation xml:lang="en">The notification channel
4939                 carries messages issued asynchronously by the Reader to the Host.</xsd:documentation>
4940             </xsd:annotation>
4941         </xsd:element>
4942         <xsd:element name="readerDevice" type="rm:ReaderDeviceReply">
4943             <xsd:annotation>
4944                 <xsd:documentation xml:lang="en">A Reader</xsd:documentation>
4945             </xsd:annotation>
4946         </xsd:element>
4947         <xsd:element name="readPoint" type="rm:ReadPointReply">
4948             <xsd:annotation>
4949                 <xsd:documentation xml:lang="en">A ReadPoint</xsd:documentation>
4950             </xsd:annotation>
4951         </xsd:element>
4952         <xsd:element name="source" type="rm:SourceReply">
4953             <xsd:annotation>
4954                 <xsd:documentation xml:lang="en">A Read source</xsd:documentation>
4955             </xsd:annotation>
4956         </xsd:element>
4957         <xsd:element name="trigger" type="rm:TriggerReply">
4958             <xsd:annotation>
4959                 <xsd:documentation xml:lang="en">A Read/Notify Trigger</xsd:documentation>
4960             </xsd:annotation>
4961         </xsd:element>
4962         <xsd:element name="tTOperationalStatusAlarmControl"
4963 type="rm:TTOperationalStatusAlarmControlReply">
4964             <xsd:annotation>
4965                 <xsd:documentation xml:lang="en">This class extends AlarmControl to control alarms
4966 generated when a monitored model element of type OperationalStatus transitions to a new
4967 value.</xsd:documentation>
4968             </xsd:annotation>
4969         </xsd:element>
4970         <xsd:any namespace="##any" processContents="lax">
4971             <xsd:annotation>
4972                 <xsd:documentation>For standard and vendor extensions</xsd:documentation>
4973             </xsd:annotation>
4974         </xsd:any>
4975     </xsd:choice>
4976 </xsd:sequence>
4977 </xsd:extension>
4978 </xsd:complexContent>
4979 </xsd:complexType>
4980 </xsd:element>
4981 <xsd:complexType name="ErrorType">
4982     <xsd:annotation>
4983         <xsd:documentation xml:lang="en">Information about the error.</xsd:documentation>
4984     </xsd:annotation>
4985     <xsd:sequence>
4986         <xsd:element name="name" type="xsd:string" minOccurs="0">
4987             <xsd:annotation>
4988                 <xsd:documentation xml:lang="en">Name of the error.</xsd:documentation>
4989             </xsd:annotation>
4990         </xsd:element>

```

```

4991     <xsd:element name="cause" type="xsd:string" minOccurs="0">
4992         <xsd:annotation>
4993             <xsd:documentation xml:lang="en">Cause of the error.</xsd:documentation>
4994         </xsd:annotation>
4995     </xsd:element>
4996     <xsd:element name="description" type="xsd:string" minOccurs="0">
4997         <xsd:annotation>
4998             <xsd:documentation xml:lang="en">Description of the error.</xsd:documentation>
4999         </xsd:annotation>
5000     </xsd:element>
5001     <xsd:any namespace="##any" processContents="lax">
5002         <xsd:annotation>
5003             <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5004         </xsd:annotation>
5005     </xsd:any>
5006 </xsd:sequence>
5007 </xsd:complexType>
5008 <!-- Reader Management (Target) Object Types -->
5009 <!-- types are ordered in alphabetical order -->
5010 <!-- inside each type replies are listed in the order in which it is defined in the RM specification. -->
5011 <xsd:complexType name="AlarmChannelReply">
5012     <xsd:choice>
5013         <xsd:annotation>
5014             <xsd:documentation xml:lang="en"> Alarm channel object replies.</xsd:documentation>
5015         </xsd:annotation>
5016         <xsd:element name="create">
5017             <xsd:annotation>
5018                 <xsd:documentation xml:lang="en">reply for create an Alarm channel.</xsd:documentation>
5019             </xsd:annotation>
5020             <xsd:complexType>
5021                 <xsd:sequence>
5022                     <xsd:element name="returnValue" type="rm:AlarmChannelParamType">
5023                         <xsd:annotation>
5024                             <xsd:documentation xml:lang="en"> created alarm channel.</xsd:documentation>
5025                         </xsd:annotation>
5026                     </xsd:element>
5027                 </xsd:sequence>
5028             </xsd:complexType>
5029         </xsd:element>
5030         <xsd:element name="getName">
5031             <xsd:annotation>
5032                 <xsd:documentation xml:lang="en"> reply for get Name.</xsd:documentation>
5033             </xsd:annotation>
5034             <xsd:complexType>
5035                 <xsd:sequence>
5036                     <xsd:element name="returnValue" type="xsd:string">
5037                         <xsd:annotation>
5038                             <xsd:documentation xml:lang="en">This AlarmChannel's name</xsd:documentation>
5039                         </xsd:annotation>
5040                     </xsd:element>
5041                 </xsd:sequence>
5042             </xsd:complexType>
5043         </xsd:element>
5044         <xsd:element name="getAddress">
5045             <xsd:annotation>
5046                 <xsd:documentation xml:lang="en"> reply for get Address.</xsd:documentation>

```

```

5047     </xsd:annotation>
5048     <xsd:complexType>
5049         <xsd:sequence>
5050             <xsd:element name="returnValue" type="rm:AddressParamType">
5051                 <xsd:annotation>
5052                     <xsd:documentation xml:lang="en">reporting address for this AlarmChannel.
5053 </xsd:documentation>
5054                 </xsd:annotation>
5055             </xsd:element>
5056         </xsd:sequence>
5057     </xsd:complexType>
5058 </xsd:element>
5059 <xsd:element name="setAddress">
5060     <xsd:annotation>
5061         <xsd:documentation xml:lang="en">reply for set Address</xsd:documentation>
5062     </xsd:annotation>
5063     <xsd:complexType/>
5064 </xsd:element>
5065 <xsd:any namespace="##any" processContents="lax">
5066     <xsd:annotation>
5067         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5068     </xsd:annotation>
5069 </xsd:any>
5070 </xsd:choice>
5071 </xsd:complexType>
5072 <xsd:complexType name="AlarmControlReply">
5073     <xsd:choice>
5074         <xsd:annotation>
5075             <xsd:documentation xml:lang="en"> Alarm control object replies. </xsd:documentation>
5076         </xsd:annotation>
5077         <xsd:element name="getName">
5078             <xsd:annotation>
5079                 <xsd:documentation xml:lang="en"> reply for get Name.</xsd:documentation>
5080             </xsd:annotation>
5081             <xsd:complexType>
5082                 <xsd:sequence>
5083                     <xsd:element name="returnValue" type="xsd:string">
5084                         <xsd:annotation>
5085                             <xsd:documentation xml:lang="en">This AlarmControl's name</xsd:documentation>
5086                         </xsd:annotation>
5087                     </xsd:element>
5088                 </xsd:sequence>
5089             </xsd:complexType>
5090         </xsd:element>
5091         <xsd:element name="getEnabled">
5092             <xsd:annotation>
5093                 <xsd:documentation xml:lang="en">reply for get enabled.</xsd:documentation>
5094             </xsd:annotation>
5095             <xsd:complexType>
5096                 <xsd:sequence>
5097                     <xsd:element name="returnValue" type="xsd:boolean">
5098                         <xsd:annotation>
5099                             <xsd:documentation xml:lang="en">Whether alarm control is enabled.</xsd:documentation>
5100                         </xsd:annotation>
5101                     </xsd:element>
5102                 </xsd:sequence>

```

```

5103     </xsd:complexType>
5104 </xsd:element>
5105 <xsd:element name="setEnabled">
5106     <xsd:annotation>
5107         <xsd:documentation xml:lang="en">reply for set enabled.</xsd:documentation>
5108     </xsd:annotation>
5109     <xsd:complexType/>
5110 </xsd:element>
5111 <xsd:element name="getLevel">
5112     <xsd:annotation>
5113         <xsd:documentation xml:lang="en">reply for get Level.</xsd:documentation>
5114     </xsd:annotation>
5115     <xsd:complexType>
5116         <xsd:sequence>
5117             <xsd:element name="returnValue" type="rm:AlarmLevelParamType">
5118                 <xsd:annotation>
5119                     <xsd:documentation xml:lang="en">current value of the AlarmControl's
5120 Level.</xsd:documentation>
5121                 </xsd:annotation>
5122             </xsd:element>
5123         </xsd:sequence>
5124     </xsd:complexType>
5125 </xsd:element>
5126 <xsd:element name="setLevel">
5127     <xsd:annotation>
5128         <xsd:documentation xml:lang="en">reply for set Level.</xsd:documentation>
5129     </xsd:annotation>
5130     <xsd:complexType/>
5131 </xsd:element>
5132 <xsd:element name="getSuppressInterval">
5133     <xsd:annotation>
5134         <xsd:documentation xml:lang="en">reply for get SuppressInterval.</xsd:documentation>
5135     </xsd:annotation>
5136     <xsd:complexType>
5137         <xsd:sequence>
5138             <xsd:element name="returnValue" type="xsd:int">
5139                 <xsd:annotation>
5140                     <xsd:documentation xml:lang="en">current value of the AlarmControl's SuppressInterval in
5141 seconds.</xsd:documentation>
5142                 </xsd:annotation>
5143             </xsd:element>
5144         </xsd:sequence>
5145     </xsd:complexType>
5146 </xsd:element>
5147 <xsd:element name="setSuppressInterval">
5148     <xsd:annotation>
5149         <xsd:documentation xml:lang="en">reply for set SuppressInterval.</xsd:documentation>
5150     </xsd:annotation>
5151     <xsd:complexType/>
5152 </xsd:element>
5153 <xsd:any namespace="##any" processContents="lax">
5154     <xsd:annotation>
5155         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5156     </xsd:annotation>
5157 </xsd:any>
5158 </xsd:choice>

```

```

5159 </xsd:complexType>
5160 <xsd:complexType name="AntennaReadPointReply">
5161   <xsd:choice>
5162     <xsd:annotation>
5163       <xsd:documentation xml:lang="en"> Antenna read point object replies </xsd:documentation>
5164     </xsd:annotation>
5165     <xsd:element name="getIdentificationCount">
5166       <xsd:annotation>
5167         <xsd:documentation xml:lang="en">reply for get identification count.</xsd:documentation>
5168       </xsd:annotation>
5169       <xsd:complexType>
5170         <xsd:sequence>
5171           <xsd:element name="returnValue" type="xsd:int">
5172             <xsd:annotation>
5173               <xsd:documentation xml:lang="en">The count of the successful tag identifiers read at this
5174 AntennaReadPoint.</xsd:documentation>
5175             </xsd:annotation>
5176           </xsd:element>
5177         </xsd:sequence>
5178       </xsd:complexType>
5179     </xsd:element>
5180     <xsd:element name="getFailedIdentificationCount">
5181       <xsd:annotation>
5182         <xsd:documentation xml:lang="en">reply for get failed identification count.</xsd:documentation>
5183       </xsd:annotation>
5184       <xsd:complexType>
5185         <xsd:sequence>
5186           <xsd:element name="returnValue" type="xsd:int">
5187             <xsd:annotation>
5188               <xsd:documentation xml:lang="en">The count of the failed attempts at reading the identifier for a
5189 tag at this antenna ReadPoint.</xsd:documentation>
5190             </xsd:annotation>
5191           </xsd:element>
5192         </xsd:sequence>
5193       </xsd:complexType>
5194     </xsd:element>
5195     <xsd:element name="getMemReadCount">
5196       <xsd:annotation>
5197         <xsd:documentation xml:lang="en">reply for get mem read count.</xsd:documentation>
5198       </xsd:annotation>
5199       <xsd:complexType>
5200         <xsd:sequence>
5201           <xsd:element name="returnValue" type="xsd:int">
5202             <xsd:annotation>
5203               <xsd:documentation xml:lang="en">The count of the successful tag memory reads at this antenna
5204 ReadPoint.</xsd:documentation>
5205             </xsd:annotation>
5206           </xsd:element>
5207         </xsd:sequence>
5208       </xsd:complexType>
5209     </xsd:element>
5210     <xsd:element name="getFailedMemReadCount">
5211       <xsd:annotation>
5212         <xsd:documentation xml:lang="en">reply for get failed mem read count.</xsd:documentation>
5213       </xsd:annotation>
5214     </xsd:complexType>

```

```

5215     <xsd:sequence>
5216         <xsd:element name="returnValue" type="xsd:int">
5217             <xsd:annotation>
5218                 <xsd:documentation xml:lang="en">The count of the failed tag memory reads at this antenna
5219 ReadPoint.</xsd:documentation>
5220             </xsd:annotation>
5221         </xsd:element>
5222     </xsd:sequence>
5223 </xsd:complexType>
5224 </xsd:element>
5225 <xsd:element name="getFailedMemReadAlarmControl">
5226     <xsd:annotation>
5227         <xsd:documentation xml:lang="en">reply for get failed mem read alarm control.</xsd:documentation>
5228     </xsd:annotation>
5229     <xsd:complexType>
5230         <xsd:sequence>
5231             <xsd:element name="returnValue" type="rm:AlarmControlParamType">
5232                 <xsd:annotation>
5233                     <xsd:documentation xml:lang="en">An alarm control for monitoring tag memory read
5234 failures.</xsd:documentation>
5235                 </xsd:annotation>
5236             </xsd:element>
5237         </xsd:sequence>
5238     </xsd:complexType>
5239 </xsd:element>
5240 <xsd:element name="getWriteCount">
5241     <xsd:annotation>
5242         <xsd:documentation xml:lang="en">reply for get write count.</xsd:documentation>
5243     </xsd:annotation>
5244     <xsd:complexType>
5245         <xsd:sequence>
5246             <xsd:element name="returnValue" type="xsd:int">
5247                 <xsd:annotation>
5248                     <xsd:documentation xml:lang="en">The count of the successful writes at this
5249 AntennaReadPoint.</xsd:documentation>
5250                 </xsd:annotation>
5251             </xsd:element>
5252         </xsd:sequence>
5253     </xsd:complexType>
5254 </xsd:element>
5255 <xsd:element name="getFailedWriteCount">
5256     <xsd:annotation>
5257         <xsd:documentation xml:lang="en">reply for get failed write count.</xsd:documentation>
5258     </xsd:annotation>
5259     <xsd:complexType>
5260         <xsd:sequence>
5261             <xsd:element name="returnValue" type="xsd:int">
5262                 <xsd:annotation>
5263                     <xsd:documentation xml:lang="en">The count of the failed writes at this
5264 AntennaReadPoint.</xsd:documentation>
5265                 </xsd:annotation>
5266             </xsd:element>
5267         </xsd:sequence>
5268     </xsd:complexType>
5269 </xsd:element>
5270 <xsd:element name="getFailedWriteAlarmControl">

```

```

5271     <xsd:annotation>
5272         <xsd:documentation xml:lang="en">reply for get failed write alarm control.</xsd:documentation>
5273     </xsd:annotation>
5274     <xsd:complexType>
5275         <xsd:sequence>
5276             <xsd:element name="returnValue" type="rm:AlarmControlParamType">
5277                 <xsd:annotation>
5278                     <xsd:documentation xml:lang="en">An alarm control for monitoring the number of failed
5279 writes.</xsd:documentation>
5280                 </xsd:annotation>
5281             </xsd:element>
5282         </xsd:sequence>
5283     </xsd:complexType>
5284 </xsd:element>
5285 <xsd:element name="getKillCount">
5286     <xsd:annotation>
5287         <xsd:documentation xml:lang="en">reply for get kill count.</xsd:documentation>
5288     </xsd:annotation>
5289     <xsd:complexType>
5290         <xsd:sequence>
5291             <xsd:element name="returnValue" type="xsd:int">
5292                 <xsd:annotation>
5293                     <xsd:documentation xml:lang="en">The count of the successful tag kills at this antenna
5294 ReadPoint.</xsd:documentation>
5295                 </xsd:annotation>
5296             </xsd:element>
5297         </xsd:sequence>
5298     </xsd:complexType>
5299 </xsd:element>
5300 <xsd:element name="getFailedKillCount">
5301     <xsd:annotation>
5302         <xsd:documentation xml:lang="en">reply for get failed kill count.</xsd:documentation>
5303     </xsd:annotation>
5304     <xsd:complexType>
5305         <xsd:sequence>
5306             <xsd:element name="returnValue" type="xsd:int">
5307                 <xsd:annotation>
5308                     <xsd:documentation xml:lang="en">The count of the failed tag kills at this antenna
5309 readpoint.</xsd:documentation>
5310                 </xsd:annotation>
5311             </xsd:element>
5312         </xsd:sequence>
5313     </xsd:complexType>
5314 </xsd:element>
5315 <xsd:element name="getFailedKillAlarmControl">
5316     <xsd:annotation>
5317         <xsd:documentation xml:lang="en">reply for get failed kill alarm control.</xsd:documentation>
5318     </xsd:annotation>
5319     <xsd:complexType>
5320         <xsd:sequence>
5321             <xsd:element name="returnValue" type="rm:AlarmControlParamType">
5322                 <xsd:annotation>
5323                     <xsd:documentation xml:lang="en">An alarm control for monitoring tag kill
5324 failures.</xsd:documentation>
5325                 </xsd:annotation>
5326             </xsd:element>

```

```

5327         </xsd:sequence>
5328     </xsd:complexType>
5329 </xsd:element>
5330 <xsd:element name="getEraseCount">
5331     <xsd:annotation>
5332         <xsd:documentation xml:lang="en">reply for get erase count.</xsd:documentation>
5333     </xsd:annotation>
5334     <xsd:complexType>
5335         <xsd:sequence>
5336             <xsd:element name="returnValue" type="xsd:int">
5337                 <xsd:annotation>
5338                     <xsd:documentation xml:lang="en">The count of the successful tag erasures at this antenna
5339 ReadPoint.</xsd:documentation>
5340                 </xsd:annotation>
5341             </xsd:element>
5342         </xsd:sequence>
5343     </xsd:complexType>
5344 </xsd:element>
5345 <xsd:element name="getFailedEraseCount">
5346     <xsd:annotation>
5347         <xsd:documentation xml:lang="en">reply for get failed erase count.</xsd:documentation>
5348     </xsd:annotation>
5349     <xsd:complexType>
5350         <xsd:sequence>
5351             <xsd:element name="returnValue" type="xsd:int">
5352                 <xsd:annotation>
5353                     <xsd:documentation xml:lang="en">The count of the failed tag erasures at this antenna
5354 readpoint.</xsd:documentation>
5355                 </xsd:annotation>
5356             </xsd:element>
5357         </xsd:sequence>
5358     </xsd:complexType>
5359 </xsd:element>
5360 <xsd:element name="getFailedEraseAlarmControl">
5361     <xsd:annotation>
5362         <xsd:documentation xml:lang="en">reply for get failed erase alarm control.</xsd:documentation>
5363     </xsd:annotation>
5364     <xsd:complexType>
5365         <xsd:sequence>
5366             <xsd:element name="returnValue" type="rm:AlarmControlParamType">
5367                 <xsd:annotation>
5368                     <xsd:documentation xml:lang="en">An alarm control for monitoring tag erasure
5369 failures.</xsd:documentation>
5370                 </xsd:annotation>
5371             </xsd:element>
5372         </xsd:sequence>
5373     </xsd:complexType>
5374 </xsd:element>
5375 <xsd:element name="getLockCount">
5376     <xsd:annotation>
5377         <xsd:documentation xml:lang="en">reply for get lock count.</xsd:documentation>
5378     </xsd:annotation>
5379     <xsd:complexType>
5380         <xsd:sequence>
5381             <xsd:element name="returnValue" type="xsd:int">
5382                 <xsd:annotation>

```

```

5383         <xsd:documentation xml:lang="en">The count of the successful tag locks at this antenna
5384 ReadPoint.</xsd:documentation>
5385     </xsd:annotation>
5386 </xsd:element>
5387 </xsd:sequence>
5388 </xsd:complexType>
5389 </xsd:element>
5390 <xsd:element name="getFailedLockCount">
5391     <xsd:annotation>
5392         <xsd:documentation xml:lang="en">reply for get failed lock count.</xsd:documentation>
5393     </xsd:annotation>
5394     <xsd:complexType>
5395         <xsd:sequence>
5396             <xsd:element name="returnValue" type="xsd:int">
5397                 <xsd:annotation>
5398                     <xsd:documentation xml:lang="en">The count of the failed tag locks at this antenna
5399 readpoint.</xsd:documentation>
5400                 </xsd:annotation>
5401             </xsd:element>
5402         </xsd:sequence>
5403     </xsd:complexType>
5404 </xsd:element>
5405 <xsd:element name="getFailedLockAlarmControl">
5406     <xsd:annotation>
5407         <xsd:documentation xml:lang="en">reply for get failed lock alarm control.</xsd:documentation>
5408     </xsd:annotation>
5409     <xsd:complexType>
5410         <xsd:sequence>
5411             <xsd:element name="returnValue" type="rm:AlarmControlParamType">
5412                 <xsd:annotation>
5413                     <xsd:documentation xml:lang="en">An alarm control for monitoring tag lock
5414 failures.</xsd:documentation>
5415                 </xsd:annotation>
5416             </xsd:element>
5417         </xsd:sequence>
5418     </xsd:complexType>
5419 </xsd:element>
5420 <xsd:element name="getTimeEnergized">
5421     <xsd:annotation>
5422         <xsd:documentation xml:lang="en">reply for get time energized.</xsd:documentation>
5423     </xsd:annotation>
5424     <xsd:complexType>
5425         <xsd:sequence>
5426             <xsd:element name="returnValue" type="xsd:int">
5427                 <xsd:annotation>
5428                     <xsd:documentation xml:lang="en">The number of milliseconds the AntennaReadPoint has been
5429 energized attempting communication with tags.</xsd:documentation>
5430                 </xsd:annotation>
5431             </xsd:element>
5432         </xsd:sequence>
5433     </xsd:complexType>
5434 </xsd:element>
5435 <xsd:element name="getPowerLevel">
5436     <xsd:annotation>
5437         <xsd:documentation xml:lang="en">reply for get power level.</xsd:documentation>
5438     </xsd:annotation>

```

```

5439     <xsd:complexType>
5440     <xsd:sequence>
5441     <xsd:element name="returnValue" type="xsd:int">
5442     <xsd:annotation>
5443     <xsd:documentation xml:lang="en">current transmit power level of the antenna
5444 ReadPoint.</xsd:documentation>
5445     </xsd:annotation>
5446     </xsd:element>
5447     </xsd:sequence>
5448     </xsd:complexType>
5449 </xsd:element>
5450 <xsd:element name="getNoiseLevel">
5451     <xsd:annotation>
5452     <xsd:documentation xml:lang="en">reply for get noise level.</xsd:documentation>
5453     </xsd:annotation>
5454     <xsd:complexType>
5455     <xsd:sequence>
5456     <xsd:element name="returnValue" type="xsd:int">
5457     <xsd:annotation>
5458     <xsd:documentation xml:lang="en">current noise level observed at the antenna
5459 readpoint.</xsd:documentation>
5460     </xsd:annotation>
5461     </xsd:element>
5462     </xsd:sequence>
5463     </xsd:complexType>
5464 </xsd:element>
5465 <xsd:any namespace="##any" processContents="lax">
5466     <xsd:annotation>
5467     <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5468     </xsd:annotation>
5469 </xsd:any>
5470 </xsd:choice>
5471 </xsd:complexType>
5472 <xsd:complexType name="EdgeTriggeredAlarmControlReply">
5473     <xsd:choice>
5474     <xsd:annotation>
5475     <xsd:documentation xml:lang="en"> Edge triggered alarm control object replies. </xsd:documentation>
5476     </xsd:annotation>
5477     <xsd:element name="getAlarmThreshold">
5478     <xsd:annotation>
5479     <xsd:documentation xml:lang="en">reply for get Alarm threshold.</xsd:documentation>
5480     </xsd:annotation>
5481     <xsd:complexType>
5482     <xsd:sequence>
5483     <xsd:element name="returnValue" type="xsd:int">
5484     <xsd:annotation>
5485     <xsd:documentation xml:lang="en">current value of the EdgeTriggeredAlarmControl's
5486 AlarmThreshold .</xsd:documentation>
5487     </xsd:annotation>
5488     </xsd:element>
5489     </xsd:sequence>
5490     </xsd:complexType>
5491 </xsd:element>
5492 <xsd:element name="setAlarmThreshold">
5493     <xsd:annotation>
5494     <xsd:documentation xml:lang="en">reply for set alarm threshold.</xsd:documentation>

```

```

5495     </xsd:annotation>
5496     <xsd:complexType/>
5497 </xsd:element>
5498 <xsd:element name="getRearmThreshold">
5499     <xsd:annotation>
5500         <xsd:documentation xml:lang="en">reply for get RearmThreshold.</xsd:documentation>
5501     </xsd:annotation>
5502     <xsd:complexType>
5503         <xsd:sequence>
5504             <xsd:element name="returnValue" type="xsd:int">
5505                 <xsd:annotation>
5506                     <xsd:documentation xml:lang="en">current value of the EdgeTriggeredAlarmControl's
5507 RearmThreshold.</xsd:documentation>
5508                 </xsd:annotation>
5509             </xsd:element>
5510         </xsd:sequence>
5511     </xsd:complexType>
5512 </xsd:element>
5513 <xsd:element name="setRearmThreshold">
5514     <xsd:annotation>
5515         <xsd:documentation xml:lang="en">reply for set RearmThreshold.</xsd:documentation>
5516     </xsd:annotation>
5517     <xsd:complexType/>
5518 </xsd:element>
5519 <xsd:element name="getDirection">
5520     <xsd:annotation>
5521         <xsd:documentation xml:lang="en">reply for get direction.</xsd:documentation>
5522     </xsd:annotation>
5523     <xsd:complexType>
5524         <xsd:sequence>
5525             <xsd:element name="returnValue" type="rm:EdgeTriggeredAlarmDirectionParamType">
5526                 <xsd:annotation>
5527                     <xsd:documentation xml:lang="en">current value of the EdgeTriggeredAlarmControl's
5528 Direction.</xsd:documentation>
5529                 </xsd:annotation>
5530             </xsd:element>
5531         </xsd:sequence>
5532     </xsd:complexType>
5533 </xsd:element>
5534 <xsd:element name="setDirection">
5535     <xsd:annotation>
5536         <xsd:documentation xml:lang="en">reply for set direction.</xsd:documentation>
5537     </xsd:annotation>
5538     <xsd:complexType/>
5539 </xsd:element>
5540 <xsd:element name="getStatus">
5541     <xsd:annotation>
5542         <xsd:documentation xml:lang="en">reply for get status.</xsd:documentation>
5543     </xsd:annotation>
5544     <xsd:complexType>
5545         <xsd:sequence>
5546             <xsd:element name="returnValue" type="rm:EdgeTriggeredAlarmStatusParamType">
5547                 <xsd:annotation>
5548                     <xsd:documentation xml:lang="en">current value of the EdgeTriggeredAlarmControl's
5549 Status.</xsd:documentation>
5550                 </xsd:annotation>

```

```

5551         </xsd:element>
5552     </xsd:sequence>
5553 </xsd:complexType>
5554 </xsd:element>
5555 <xsd:any namespace="##any" processContents="lax">
5556     <xsd:annotation>
5557         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5558     </xsd:annotation>
5559 </xsd:any>
5560 </xsd:choice>
5561 </xsd:complexType>
5562 <xsd:complexType name="IOPortReply">
5563     <xsd:choice>
5564         <xsd:annotation>
5565             <xsd:documentation xml:lang="en">IO port object replies. </xsd:documentation>
5566         </xsd:annotation>
5567         <xsd:element name="getName">
5568             <xsd:annotation>
5569                 <xsd:documentation xml:lang="en"> reply for get Name.</xsd:documentation>
5570             </xsd:annotation>
5571             <xsd:complexType>
5572                 <xsd:sequence>
5573                     <xsd:element name="returnValue" type="xsd:string">
5574                         <xsd:annotation>
5575                             <xsd:documentation xml:lang="en">This IO port's name</xsd:documentation>
5576                         </xsd:annotation>
5577                     </xsd:element>
5578                 </xsd:sequence>
5579             </xsd:complexType>
5580         </xsd:element>
5581         <xsd:element name="getDescription">
5582             <xsd:annotation>
5583                 <xsd:documentation xml:lang="en">reply for get description.</xsd:documentation>
5584             </xsd:annotation>
5585             <xsd:complexType>
5586                 <xsd:sequence>
5587                     <xsd:element name="returnValue" type="xsd:string">
5588                         <xsd:annotation>
5589                             <xsd:documentation xml:lang="en">a textual description of the IO-port.</xsd:documentation>
5590                         </xsd:annotation>
5591                     </xsd:element>
5592                 </xsd:sequence>
5593             </xsd:complexType>
5594         </xsd:element>
5595         <xsd:element name="setDescription">
5596             <xsd:annotation>
5597                 <xsd:documentation xml:lang="en">reply for set description.</xsd:documentation>
5598             </xsd:annotation>
5599             <xsd:complexType/>
5600         </xsd:element>
5601         <xsd:element name="getOperStatus">
5602             <xsd:annotation>
5603                 <xsd:documentation xml:lang="en">reply for get operational status.</xsd:documentation>
5604             </xsd:annotation>
5605             <xsd:complexType>
5606                 <xsd:sequence>

```

```

5607     <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
5608         <xsd:annotation>
5609             <xsd:documentation xml:lang="en">the operational status of the IO-port.</xsd:documentation>
5610         </xsd:annotation>
5611     </xsd:element>
5612 </xsd:sequence>
5613 </xsd:complexType>
5614 </xsd:element>
5615 <xsd:element name="getAdminStatus">
5616     <xsd:annotation>
5617         <xsd:documentation xml:lang="en">reply for get administrative status.</xsd:documentation>
5618     </xsd:annotation>
5619     <xsd:complexType>
5620         <xsd:sequence>
5621             <xsd:element name="returnValue" type="rm:AdministrativeStatusParamType">
5622                 <xsd:annotation>
5623                     <xsd:documentation xml:lang="en">the administrative status of the IO-port.</xsd:documentation>
5624                 </xsd:annotation>
5625             </xsd:element>
5626         </xsd:sequence>
5627     </xsd:complexType>
5628 </xsd:element>
5629 <xsd:element name="setAdminStatus">
5630     <xsd:annotation>
5631         <xsd:documentation xml:lang="en">reply for set admin status.</xsd:documentation>
5632     </xsd:annotation>
5633     <xsd:complexType/>
5634 </xsd:element>
5635 <xsd:element name="getOperStatusAlarmControl">
5636     <xsd:annotation>
5637         <xsd:documentation xml:lang="en">reply for get operational status alarm control.</xsd:documentation>
5638     </xsd:annotation>
5639     <xsd:complexType>
5640         <xsd:sequence>
5641             <xsd:element name="returnValue" type="rm:TTOperationalStatusAlarmControlParamType">
5642                 <xsd:annotation>
5643                     <xsd:documentation xml:lang="en">An alarm control for monitoring the operational status of the
5644 IOPort.</xsd:documentation>
5645                 </xsd:annotation>
5646             </xsd:element>
5647         </xsd:sequence>
5648     </xsd:complexType>
5649 </xsd:element>
5650 <xsd:any namespace="##any" processContents="lax">
5651     <xsd:annotation>
5652         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5653     </xsd:annotation>
5654 </xsd:any>
5655 </xsd:choice>
5656 </xsd:complexType>
5657 <xsd:complexType name="NotificationChannelReply">
5658     <xsd:choice>
5659         <xsd:annotation>
5660             <xsd:documentation xml:lang="en">Notification channel object replies</xsd:documentation>
5661         </xsd:annotation>
5662     <xsd:element name="getLastNotificationAttempt">

```

```

5663     <xsd:annotation>
5664         <xsd:documentation xml:lang="en">reply for get last notification attempt.</xsd:documentation>
5665     </xsd:annotation>
5666     <xsd:complexType>
5667         <xsd:sequence>
5668             <xsd:element name="returnValue" type="xsd:int">
5669                 <xsd:annotation>
5670                     <xsd:documentation xml:lang="en">the timestamp (TimeTicks) when the last attempt was made to
5671 send a notification to the given address.</xsd:documentation>
5672                 </xsd:annotation>
5673             </xsd:element>
5674         </xsd:sequence>
5675     </xsd:complexType>
5676 </xsd:element>
5677 <xsd:element name="getLastSuccessfulNotification">
5678     <xsd:annotation>
5679         <xsd:documentation xml:lang="en">reply for get last successful notification.</xsd:documentation>
5680     </xsd:annotation>
5681     <xsd:complexType>
5682         <xsd:sequence>
5683             <xsd:element name="returnValue" type="xsd:int">
5684                 <xsd:annotation>
5685                     <xsd:documentation xml:lang="en">the timestamp (TimeTicks) when the last successful
5686 notification was send to the given address.</xsd:documentation>
5687                 </xsd:annotation>
5688             </xsd:element>
5689         </xsd:sequence>
5690     </xsd:complexType>
5691 </xsd:element>
5692 <xsd:element name="getOperStatus">
5693     <xsd:annotation>
5694         <xsd:documentation xml:lang="en">reply for get operational status.</xsd:documentation>
5695     </xsd:annotation>
5696     <xsd:complexType>
5697         <xsd:sequence>
5698             <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
5699                 <xsd:annotation>
5700                     <xsd:documentation xml:lang="en">the operational status of the notification
5701 channel.</xsd:documentation>
5702                 </xsd:annotation>
5703             </xsd:element>
5704         </xsd:sequence>
5705     </xsd:complexType>
5706 </xsd:element>
5707 <xsd:element name="setAdminStatus">
5708     <xsd:annotation>
5709         <xsd:documentation xml:lang="en">reply for set administrative status.</xsd:documentation>
5710     </xsd:annotation>
5711     <xsd:complexType/>
5712 </xsd:element>
5713 <xsd:element name="getAdminStatus">
5714     <xsd:annotation>
5715         <xsd:documentation xml:lang="en">reply for get administrative status.</xsd:documentation>
5716     </xsd:annotation>
5717     <xsd:complexType>
5718         <xsd:sequence>

```

```

5719         <xsd:element name="returnValue" type="rm:AdministrativeStatusParamType">
5720             <xsd:annotation>
5721                 <xsd:documentation xml:lang="en">the administrative status of the notification
5722 channel.</xsd:documentation>
5723             </xsd:annotation>
5724         </xsd:element>
5725     </xsd:sequence>
5726 </xsd:complexType>
5727 </xsd:element>
5728 <xsd:element name="getOperStatusAlarmControl">
5729     <xsd:annotation>
5730         <xsd:documentation xml:lang="en">reply for get operational status alarm control.</xsd:documentation>
5731     </xsd:annotation>
5732     <xsd:complexType>
5733         <xsd:sequence>
5734             <xsd:element name="returnValue" type="rm:TTOperationalStatusAlarmControlParamType">
5735                 <xsd:annotation>
5736                     <xsd:documentation xml:lang="en">An alarm control for monitoring the operational status of the
5737 notification channel.</xsd:documentation>
5738                 </xsd:annotation>
5739             </xsd:element>
5740         </xsd:sequence>
5741     </xsd:complexType>
5742 </xsd:element>
5743 <xsd:any namespace="##any" processContents="lax">
5744     <xsd:annotation>
5745         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
5746     </xsd:annotation>
5747 </xsd:any>
5748 </xsd:choice>
5749 </xsd:complexType>
5750 <xsd:complexType name="ReaderDeviceReply">
5751     <xsd:annotation>
5752         <xsd:documentation xml:lang="en">Reader Device object replies. </xsd:documentation>
5753     </xsd:annotation>
5754     <xsd:choice>
5755         <xsd:element name="getDescription">
5756             <xsd:annotation>
5757                 <xsd:documentation xml:lang="en">reply for get description.</xsd:documentation>
5758             </xsd:annotation>
5759             <xsd:complexType>
5760                 <xsd:sequence>
5761                     <xsd:element name="returnValue" type="xsd:string">
5762                         <xsd:annotation>
5763                             <xsd:documentation xml:lang="en">user defined description of the reader.</xsd:documentation>
5764                         </xsd:annotation>
5765                     </xsd:element>
5766                 </xsd:sequence>
5767             </xsd:complexType>
5768         </xsd:element>
5769         <xsd:element name="setDescription">
5770             <xsd:annotation>
5771                 <xsd:documentation xml:lang="en">reply for set description.</xsd:documentation>
5772             </xsd:annotation>
5773             <xsd:complexType/>
5774         </xsd:element>

```

```

5775 <xsd:element name="getLocationDescription">
5776 <xsd:annotation>
5777 <xsd:documentation xml:lang="en">reply for get location description.</xsd:documentation>
5778 </xsd:annotation>
5779 <xsd:complexType>
5780 <xsd:sequence>
5781 <xsd:element name="returnValue" type="xsd:string">
5782 <xsd:annotation>
5783 <xsd:documentation xml:lang="en">user defined location description of the reader
5784 device.</xsd:documentation>
5785 </xsd:annotation>
5786 </xsd:element>
5787 </xsd:sequence>
5788 </xsd:complexType>
5789 </xsd:element>
5790 <xsd:element name="setLocationDescription">
5791 <xsd:annotation>
5792 <xsd:documentation xml:lang="en">reply for set location description.</xsd:documentation>
5793 </xsd:annotation>
5794 <xsd:complexType/>
5795 </xsd:element>
5796 <xsd:element name="getContact">
5797 <xsd:annotation>
5798 <xsd:documentation xml:lang="en">reply for get contact.</xsd:documentation>
5799 </xsd:annotation>
5800 <xsd:complexType>
5801 <xsd:sequence>
5802 <xsd:element name="returnValue" type="xsd:string">
5803 <xsd:annotation>
5804 <xsd:documentation xml:lang="en">user defined contact description.</xsd:documentation>
5805 </xsd:annotation>
5806 </xsd:element>
5807 </xsd:sequence>
5808 </xsd:complexType>
5809 </xsd:element>
5810 <xsd:element name="setContact">
5811 <xsd:annotation>
5812 <xsd:documentation xml:lang="en">reply for set contact.</xsd:documentation>
5813 </xsd:annotation>
5814 <xsd:complexType/>
5815 </xsd:element>
5816 <xsd:element name="getSerialNumber">
5817 <xsd:annotation>
5818 <xsd:documentation xml:lang="en">reply for get serial number.</xsd:documentation>
5819 </xsd:annotation>
5820 <xsd:complexType>
5821 <xsd:sequence>
5822 <xsd:element name="returnValue" type="xsd:string">
5823 <xsd:annotation>
5824 <xsd:documentation xml:lang="en">the serial number of the reader device.</xsd:documentation>
5825 </xsd:annotation>
5826 </xsd:element>
5827 </xsd:sequence>
5828 </xsd:complexType>
5829 </xsd:element>
5830 <xsd:element name="getOperStatus">

```

```

5831     <xsd:annotation>
5832         <xsd:documentation xml:lang="en">reply for get operational status.</xsd:documentation>
5833     </xsd:annotation>
5834     <xsd:complexType>
5835         <xsd:sequence>
5836             <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
5837                 <xsd:annotation>
5838                     <xsd:documentation xml:lang="en">the operational status of the reader.</xsd:documentation>
5839                 </xsd:annotation>
5840             </xsd:element>
5841         </xsd:sequence>
5842     </xsd:complexType>
5843 </xsd:element>
5844 <xsd:element name="getOperStatusAlarmControl">
5845     <xsd:annotation>
5846         <xsd:documentation xml:lang="en">reply for get operational status alarm control.</xsd:documentation>
5847     </xsd:annotation>
5848     <xsd:complexType>
5849         <xsd:sequence>
5850             <xsd:element name="returnValue" type="rm:TTOperationalStatusAlarmControlParamType">
5851                 <xsd:annotation>
5852                     <xsd:documentation xml:lang="en">An alarm control for monitoring the operational status of the
5853 reader.</xsd:documentation>
5854                 </xsd:annotation>
5855             </xsd:element>
5856         </xsd:sequence>
5857     </xsd:complexType>
5858 </xsd:element>
5859 <xsd:element name="getFreeMemory">
5860     <xsd:annotation>
5861         <xsd:documentation xml:lang="en">reply for get free memory.</xsd:documentation>
5862     </xsd:annotation>
5863     <xsd:complexType>
5864         <xsd:sequence>
5865             <xsd:element name="returnValue" type="xsd:int">
5866                 <xsd:annotation>
5867                     <xsd:documentation xml:lang="en">available free memory.</xsd:documentation>
5868                 </xsd:annotation>
5869             </xsd:element>
5870         </xsd:sequence>
5871     </xsd:complexType>
5872 </xsd:element>
5873 <xsd:element name="getFreeMemoryAlarmControl">
5874     <xsd:annotation>
5875         <xsd:documentation xml:lang="en">reply for get free memory alarm control.</xsd:documentation>
5876     </xsd:annotation>
5877     <xsd:complexType>
5878         <xsd:sequence>
5879             <xsd:element name="returnValue" type="rm:EdgeTriggeredAlarmControlParamType">
5880                 <xsd:annotation>
5881                     <xsd:documentation xml:lang="en">FreeMemoryAlarmControl.</xsd:documentation>
5882                 </xsd:annotation>
5883             </xsd:element>
5884         </xsd:sequence>
5885     </xsd:complexType>
5886 </xsd:element>

```

```

5887 <xsd:element name="getNTPServers">
5888   <xsd:annotation>
5889     <xsd:documentation xml:lang="en">reply for get NTP servers.</xsd:documentation>
5890   </xsd:annotation>
5891   <xsd:complexType>
5892     <xsd:sequence>
5893       <xsd:element name="returnValue">
5894         <xsd:complexType>
5895           <xsd:sequence>
5896             <xsd:element name="list">
5897               <xsd:complexType>
5898                 <xsd:sequence>
5899                   <xsd:element name="value" type="xsd:string"
5900 minOccurs="0" maxOccurs="unbounded">
5901                     <xsd:annotation>
5902                       <xsd:documentation xml:lang="en">
5903 NTP servers used by the device to synchronize its current UTC clock (TimeUTC). </xsd:documentation>
5904                     </xsd:annotation>
5905                   </xsd:element>
5906                 </xsd:sequence>
5907               </xsd:complexType>
5908             </xsd:element>
5909           </xsd:sequence>
5910         </xsd:complexType>
5911       </xsd:element>
5912     </xsd:sequence>
5913   </xsd:complexType>
5914 </xsd:element>
5915 <xsd:element name="getDHCPServer">
5916   <xsd:annotation>
5917     <xsd:documentation xml:lang="en">reply for get DHCP server.</xsd:documentation>
5918   </xsd:annotation>
5919   <xsd:complexType>
5920     <xsd:sequence>
5921       <xsd:element name="returnValue" type="xsd:string">
5922         <xsd:annotation>
5923           <xsd:documentation xml:lang="en">the DHCP server currently used by the device for DHCP
5924 requests.</xsd:documentation>
5925         </xsd:annotation>
5926       </xsd:element>
5927     </xsd:sequence>
5928   </xsd:complexType>
5929 </xsd:element>
5930 <xsd:element name="getIOPort">
5931   <xsd:annotation>
5932     <xsd:documentation xml:lang="en">reply for get IO port.</xsd:documentation>
5933   </xsd:annotation>
5934   <xsd:complexType>
5935     <xsd:sequence>
5936       <xsd:element name="returnValue" type="rm:IOPortParamType">
5937         <xsd:annotation>
5938           <xsd:documentation xml:lang="en">the named IO port object.</xsd:documentation>
5939         </xsd:annotation>
5940       </xsd:element>
5941     </xsd:sequence>
5942   </xsd:complexType>

```

```

5943 </xsd:element>
5944 <xsd:element name="getAllIOPorts">
5945   <xsd:annotation>
5946     <xsd:documentation xml:lang="en">reply for get all IO ports.</xsd:documentation>
5947   </xsd:annotation>
5948   <xsd:complexType>
5949     <xsd:sequence>
5950       <xsd:element name="returnValue">
5951         <xsd:complexType>
5952           <xsd:sequence>
5953             <xsd:element name="list">
5954               <xsd:complexType>
5955                 <xsd:sequence>
5956                   <xsd:element name="value" type="rm:IOPortParamType"
minOccurs="0" maxOccurs="unbounded">
5957                     <xsd:annotation>
5958                       <xsd:documentation xml:lang="en">
5959                         IOPort objects. </xsd:documentation>
5960                     </xsd:annotation>
5961                   </xsd:element>
5962                 </xsd:sequence>
5963               </xsd:complexType>
5964             </xsd:element>
5965           </xsd:sequence>
5966         </xsd:complexType>
5967       </xsd:element>
5968     </xsd:sequence>
5969   </xsd:complexType>
5970 </xsd:element>
5971 <xsd:element name="resetStatistics">
5972   <xsd:annotation>
5973     <xsd:documentation xml:lang="en">reply for reset statistics.</xsd:documentation>
5974   </xsd:annotation>
5975   <xsd:complexType/>
5976 </xsd:element>
5977 <xsd:element name="removeAlarmChannels">
5978   <xsd:annotation>
5979     <xsd:documentation xml:lang="en">reply for remove alarm channels.</xsd:documentation>
5980   </xsd:annotation>
5981   <xsd:complexType/>
5982 </xsd:element>
5983 <xsd:element name="removeAllAlarmChannels">
5984   <xsd:annotation>
5985     <xsd:documentation xml:lang="en">reply for remove all alarm channels.</xsd:documentation>
5986   </xsd:annotation>
5987   <xsd:complexType/>
5988 </xsd:element>
5989 <xsd:element name="getAlarmChannel">
5990   <xsd:annotation>
5991     <xsd:documentation xml:lang="en">reply for get alarm channel.</xsd:documentation>
5992   </xsd:annotation>
5993   <xsd:complexType>
5994     <xsd:sequence>
5995       <xsd:element name="returnValue" type="rm:AlarmChannelParamType">
5996         <xsd:annotation>
5997           <xsd:documentation xml:lang="en">the named AlarmChannel object.</xsd:documentation>
5998         </xsd:annotation>

```

```

5999         </xsd:annotation>
6000     </xsd:element>
6001 </xsd:sequence>
6002 </xsd:complexType>
6003 </xsd:element>
6004 <xsd:element name="getAllAlarmChannels">
6005     <xsd:annotation>
6006         <xsd:documentation xml:lang="en">reply for get all alarm channels.</xsd:documentation>
6007     </xsd:annotation>
6008     <xsd:complexType>
6009         <xsd:sequence>
6010             <xsd:element name="returnValue">
6011                 <xsd:complexType>
6012                     <xsd:sequence>
6013                         <xsd:element name="list">
6014                             <xsd:complexType>
6015                                 <xsd:sequence>
6016                                     <xsd:element name="value"
6017 type="rm:AlarmChannelParamType" minOccurs="0" maxOccurs="unbounded">
6018                                     <xsd:annotation>
6019                                         <xsd:documentation xml:lang="en">
6020 Alarm channels. </xsd:documentation>
6021                                     </xsd:annotation>
6022                                 </xsd:element>
6023                             </xsd:sequence>
6024                         </xsd:complexType>
6025                     </xsd:element>
6026                 </xsd:sequence>
6027             </xsd:complexType>
6028         </xsd:element>
6029     </xsd:sequence>
6030 </xsd:complexType>
6031 </xsd:element>
6032 <xsd:any namespace="##any" processContents="lax">
6033     <xsd:annotation>
6034         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
6035     </xsd:annotation>
6036 </xsd:any>
6037 </xsd:choice>
6038 </xsd:complexType>
6039 <xsd:complexType name="ReadPointReply">
6040     <xsd:annotation>
6041         <xsd:documentation xml:lang="en">Read point object replies.</xsd:documentation>
6042     </xsd:annotation>
6043     <xsd:choice>
6044         <xsd:element name="getClassName">
6045             <xsd:annotation>
6046                 <xsd:documentation xml:lang="en">reply for get class name.</xsd:documentation>
6047             </xsd:annotation>
6048             <xsd:complexType>
6049                 <xsd:sequence>
6050                     <xsd:element name="returnValue" type="xsd:string">
6051                         <xsd:annotation>
6052                             <xsd:documentation xml:lang="en">The class name for the ReadPoint. The only supported return
6053 value is "AntennaReadPoint".</xsd:documentation>
6054                         </xsd:annotation>

```

```

6055         </xsd:element>
6056     </xsd:sequence>
6057 </xsd:complexType>
6058 </xsd:element>
6059 <xsd:element name="getDescription">
6060     <xsd:annotation>
6061         <xsd:documentation xml:lang="en">reply for get description.</xsd:documentation>
6062     </xsd:annotation>
6063     <xsd:complexType>
6064         <xsd:sequence>
6065             <xsd:element name="returnValue" type="xsd:string">
6066                 <xsd:annotation>
6067                     <xsd:documentation xml:lang="en">The readpoint descriptive name.</xsd:documentation>
6068                 </xsd:annotation>
6069             </xsd:element>
6070         </xsd:sequence>
6071     </xsd:complexType>
6072 </xsd:element>
6073 <xsd:element name="setDescription">
6074     <xsd:annotation>
6075         <xsd:documentation xml:lang="en">reply for set description.</xsd:documentation>
6076     </xsd:annotation>
6077     <xsd:complexType/>
6078 </xsd:element>
6079 <xsd:element name="getAdminStatus">
6080     <xsd:annotation>
6081         <xsd:documentation xml:lang="en">reply for get administrative status.</xsd:documentation>
6082     </xsd:annotation>
6083     <xsd:complexType>
6084         <xsd:sequence>
6085             <xsd:element name="returnValue" type="rm:AdministrativeStatusParamType">
6086                 <xsd:annotation>
6087                     <xsd:documentation xml:lang="en">the administrative status of the read
6088 point.</xsd:documentation>
6089                 </xsd:annotation>
6090             </xsd:element>
6091         </xsd:sequence>
6092     </xsd:complexType>
6093 </xsd:element>
6094 <xsd:element name="setAdminStatus">
6095     <xsd:annotation>
6096         <xsd:documentation xml:lang="en">reply for set administrative status.</xsd:documentation>
6097     </xsd:annotation>
6098     <xsd:complexType/>
6099 </xsd:element>
6100 <xsd:element name="getOperStatus">
6101     <xsd:annotation>
6102         <xsd:documentation xml:lang="en">reply for get operational status.</xsd:documentation>
6103     </xsd:annotation>
6104     <xsd:complexType>
6105         <xsd:sequence>
6106             <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
6107                 <xsd:annotation>
6108                     <xsd:documentation xml:lang="en">the operational status of the read point.</xsd:documentation>
6109                 </xsd:annotation>
6110             </xsd:element>

```

```

6111         </xsd:sequence>
6112     </xsd:complexType>
6113 </xsd:element>
6114 <xsd:element name="getOperStatusAlarmControl">
6115     <xsd:annotation>
6116         <xsd:documentation xml:lang="en">reply for get operational status alarm control.</xsd:documentation>
6117     </xsd:annotation>
6118     <xsd:complexType>
6119         <xsd:sequence>
6120             <xsd:element name="returnValue" type="rm:TTOperationalStatusAlarmControlParamType">
6121                 <xsd:annotation>
6122                     <xsd:documentation xml:lang="en">An alarm control for monitoring the operational status of the
6123 read point.</xsd:documentation>
6124                 </xsd:annotation>
6125             </xsd:element>
6126         </xsd:sequence>
6127     </xsd:complexType>
6128 </xsd:element>
6129 <xsd:any namespace="##any" processContents="lax">
6130     <xsd:annotation>
6131         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
6132     </xsd:annotation>
6133 </xsd:any>
6134 </xsd:choice>
6135 </xsd:complexType>
6136 <xsd:complexType name="SourceReply">
6137     <xsd:choice>
6138         <xsd:annotation>
6139             <xsd:documentation xml:lang="en">Source object replies. </xsd:documentation>
6140         </xsd:annotation>
6141         <xsd:element name="getUnknownToGlimpsedCount">
6142             <xsd:annotation>
6143                 <xsd:documentation xml:lang="en">reply for get unknown to glimpsed count.</xsd:documentation>
6144             </xsd:annotation>
6145             <xsd:complexType>
6146                 <xsd:sequence>
6147                     <xsd:element name="returnValue" type="xsd:int">
6148                         <xsd:annotation>
6149                             <xsd:documentation xml:lang="en">the number of times a transition from state Unknown to state
6150 Glimpsed have been detected for the particular source.</xsd:documentation>
6151                         </xsd:annotation>
6152                     </xsd:element>
6153                 </xsd:sequence>
6154             </xsd:complexType>
6155         </xsd:element>
6156         <xsd:element name="getGlimpsedToUnknownCount">
6157             <xsd:annotation>
6158                 <xsd:documentation xml:lang="en">reply for get glimpsed to unknown count.</xsd:documentation>
6159             </xsd:annotation>
6160             <xsd:complexType>
6161                 <xsd:sequence>
6162                     <xsd:element name="returnValue" type="xsd:int">
6163                         <xsd:annotation>
6164                             <xsd:documentation xml:lang="en">the number of times a transition from state Glimpsed to state
6165 Unknown have been detected for the particular source.</xsd:documentation>
6166                         </xsd:annotation>

```

```

6167         </xsd:element>
6168     </xsd:sequence>
6169 </xsd:complexType>
6170 </xsd:element>
6171 <xsd:element name="getGlimpsedToObservedCount">
6172     <xsd:annotation>
6173         <xsd:documentation xml:lang="en">reply for get Glimpsed to Observed count.</xsd:documentation>
6174     </xsd:annotation>
6175     <xsd:complexType>
6176         <xsd:sequence>
6177             <xsd:element name="returnValue" type="xsd:int">
6178                 <xsd:annotation>
6179                     <xsd:documentation xml:lang="en">the number of times a transition from state Glimpsed to state
6180 Observed have been detected for the particular source.</xsd:documentation>
6181                 </xsd:annotation>
6182             </xsd:element>
6183         </xsd:sequence>
6184     </xsd:complexType>
6185 </xsd:element>
6186 <xsd:element name="getObservedToLostCount">
6187     <xsd:annotation>
6188         <xsd:documentation xml:lang="en">reply for get Observed to Lost count.</xsd:documentation>
6189     </xsd:annotation>
6190     <xsd:complexType>
6191         <xsd:sequence>
6192             <xsd:element name="returnValue" type="xsd:int">
6193                 <xsd:annotation>
6194                     <xsd:documentation xml:lang="en">the number of times a transition from state Observed to state
6195 Lost have been detected for the particular source.</xsd:documentation>
6196                 </xsd:annotation>
6197             </xsd:element>
6198         </xsd:sequence>
6199     </xsd:complexType>
6200 </xsd:element>
6201 <xsd:element name="getLostToGlimpsedCount">
6202     <xsd:annotation>
6203         <xsd:documentation xml:lang="en">reply for get Lost to Glimpsed count.</xsd:documentation>
6204     </xsd:annotation>
6205     <xsd:complexType>
6206         <xsd:sequence>
6207             <xsd:element name="returnValue" type="xsd:int">
6208                 <xsd:annotation>
6209                     <xsd:documentation xml:lang="en">the number of times a transition from state Lost to state
6210 Observed have been detected for the particular source.</xsd:documentation>
6211                 </xsd:annotation>
6212             </xsd:element>
6213         </xsd:sequence>
6214     </xsd:complexType>
6215 </xsd:element>
6216 <xsd:element name="getLostToUnknownCount">
6217     <xsd:annotation>
6218         <xsd:documentation xml:lang="en">reply for get Lost to Unknown count.</xsd:documentation>
6219     </xsd:annotation>
6220     <xsd:complexType>
6221         <xsd:sequence>
6222             <xsd:element name="returnValue" type="xsd:int">

```

```

6223         <xsd:annotation>
6224             <xsd:documentation xml:lang="en">the number of times a transition from state Lost to state
6225 Unknown have been detected for the particular source.</xsd:documentation>
6226         </xsd:annotation>
6227     </xsd:element>
6228 </xsd:sequence>
6229 </xsd:complexType>
6230 </xsd:element>
6231 <xsd:element name="getOperStatus">
6232     <xsd:annotation>
6233         <xsd:documentation xml:lang="en">reply for get operational status.</xsd:documentation>
6234     </xsd:annotation>
6235     <xsd:complexType>
6236         <xsd:sequence>
6237             <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
6238                 <xsd:annotation>
6239                     <xsd:documentation xml:lang="en">the operational status of the source.</xsd:documentation>
6240                 </xsd:annotation>
6241             </xsd:element>
6242         </xsd:sequence>
6243     </xsd:complexType>
6244 </xsd:element>
6245 <xsd:element name="getAdminStatus">
6246     <xsd:annotation>
6247         <xsd:documentation xml:lang="en">reply for get administrative status.</xsd:documentation>
6248     </xsd:annotation>
6249     <xsd:complexType>
6250         <xsd:sequence>
6251             <xsd:element name="returnValue" type="rm:AdministrativeStatusParamType">
6252                 <xsd:annotation>
6253                     <xsd:documentation xml:lang="en">the administrative status of the source.</xsd:documentation>
6254                 </xsd:annotation>
6255             </xsd:element>
6256         </xsd:sequence>
6257     </xsd:complexType>
6258 </xsd:element>
6259 <xsd:element name="setAdminStatus">
6260     <xsd:annotation>
6261         <xsd:documentation xml:lang="en">reply for administrative status.</xsd:documentation>
6262     </xsd:annotation>
6263     <xsd:complexType/>
6264 </xsd:element>
6265 <xsd:element name="getOperStatusAlarmControl">
6266     <xsd:annotation>
6267         <xsd:documentation xml:lang="en">reply for get operational status alarm control.</xsd:documentation>
6268     </xsd:annotation>
6269     <xsd:complexType>
6270         <xsd:sequence>
6271             <xsd:element name="returnValue" type="rm:TTOperationalStatusAlarmControlParamType">
6272                 <xsd:annotation>
6273                     <xsd:documentation xml:lang="en">An alarm control for monitoring the operational status of
6274 source.</xsd:documentation>
6275                 </xsd:annotation>
6276             </xsd:element>
6277         </xsd:sequence>
6278     </xsd:complexType>

```

```

6279     </xsd:element>
6280     <xsd:any namespace="##any" processContents="lax">
6281         <xsd:annotation>
6282             <xsd:documentation>For standard and vendor extensions</xsd:documentation>
6283         </xsd:annotation>
6284     </xsd:any>
6285 </xsd:choice>
6286 </xsd:complexType>
6287 <xsd:complexType name="TriggerReply">
6288     <xsd:choice>
6289         <xsd:annotation>
6290             <xsd:documentation xml:lang="en"> Trigger object replies. </xsd:documentation>
6291         </xsd:annotation>
6292         <xsd:element name="getFireCount">
6293             <xsd:annotation>
6294                 <xsd:documentation xml:lang="en">reply for get fire count.</xsd:documentation>
6295             </xsd:annotation>
6296             <xsd:complexType>
6297                 <xsd:sequence>
6298                     <xsd:element name="returnValue" type="xsd:int">
6299                         <xsd:annotation>
6300                             <xsd:documentation xml:lang="en">the number of times a particular trigger has
6301 fired.</xsd:documentation>
6302                         </xsd:annotation>
6303                     </xsd:element>
6304                 </xsd:sequence>
6305             </xsd:complexType>
6306         </xsd:element>
6307     <xsd:any namespace="##any" processContents="lax">
6308         <xsd:annotation>
6309             <xsd:documentation>For standard and vendor extensions</xsd:documentation>
6310         </xsd:annotation>
6311     </xsd:any>
6312 </xsd:choice>
6313 </xsd:complexType>
6314 <xsd:complexType name="TTOperationalStatusAlarmControlReply">
6315     <xsd:choice>
6316         <xsd:annotation>
6317             <xsd:documentation xml:lang="en"> Transition triggered alarm control object replies. </xsd:documentation>
6318         </xsd:annotation>
6319         <xsd:element name="getTriggerFromState">
6320             <xsd:annotation>
6321                 <xsd:documentation xml:lang="en">reply for get trigger from state.</xsd:documentation>
6322             </xsd:annotation>
6323             <xsd:complexType>
6324                 <xsd:sequence>
6325                     <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
6326                         <xsd:annotation>
6327                             <xsd:documentation xml:lang="en">current value of the TTOperationalStatusAlarmControl's
6328 TriggerFromState.</xsd:documentation>
6329                         </xsd:annotation>
6330                     </xsd:element>
6331                 </xsd:sequence>
6332             </xsd:complexType>
6333         </xsd:element>
6334     <xsd:element name="setTriggerFromState">

```

```

6335     <xsd:annotation>
6336         <xsd:documentation xml:lang="en">reply for set trigger from state.</xsd:documentation>
6337     </xsd:annotation>
6338     <xsd:complexType/>
6339 </xsd:element>
6340 <xsd:element name="getTriggerToState">
6341     <xsd:annotation>
6342         <xsd:documentation xml:lang="en">reply for get trigger to state.</xsd:documentation>
6343     </xsd:annotation>
6344     <xsd:complexType>
6345         <xsd:sequence>
6346             <xsd:element name="returnValue" type="rm:OperationalStatusParamType">
6347                 <xsd:annotation>
6348                     <xsd:documentation xml:lang="en">current value of the TTOperationalStatusAlarmControl's
6349 TriggerToState.</xsd:documentation>
6350                 </xsd:annotation>
6351             </xsd:element>
6352         </xsd:sequence>
6353     </xsd:complexType>
6354 </xsd:element>
6355 <xsd:element name="setTriggerToState">
6356     <xsd:annotation>
6357         <xsd:documentation xml:lang="en">reply for set trigger to state.</xsd:documentation>
6358     </xsd:annotation>
6359     <xsd:complexType/>
6360 </xsd:element>
6361 <xsd:any namespace="##any" processContents="lax">
6362     <xsd:annotation>
6363         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
6364     </xsd:annotation>
6365 </xsd:any>
6366 </xsd:choice>
6367 </xsd:complexType>
6368 </xsd:schema>
6369

```

10.3.3 Alarm Notification XML Message Encoding (Reader-To-Host)

```

6370 <?xml version="1.0" encoding="UTF-8"?>
6371 <!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Gerhard Gangl (7iD) -->
6372 <xsd:schema targetNamespace="urn:epcglobal:rm:xsd:1" elementFormDefault="unqualified"
6373 attributeFormDefault="unqualified" version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6374 xmlns:epcglobal="urn:epcglobal:xsd:1" xmlns:rm="urn:epcglobal:rm:xsd:1">
6375     <xsd:annotation>
6376         <xsd:documentation xml:lang="en">
6377             <epcglobal:copyright>Copyright&#169;2005-2006 Epcglobal Inc., All Rights Reserved.</epcglobal:copyright>
6378             <epcglobal:disclaimer>EPCglobal Inc., its members, officers, directors, employees, or
6379             agents shall not be liable for any injury, loss, damages, financial or otherwise,
6380             arising from, related to, or caused by the use of this document. The use of said
6381             document shall constitute your express consent to the foregoing exculpation.</epcglobal:disclaimer>
6382             <epcglobal:specification>Reader Management (RM) version 1.0</epcglobal:specification>
6383         </xsd:documentation>
6384     </xsd:annotation>
6385     <xsd:include schemaLocation="RmCommon.xsd"/>
6386

```

```

6388 <xsd:import namespace="urn:epcglobal:xsd:1" schemaLocation="EpcGlobal.xsd"/>
6389 <!-- Reader Management Alarm -->
6390 <xsd:element name="alarm" type="rm:AlarmType">
6391   <xsd:annotation>
6392     <xsd:documentation xml:lang="en"> This element defines a reader management alarm.</xsd:documentation>
6393   </xsd:annotation>
6394 </xsd:element>
6395 <xsd:complexType name="ReaderType">
6396   <xsd:annotation>
6397     <xsd:documentation xml:lang="en">Reader Information</xsd:documentation>
6398   </xsd:annotation>
6399   <xsd:sequence>
6400     <xsd:element name="readerEPC" type="epcglobal:EPC" minOccurs="0">
6401       <xsd:annotation>
6402         <xsd:documentation xml:lang="en">Reader EPC</xsd:documentation>
6403       </xsd:annotation>
6404     </xsd:element>
6405     <xsd:element name="readerName" type="xsd:string" minOccurs="0">
6406       <xsd:annotation>
6407         <xsd:documentation xml:lang="en">Reader name</xsd:documentation>
6408       </xsd:annotation>
6409     </xsd:element>
6410     <xsd:element name="readerHandle" type="xsd:int" minOccurs="0">
6411       <xsd:annotation>
6412         <xsd:documentation xml:lang="en">Reader handle</xsd:documentation>
6413       </xsd:annotation>
6414     </xsd:element>
6415     <xsd:element name="readerRole" type="xsd:string" minOccurs="0">
6416       <xsd:annotation>
6417         <xsd:documentation xml:lang="en">Reader role</xsd:documentation>
6418       </xsd:annotation>
6419     </xsd:element>
6420     <xsd:element name="readerTime" type="rm:readerTime" minOccurs="0">
6421       <xsd:annotation>
6422         <xsd:documentation xml:lang="en">Reader time</xsd:documentation>
6423       </xsd:annotation>
6424     </xsd:element>
6425   </xsd:sequence>
6426 </xsd:complexType>
6427 <!-- Alarms -->
6428 <!-- Alarm types (including their base types) are ordered in alphabetical order -->
6429 <xsd:complexType name="AlarmType" abstract="true">
6430   <xsd:annotation>
6431     <xsd:documentation xml:lang="en">Alarm is the base class for all of classes within the RMP object model that
6432     define the contents of alarm messages</xsd:documentation>
6433   </xsd:annotation>
6434   <xsd:complexContent>
6435     <xsd:extension base="epcglobal:Document">
6436       <xsd:sequence>
6437         <xsd:element name="id" type="xsd:string">
6438           <xsd:annotation>
6439             <xsd:documentation xml:lang="en">id of this alarm.</xsd:documentation>
6440           </xsd:annotation>
6441         </xsd:element>
6442         <xsd:element name="reader" type="rm:ReaderType" minOccurs="0">
6443           <xsd:annotation>

```

```

6444         <xsd:documentation xml:lang="en">Reader sending the alarm</xsd:documentation>
6445     </xsd:annotation>
6446 </xsd:element>
6447 <xsd:element name="name" type="xsd:string">
6448     <xsd:annotation>
6449         <xsd:documentation xml:lang="en">Type of Alarm message</xsd:documentation>
6450     </xsd:annotation>
6451 </xsd:element>
6452 <xsd:element name="level" type="rm:AlarmLevelParamType">
6453     <xsd:annotation>
6454         <xsd:documentation xml:lang="en">Severity level assigned to the alarm</xsd:documentation>
6455     </xsd:annotation>
6456 </xsd:element>
6457 <xsd:element name="suppressCount" type="xsd:int">
6458     <xsd:annotation>
6459         <xsd:documentation xml:lang="en">Number of times the generation of this Alarm has been
6460 suppressed</xsd:documentation>
6461     </xsd:annotation>
6462 </xsd:element>
6463 <xsd:any namespace="##any" processContents="lax">
6464     <xsd:annotation>
6465         <xsd:documentation>For standard and vendor extensions</xsd:documentation>
6466     </xsd:annotation>
6467 </xsd:any>
6468 </xsd:sequence>
6469 </xsd:extension>
6470 </xsd:complexContent>
6471 </xsd:complexType>
6472 <xsd:complexType name="FailedEraseAlarmType">
6473     <xsd:annotation>
6474         <xsd:documentation xml:lang="en">This alarm's receipt signals a tag erase failure.</xsd:documentation>
6475     </xsd:annotation>
6476 <xsd:complexContent>
6477     <xsd:extension base="rm:AlarmType">
6478         <xsd:sequence>
6479             <xsd:element name="readPointName" type="xsd:string">
6480                 <xsd:annotation>
6481                     <xsd:documentation xml:lang="en">Read point where the erase failure
6482 occurred.</xsd:documentation>
6483                 </xsd:annotation>
6484             </xsd:element>
6485             <xsd:element name="failedEraseCount" type="xsd:int">
6486                 <xsd:annotation>
6487                     <xsd:documentation xml:lang="en">Value of AntennaReadPoint.FailedEraseCount element after the
6488 erase failure occurred.</xsd:documentation>
6489                 </xsd:annotation>
6490             </xsd:element>
6491             <xsd:element name="noiseLevel" type="xsd:int">
6492                 <xsd:annotation>
6493                     <xsd:documentation xml:lang="en">Value of the AntennaReadPoint.NoiseLevel element when the
6494 erase failure occurred.</xsd:documentation>
6495                 </xsd:annotation>
6496             </xsd:element>
6497         </xsd:sequence>
6498     </xsd:extension>
6499 </xsd:complexContent>

```

```

6500 </xsd:complexType>
6501 <xsd:complexType name="FailedKillAlarmType">
6502   <xsd:annotation>
6503     <xsd:documentation xml:lang="en">This alarm's receipt signals a tag kill failure.</xsd:documentation>
6504   </xsd:annotation>
6505   <xsd:complexContent>
6506     <xsd:extension base="rm:AlarmType">
6507       <xsd:sequence>
6508         <xsd:element name="readPointName" type="xsd:string">
6509           <xsd:annotation>
6510             <xsd:documentation xml:lang="en">Read point where the kill failure occurred.</xsd:documentation>
6511           </xsd:annotation>
6512         </xsd:element>
6513         <xsd:element name="failedKillCount" type="xsd:int">
6514           <xsd:annotation>
6515             <xsd:documentation xml:lang="en">Value of AntennaReadPoint.FailedKillCount element after the kill
6516 failure occurred.</xsd:documentation>
6517           </xsd:annotation>
6518         </xsd:element>
6519         <xsd:element name="noiseLevel" type="xsd:int">
6520           <xsd:annotation>
6521             <xsd:documentation xml:lang="en">Value of the AntennaReadPoint.NoiseLevel element when the kill
6522 failure occurred.</xsd:documentation>
6523           </xsd:annotation>
6524         </xsd:element>
6525       </xsd:sequence>
6526     </xsd:extension>
6527   </xsd:complexContent>
6528 </xsd:complexType>
6529 <xsd:complexType name="FailedLockAlarmType">
6530   <xsd:annotation>
6531     <xsd:documentation xml:lang="en">This alarm's receipt signals a tag lock failure.</xsd:documentation>
6532   </xsd:annotation>
6533   <xsd:complexContent>
6534     <xsd:extension base="rm:AlarmType">
6535       <xsd:sequence>
6536         <xsd:element name="readPointName" type="xsd:string">
6537           <xsd:annotation>
6538             <xsd:documentation xml:lang="en">Read point where the lock failure occurred.</xsd:documentation>
6539           </xsd:annotation>
6540         </xsd:element>
6541         <xsd:element name="failedLockCount" type="xsd:int">
6542           <xsd:annotation>
6543             <xsd:documentation xml:lang="en">Value of AntennaReadPoint.FailedLockCount element after the
6544 lock failure occurred.</xsd:documentation>
6545           </xsd:annotation>
6546         </xsd:element>
6547         <xsd:element name="noiseLevel" type="xsd:int">
6548           <xsd:annotation>
6549             <xsd:documentation xml:lang="en">Value of the AntennaReadPoint.NoiseLevel element when the
6550 lock failure occurred.</xsd:documentation>
6551           </xsd:annotation>
6552         </xsd:element>
6553       </xsd:sequence>
6554     </xsd:extension>
6555   </xsd:complexContent>

```

```

6556 </xsd:complexType>
6557 <xsd:complexType name="FailedMemReadAlarmType">
6558   <xsd:annotation>
6559     <xsd:documentation xml:lang="en">This alarm's receipt signals a tag user-memory read
6560 failure.</xsd:documentation>
6561   </xsd:annotation>
6562   <xsd:complexContent>
6563     <xsd:extension base="rm:AlarmType">
6564       <xsd:sequence>
6565         <xsd:element name="readPointName" type="xsd:string">
6566           <xsd:annotation>
6567             <xsd:documentation xml:lang="en">Read point where the memory read failure
6568 occurred.</xsd:documentation>
6569           </xsd:annotation>
6570         </xsd:element>
6571         <xsd:element name="failedMemReadCount" type="xsd:int">
6572           <xsd:annotation>
6573             <xsd:documentation xml:lang="en">Value of AntennaReadPoint.FailedMemReadCount element after
6574 the memory read failure occurred.</xsd:documentation>
6575           </xsd:annotation>
6576         </xsd:element>
6577         <xsd:element name="noiseLevel" type="xsd:int">
6578           <xsd:annotation>
6579             <xsd:documentation xml:lang="en">Value of the AntennaReadPoint.NoiseLevel element when the
6580 memory read failure occurred.</xsd:documentation>
6581           </xsd:annotation>
6582         </xsd:element>
6583       </xsd:sequence>
6584     </xsd:extension>
6585   </xsd:complexContent>
6586 </xsd:complexType>
6587 <xsd:complexType name="FailedWriteAlarmType">
6588   <xsd:annotation>
6589     <xsd:documentation xml:lang="en">This alarm's receipt signals a tag write failure.</xsd:documentation>
6590   </xsd:annotation>
6591   <xsd:complexContent>
6592     <xsd:extension base="rm:AlarmType">
6593       <xsd:sequence>
6594         <xsd:element name="readPointName" type="xsd:string">
6595           <xsd:annotation>
6596             <xsd:documentation xml:lang="en">Read point where the write failure
6597 occurred.</xsd:documentation>
6598           </xsd:annotation>
6599         </xsd:element>
6600         <xsd:element name="failedWriteCount" type="xsd:int">
6601           <xsd:annotation>
6602             <xsd:documentation xml:lang="en">Value of AntennaReadPoint.FailedWriteCount element after the
6603 write failure occurred.</xsd:documentation>
6604           </xsd:annotation>
6605         </xsd:element>
6606         <xsd:element name="noiseLevel" type="xsd:int">
6607           <xsd:annotation>
6608             <xsd:documentation xml:lang="en">Value of the AntennaReadPoint.NoiseLevel element when the
6609 write failure occurred.</xsd:documentation>
6610           </xsd:annotation>
6611         </xsd:element>

```

```

6612     </xsd:sequence>
6613   </xsd:extension>
6614 </xsd:complexContent>
6615 </xsd:complexType>
6616 <xsd:complexType name="FreeMemoryAlarmType">
6617   <xsd:annotation>
6618     <xsd:documentation xml:lang="en">This alarm's receipt signals the movement of a reader device's free
6619     memory (represented in the abstract model by ReaderDevice.FreeMemory) below a specified threshold
6620     value.</xsd:documentation>
6621   </xsd:annotation>
6622   <xsd:complexContent>
6623     <xsd:extension base="rm:AlarmType">
6624       <xsd:sequence>
6625         <xsd:element name="freeMemory" type="xsd:int">
6626           <xsd:annotation>
6627             <xsd:documentation xml:lang="en">Value of ReaderDevice.FreeMemory when the alarm was
6628             triggered.</xsd:documentation>
6629           </xsd:annotation>
6630         </xsd:element>
6631       </xsd:sequence>
6632     </xsd:extension>
6633   </xsd:complexContent>
6634 </xsd:complexType>
6635 <xsd:complexType name="IOPortOperStatusAlarmType">
6636   <xsd:annotation>
6637     <xsd:documentation xml:lang="en">This alarm's receipt signals a change in the operational status of a
6638     Reader's IO Port. </xsd:documentation>
6639   </xsd:annotation>
6640   <xsd:complexContent>
6641     <xsd:extension base="rm:TTOperStatusAlarmType">
6642       <xsd:sequence>
6643         <xsd:element name="iOPortName" type="xsd:string">
6644           <xsd:annotation>
6645             <xsd:documentation xml:lang="en">IO port name.</xsd:documentation>
6646           </xsd:annotation>
6647         </xsd:element>
6648       </xsd:sequence>
6649     </xsd:extension>
6650   </xsd:complexContent>
6651 </xsd:complexType>
6652 <xsd:complexType name="NotificationChannelOperStatusAlarmType">
6653   <xsd:annotation>
6654     <xsd:documentation xml:lang="en">This alarm's receipt signals a change in the operational status of one of a
6655     Reader's notification channels.</xsd:documentation>
6656   </xsd:annotation>
6657   <xsd:complexContent>
6658     <xsd:extension base="rm:TTOperStatusAlarmType">
6659       <xsd:sequence>
6660         <xsd:element name="notificationChannelName" type="xsd:string">
6661           <xsd:annotation>
6662             <xsd:documentation xml:lang="en"> Notification channel name.</xsd:documentation>
6663           </xsd:annotation>
6664         </xsd:element>
6665       </xsd:sequence>
6666     </xsd:extension>
6667   </xsd:complexContent>

```

```

6668 </xsd:complexType>
6669 <xsd:complexType name="ReaderDeviceOperStatusAlarmType">
6670   <xsd:annotation>
6671     <xsd:documentation xml:lang="en">This alarm's receipt signals a change in the operational status of a Reader.
6672   </xsd:documentation>
6673   </xsd:annotation>
6674   <xsd:complexContent>
6675     <xsd:extension base="rm:TTOperStatusAlarmType"/>
6676   </xsd:complexContent>
6677 </xsd:complexType>
6678 <xsd:complexType name="ReadPointOperStatusAlarmType">
6679   <xsd:annotation>
6680     <xsd:documentation xml:lang="en">This alarm's receipt signals a change in the operational status of one of a
6681   Reader's Read Points. </xsd:documentation>
6682   </xsd:annotation>
6683   <xsd:complexContent>
6684     <xsd:extension base="rm:TTOperStatusAlarmType">
6685       <xsd:sequence>
6686         <xsd:element name="readPointName" type="xsd:string">
6687           <xsd:annotation>
6688             <xsd:documentation xml:lang="en"> Read point name.</xsd:documentation>
6689           </xsd:annotation>
6690         </xsd:element>
6691       </xsd:sequence>
6692     </xsd:extension>
6693   </xsd:complexContent>
6694 </xsd:complexType>
6695 <xsd:complexType name="SourceOperStatusAlarmType">
6696   <xsd:annotation>
6697     <xsd:documentation xml:lang="en">This alarm's receipt signals a change in the operational status of a logical
6698   source of EPC data on a Reader. </xsd:documentation>
6699   </xsd:annotation>
6700   <xsd:complexContent>
6701     <xsd:extension base="rm:TTOperStatusAlarmType">
6702       <xsd:sequence>
6703         <xsd:element name="sourceName" type="xsd:string">
6704           <xsd:annotation>
6705             <xsd:documentation xml:lang="en"> Source name.</xsd:documentation>
6706           </xsd:annotation>
6707         </xsd:element>
6708       </xsd:sequence>
6709     </xsd:extension>
6710   </xsd:complexContent>
6711 </xsd:complexType>
6712 <xsd:complexType name="TTOperStatusAlarmType" abstract="true">
6713   <xsd:annotation>
6714     <xsd:documentation xml:lang="en">Base class for all transition triggered Operational Status Alarms.
6715   </xsd:documentation>
6716   </xsd:annotation>
6717   <xsd:complexContent>
6718     <xsd:extension base="rm:AlarmType">
6719       <xsd:sequence>
6720         <xsd:element name="fromState" type="rm:OperationalStatusParamType">
6721           <xsd:annotation>
6722             <xsd:documentation xml:lang="en">From state of the transition.</xsd:documentation>
6723           </xsd:annotation>

```

```

        </xsd:element>
        <xsd:element name="toState" type="rm:OperationalStatusParamType">
          <xsd:annotation>
            <xsd:documentation xml:lang="en">To state of the transition.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:schema>

```

10.3.4 Common Data Formats

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Gerhard Gangl (7iD) -->
<xsd:schema targetNamespace="urn:epcglobal:rm:xsd:1" elementFormDefault="unqualified"
attributeFormDefault="unqualified" version="1.0" xmlns:rm="urn:epcglobal:rm:xsd:1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    </xsd:documentation>
  </xsd:annotation>
  <!-- Reader Management types -->
  <!-- types are ordered in alphabetical order -->
  <xsd:annotation>
    <xsd:documentation xml:lang="en">Reader management common types. All types end with the
      suffix "Type". All the command parameter or reply return value reference types are
      suffixed with "ParamType". All named objects are referenced using "name". For example,
      NotificationChannelParamType contains a reference (its name) instead of actual notification channel
      itself.</xsd:documentation>
    </xsd:annotation>
    <xsd:simpleType name="AddressParamType">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">The address for a channel</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:anyURI"/>
    </xsd:simpleType>
    <xsd:simpleType name="AdministrativeStatusParamType">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">administrative status</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="UP"/>
        <xsd:enumeration value="DOWN"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="AlarmChannelListParamType">
      <xsd:annotation>
        <xsd:documentation>A list of alarm channels</xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="list">
          <xsd:complexType>

```

```

6777         <xsd:sequence>
6778             <xsd:element name="value" type="rm:AlarmChannelParamType" maxOccurs="unbounded"/>
6779         </xsd:sequence>
6780     </xsd:complexType>
6781 </xsd:element>
6782 </xsd:sequence>
6783 </xsd:complexType>
6784 <xsd:simpleType name="AlarmChannelParamType">
6785     <xsd:annotation>
6786         <xsd:documentation xml:lang="en">The name of a alarm channel.</xsd:documentation>
6787     </xsd:annotation>
6788     <xsd:restriction base="xsd:string"/>
6789 </xsd:simpleType>
6790 <xsd:simpleType name="AlarmControlParamType">
6791     <xsd:annotation>
6792         <xsd:documentation xml:lang="en">The name of an alarm control.</xsd:documentation>
6793     </xsd:annotation>
6794     <xsd:restriction base="xsd:string"/>
6795 </xsd:simpleType>
6796 <xsd:simpleType name="AlarmLevelParamType">
6797     <xsd:annotation>
6798         <xsd:documentation xml:lang="en"> Alarm level</xsd:documentation>
6799     </xsd:annotation>
6800     <xsd:restriction base="xsd:string">
6801         <xsd:enumeration value="EMERGENCY"/>
6802         <xsd:enumeration value="ALERT"/>
6803         <xsd:enumeration value="CRITICAL"/>
6804     </xsd:restriction>

```

6805 10.3.5 EPCglobal Standard Header

```

6806 <?xml version="1.0" encoding="UTF-8"?>
6807 <xsd:schema xmlns:epcglobal="urn:epcglobal:xsd:1" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6808 targetNamespace="urn:epcglobal:xsd:1" elementFormDefault="unqualified" attributeFormDefault="unqualified"
6809 version="1.0">
6810     <xsd:annotation>
6811         <xsd:documentation>
6812             <epcglobal:copyright>Copyright&#169;2005-2006 Epcglobal Inc., All Rights Reserved.</epcglobal:copyright>
6813             <epcglobal:disclaimer>EPCglobal Inc., its members, officers, directors, employees, or agents shall not be liable
6814 for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document.
6815 The use of said document shall constitute your express consent to the foregoing exculpation.</epcglobal:disclaimer>
6816             <epcglobal:specification>EPCglobal common components Version 1.0</epcglobal:specification>
6817         </xsd:documentation>
6818     </xsd:annotation>
6819     <xsd:complexType name="Document" abstract="true">
6820         <xsd:annotation>
6821             <xsd:documentation xml:lang="en">
6822 EPCglobal document properties for all messages.
6823             </xsd:documentation>
6824         </xsd:annotation>
6825         <xsd:attribute name="schemaVersion" type="xsd:decimal" use="required">
6826             <xsd:annotation>
6827                 <xsd:documentation xml:lang="en">
6828 The version of the schema corresponding to which the instance conforms.
6829                 </xsd:documentation>
6830             </xsd:annotation>

```

```

6831     </xsd:attribute>
6832     <xsd:attribute name="creationDate" type="xsd:dateTime" use="required">
6833         <xsd:annotation>
6834             <xsd:documentation xml:lang="en">
6835                 The date the message was created. Used for auditing and logging.
6836             </xsd:documentation>
6837         </xsd:annotation>
6838     </xsd:attribute>
6839 </xsd:complexType>
6840 <xsd:complexType name="EPC">
6841     <xsd:annotation>
6842         <xsd:documentation xml:lang="en">
6843             EPC represents the Electronic Product Code.
6844         </xsd:documentation>
6845     </xsd:annotation>
6846     <xsd:simpleContent>
6847         <xsd:extension base="xsd:string"/>
6848     </xsd:simpleContent>
6849 </xsd:complexType>
6850 </xsd:schema>
6851
6852     </xsd:attribute>
6853 </xsd:complexType>
6854 <xsd:complexType name="EPC">
6855     <xsd:annotation>
6856         <xsd:documentation xml:lang="en">
6857             EPC represents the Electronic Product Code.
6858         </xsd:documentation>
6859     </xsd:annotation>
6860     <xsd:simpleContent>
6861         <xsd:extension base="xsd:string"/>
6862     </xsd:simpleContent>
6863 </xsd:complexType>
6864 </xsd:schema>

```

10.4 SNMP MIB

This specification defines two MIB modules in the EPCglobal enterprise tree that expose the EPCglobal Reader Object Model across an SNMP protocol interface.

The first of these MIB modules, EPCGLOBAL-SMI-MIB, is the EPCglobal central registration module. It contains the specification of the private enterprise number (22695) the Internet Assigned Numbers Authority (IANA) assigned to EPCglobal. In addition, it defines the top-level organization of the EPCglobal private enterprise namespace.

The second MIB module, EPCGLOBAL-READER-MIB, contains MIB definitions corresponding to objects within the Reader Object Model. This MIB defines model objects that are particular to the health monitoring of RFID Readers.

When there is a clear mapping of Reader Object Model elements to standard MIB-II object definitions, we rely on the IETF-standardized MIB modules to expose these model elements across SNMP, rather than duplicating these object definitions within EPCglobal MIB modules. Below is a table displaying the mapping of selected Reader Object Model

6881 elements to object definitions within IETF MIB modules (in particular, SNMPv2-MIB
6882 defined in RFC3418/STD 62 and SNMP-TARGET-MIB, RFC3413/STD62).

Reader Object Model element	IETF-Defined MIB Module	MIB OID
ReaderDevice.Manufacturer ReaderDevice.Model ReaderDevice.ManufacturerDescription	SNMPv2-MIB	sysDescr
ReaderDevice.LocationDescription	SNMPv2-MIB	location
ReaderDevice.Contact	SNMPv2-MIB	sysContact
ReaderDevice.TimeTicks	SNMPv2-MIB	sysUpTime
ReaderDevice.AlarmChannels	SNMP-TARGET-MIB	snmpTargetObjects
AlarmChannel.Name	SNMP-TARGET-MIB	snmpTargetAddrName
AlarmChannel.Address	SNMP-TARGET-MIB	snmpTargetAddrTDomain snmpTargetAddrTAddress
Alarm.TimeTicks	Protocol Operations for SNMP (RFC3416)	sysUpTime

6883

6884

6885 In addition to compliance with the EPC Global Reader Management MIB, a compliant
6886 system that supports the SNMP binding SHALL implement support for the following
6887 IETF-standardized MIB groups:

- 6888 • the MIB-II System Group defined in the SNMPv2-MIB module, defined in RFC
6889 3418,
- 6890 • the MIB-II IP Group, defined in the IP-MIB module, RFC 2011,
- 6891 • the MIB-II Interfaces Group, defined in IF-MIB module, RFC 2863

6892 These groups define the basic System Identification operations, Internet Protocol and
6893 Network Interfaces monitoring objects and operations. In order to avoid unnecessary
6894 duplication and to harmonize reader management, as a networked device, with that of
6895 other SNMP-manageable devices, support of these MIB-II groups is chosen in place of
6896 defining these objects within in the EPCglobal RFID Reader Management MIB.

6897 Protocol support for the management of Alarm Channels is optional; however, if a reader
6898 vendor chooses to support SNMP-based management of Alarm Channels, the vendor
6899 SHALL do so by implementing the SNMP-TARGET-MIB module, defined in RFC
6900 3413.

6901 Within the EPCglobal MIB structure, elements referred to as "Alarms" in the Abstract
6902 Model are referred to as "Notifications". This is because the term "Alarm" in SNMP is
6903 used as a term of art (for example, in RFC 3877 and RFC 3878) and could lead to
6904 confusion within existing SNMP installations.

6905 Alarm TimeTicks are included in the SNMP Notification data structure mandated by
6906 RFC 3416, therefore not explicitly included in the MIB.

6907 Unless otherwise specified within the MIB, vendors MAY implement index persistence
6908 for any of the EPCglobal defined MIB structures. This statement only applies to the
6909 EPCglobal enterprise space.

6910 **10.4.1 Vendor Extension Details**

6911 Vendors wishing to provide vendor-specific extensions in SNMP MAY provide support
6912 for MIB modules outside the EPCglobal MIB space. For example, if a vendor supports
6913 vendor-specific MIB on their devices, they MAY also support these MIB modules on
6914 EPCglobal compliant readers. Additionally, a EPCglobal compliant readers MAY
6915 provide support for any IETF-standard MIB module, such as the HOST-RESOURCES-
6916 MIB (RFC 2790).

6918 10.4.2 EPCglobal RFID Reader Management MIB

6919 10.4.2.1 EPCglobal SMI MIB

```
6920 -- *****
6921 -- Copyright (c) 2005-2006 EPCglobal Inc(tm), All Rights Reserved.
6922 -- *****
6923 EPCGLOBAL-SMI-MIB DEFINITIONS ::= BEGIN
6924
6925 IMPORTS
6926     enterprises,
6927     MODULE-IDENTITY
6928     FROM SNMPv2-SMI;
6929
6930
6931 epcglobal MODULE-IDENTITY
6932     LAST-UPDATED "200610040000Z"
6933     ORGANIZATION "EPCglobal, Inc."
6934     CONTACT-INFO
6935         "
6936             EPCglobal MIB Administrator
6937             GS1/EPCglobal, Inc.
6938             Princeton Pike Corporate Center
6939             1009 Lenox Drive, Suite 202
6940             Lawrenceville, NJ 08648
6941             US
6942
6943             Tel: +1 609 620 0200
6944             Email: mibs@lists.epcglobalinc.org"
6945
6946 DESCRIPTION
6947     "The EPCglobal central registration module, containing
6948     the top-level organization of the EPCglobal private
6949     enterprise namespace."
6950
6951 REVISION      "200610040000Z"
6952
6953 DESCRIPTION
6954     "Defined in conformance with the EPCglobal Reader Management and
6955     Reader Protocol specifications"
6956
6957 ::= {enterprises 22695} -- assigned by IANA
6958
6959 --
6960 -- The EPCglobal private enterprise number assigned by
6961 -- the Internet Assigned Numbers Authority (IANA).
6962 --
6963 epcgSmiManagement      OBJECT IDENTIFIER ::= { epcglobal 1 }
6964 epcgSmiExperimental    OBJECT IDENTIFIER ::= { epcglobal 2 }
6965
6966 END
```

6966 10.4.2.2 EPCglobal Reader MIB

```
6967 -- Copyright (c) 2005, 2006 EPCglobal Inc(tm), All Rights Reserved.
6968 EPCGLOBAL-READER-MIB DEFINITIONS ::= BEGIN
6969
6970 IMPORTS
6971     Gauge32,
6972     Integer32,
6973     MODULE-IDENTITY,
6974     NOTIFICATION-TYPE,
```

```

6975 OBJECT-TYPE,
6976 Unsigned32,
6977 Counter32
6978     FROM SNMPv2-SMI
6979
6980 SnmpAdminString
6981     FROM SNMP-FRAMEWORK-MIB
6982
6983 DateAndTime,
6984 RowPointer,
6985 RowStatus,
6986 TEXTUAL-CONVENTION,
6987 TruthValue
6988     FROM SNMPv2-TC
6989
6990 MODULE-COMPLIANCE,
6991 NOTIFICATION-GROUP,
6992 OBJECT-GROUP
6993     FROM SNMPv2-CONF
6994
6995 InetAddress,
6996 InetAddressType
6997     FROM INET-ADDRESS-MIB
6998
6999 sysName
7000     FROM SNMPv2-MIB
7001
7002 epcgSmiManagement
7003     FROM EPCGLOBAL-SMI-MIB;
7004
7005 epcgReaderMIB MODULE-IDENTITY
7006     LAST-UPDATED "200610270000Z"
7007     ORGANIZATION "EPCglobal, Inc."
7008     CONTACT-INFO
7009         "      EPCglobal MIB Administrator
7010             GS1/EPCglobal, Inc.
7011             Princeton Pike Corporate Center
7012             1009 Lenox Drive, Suite 202
7013             Lawrenceville, NJ 08648
7014             US
7015
7016             Tel:   +1 609 620 0200
7017             Email: mibs@lists.epcglobalinc.org"
7018     DESCRIPTION
7019         "The MIB Module describing an EPCglobal compliant RFID reader."
7020
7021     REVISION      "200610270000Z"
7022
7023     DESCRIPTION
7024         "Defined in conformance with the EPCglobal Reader Management and
7025         Reader Protocol specifications.
7026
7027         Abbreviations:
7028
7029         DHCP   - Dynamic Host Configuration Protocol
7030         DNS    - Domain Name System
7031         EPC    - Electronic Product Code
7032         URI    - Uniform Resource Identifier
7033         RFID   - Radio Frequency Identification
7034         UTC    - Coordinated Universal Time
7035
7036         Reference EPCglobal Reader Protocol Specification 1.1"

```

```

7037 ::= {epcgSmiManagement 1}
7038
7039
7040 --*****
7041 --** The EPCglobal Reader MIB module sub-trees
7042 --*****
7043 epcgReaderNotifs          OBJECT IDENTIFIER ::= { epcgReaderMIB 0 }
7044 epcgReaderObjects        OBJECT IDENTIFIER ::= { epcgReaderMIB 1 }
7045 epcgReaderConformance    OBJECT IDENTIFIER ::= { epcgReaderMIB 2 }
7046 epcgReaderDevice         OBJECT IDENTIFIER ::= { epcgReaderObjects 1 }
7047 epcgReadPoints           OBJECT IDENTIFIER ::= { epcgReaderObjects 2 }
7048 epcgAntennaReadPoints    OBJECT IDENTIFIER ::= { epcgReaderObjects 3 }
7049 epcgIoPorts              OBJECT IDENTIFIER ::= { epcgReaderObjects 4 }
7050 epcgSources               OBJECT IDENTIFIER ::= { epcgReaderObjects 5 }
7051 epcgNotificationChannels  OBJECT IDENTIFIER ::= { epcgReaderObjects 6 }
7052 epcgTriggers             OBJECT IDENTIFIER ::= { epcgReaderObjects 7 }
7053
7054
7055 --*****
7056 --** Textual Conventions used by EPCglobal Reader MIB modules **
7057 --*****
7058
7059 EpcgCurrentCountSinceReset ::= TEXTUAL-CONVENTION
7060     STATUS current
7061     DESCRIPTION
7062         "A counter that always increments until it is reset."
7063     SYNTAX Gauge32
7064
7065 EpcgNotifLevel ::= TEXTUAL-CONVENTION
7066     STATUS current
7067     DESCRIPTION
7068         "Severity level of an notification."
7069     SYNTAX INTEGER {
7070         emergency (0),
7071         alert (1),
7072         critical (2),
7073         error (3),
7074         warning (4),
7075         notice (5),
7076         informational (6),
7077         debug (7)
7078     }
7079
7080 EpcgOperationalEnable ::= TEXTUAL-CONVENTION
7081     STATUS current
7082     DESCRIPTION
7083         "Each bit represent whether a particular state defined
7084         by the EpcgOperationalStatus is enable for notifications."
7085     SYNTAX BITS { unknown(0),
7086         other (1),
7087         up (2),
7088         down (3)
7089     }
7090
7091 EpcgOperationalStatus ::= TEXTUAL-CONVENTION
7092     STATUS current
7093     DESCRIPTION
7094         "A value representing operational status."
7095     SYNTAX INTEGER { unknown (1),
7096         other (2),
7097         up (3),
7098         down (4)

```

```

7099         }
7100
7101 EpcgThreshold ::= TEXTUAL-CONVENTION
7102     STATUS          current
7103     DESCRIPTION
7104         "This textual convention defines the allowed values
7105         used to define a threshold."
7106     SYNTAX Unsigned32 (0..4294967295)
7107
7108 EpcgTriggerType ::= TEXTUAL-CONVENTION
7109     STATUS          current
7110     DESCRIPTION "The types of allowed triggers."
7111     SYNTAX INTEGER { none (1),
7112                     timer (2),
7113                     continuous (3),
7114                     ioEdge (4),
7115                     vendorExtension (5)
7116                     }
7117
7118
7119
7120 --*****
7121 --** Reader Notifications **
7122 --*****
7123
7124 epcgReaderDeviceOperationState NOTIFICATION-TYPE
7125     OBJECTS { sysName,
7126               epcgRdrDevTimeUtc,
7127               epcgRdrDevOperNotifStateLevel,
7128               epcgRdrDevOperStatusPrior,
7129               epcgRdrDevOperStatus
7130             }
7131     STATUS current
7132     DESCRIPTION
7133         "This notification is generated when a reader device
7134         undergoes a change in operational status. In some
7135         environment certain states may be transitory and
7136         do not need to generate notifications. In addition,
7137         a device may cycle between two states. Therefore,
7138         notifications will be generated whenever the state
7139         transitions into a state enabled for notifications
7140         as defined by the epcgRdrDevOperNotifToState
7141         object. Also, notifications will be generate whenever
7142         the state transitions out of a state define by the
7143         epcgRdrDevOperNotifFromState object. The
7144         epcgRdrDevOperStateSuppressInterval object can be
7145         configured to suppress the excessive generation of
7146         notifications."
7147     ::= { epcgReaderNotifs 1 }
7148
7149 epcgRdrDevMemoryState NOTIFICATION-TYPE
7150     OBJECTS { sysName,
7151               epcgRdrDevTimeUtc,
7152               epcgRdrDevFreeMemoryNotifLevel,
7153               epcgRdrDevFreeMemory
7154             }
7155     STATUS current
7156     DESCRIPTION
7157         "This notification is generated when the reader's
7158         free memory state changes and the following transitions
7159         are possible
7160

```

```

7161     normal -> shortage
7162
7163     A memory shortage is detected when the epcgRdrDevFreeMemory
7164     object changes to a value that is equal to or below the
7165     value specified by the epcgRdrDevFreeMemoryOnsetThreshold
7166     object the object.
7167
7168     shortage -> normal
7169
7170     A memory shortage ends when the epcgRdrDevFreeMemory
7171     object changes to a value that is equal to or greater than
7172     the value specified by the
7173     epcgRdrDevFreeMemoryAbateThreshold object.
7174
7175     The epcgRdrDevFreeMemoryOnsetThreshold and
7176     epcgRdrDevFreeMemoryAbateThreshold threshold will be
7177     different for each platform and should be related to the
7178     amount of available memory. For example, a device may
7179     decided to set the onset threshold to 20 percent of
7180     total memory and the abate to 35 percent of total memory.
7181
7182     The value of the epcgRdrDevFreeMemoryOnsetThreshold
7183     object must be less than the value specified for the
7184     epcgRdrDevFreeMemoryOnsetAbate object. In addition,
7185     the difference between these object should be sufficient
7186     to prevent excessive number of notification being
7187     generated when small amounts of memory are used then
7188     freed. The epcgRdrDevMemStateSuppressInterval object can
7189     be configured to suppress the excessive generation of
7190     notifications."
7191     ::= { epcgReaderNotifs 2 }
7192
7193 epcgReadPointOperationState NOTIFICATION-TYPE
7194     OBJECTS { sysName,
7195               epcgRdrDevTimeUtc,
7196               epcgReadPointOperNotifyStateLevel,
7197               epcgReadPointName,
7198               epcgReadPointPriorOperStatus,
7199               epcgReadPointOperStatus
7200             }
7201     STATUS current
7202     DESCRIPTION
7203         "This notification is generated when a read point
7204         undergoes a change in operational status. In some
7205         environment certain states may be transitory and
7206         do not need to generate notifications. In addition,
7207         a device may cycle between two states. Therefore,
7208         notifications will be generated whenever the state
7209         transitions into a state enabled for notifications
7210         as defined by the epcgRdrDevOperNotifToState
7211         object. Also, notifications will be generate whenever
7212         the state transitions out of a state define by the
7213         epcgRdrDevOperNotifFromState object. The
7214         epcgReadPointOperStateSuppressInterval object can be
7215         configured to suppress the excessive generation of
7216         notifications."
7217     ::= { epcgReaderNotifs 3 }
7218
7219 epcgReaderAntennaReadFailure NOTIFICATION-TYPE
7220     OBJECTS { sysName,
7221               epcgRdrDevTimeUtc,
7222               epcgAntRdPntReadFailureNotifLevel,

```

```

7223         epcgReadPointName,
7224         epcgAntRdPntMemoryReadFailures,
7225         epcgAntRdPntNoiseLevel
7226     }
7227     STATUS current
7228     DESCRIPTION
7229         "This notification is generated when a memory read operation
7230         across an antenna fails. To prevent an excessive number
7231         of notifications the epcgAntRdPntSuppressInterval can be
7232         configured to suppress generation."
7233     ::= { epcgReaderNotifs 4 }
7234
7235     epcgReaderAntennaWriteFailure NOTIFICATION-TYPE
7236     OBJECTS { sysName,
7237         epcgRdrDevTimeUtc,
7238         epcgAntRdPntWriteFailuresNotifLevel,
7239         epcgReadPointName,
7240         epcgAntRdPntWriteFailures,
7241         epcgAntRdPntNoiseLevel
7242     }
7243     STATUS current
7244     DESCRIPTION
7245         "This notification is generated when a write operation
7246         across an antenna fails. To prevent an excessive number
7247         of notifications the epcgAntRdPntSuppressInterval can be
7248         configured to suppress generation"
7249     ::= { epcgReaderNotifs 5 }
7250
7251     epcgReaderAntennaKillFailure NOTIFICATION-TYPE
7252     OBJECTS { sysName,
7253         epcgRdrDevTimeUtc,
7254         epcgAntRdPntKillFailuresNotifLevel,
7255         epcgReadPointName,
7256         epcgAntRdPntKillFailures,
7257         epcgAntRdPntNoiseLevel
7258     }
7259     STATUS current
7260     DESCRIPTION
7261         "This notification is generated when a kill operation
7262         across an antenna fails. To prevent an excessive number
7263         of notifications the epcgAntRdPntSuppressInterval can be
7264         configured to suppress generation"
7265     ::= { epcgReaderNotifs 6 }
7266
7267     epcgReaderAntennaEraseFailure NOTIFICATION-TYPE
7268     OBJECTS { sysName,
7269         epcgRdrDevTimeUtc,
7270         epcgAntRdPntFailEraseFailuresNotifLevel,
7271         epcgReadPointName,
7272         epcgAntRdPntEraseFailures,
7273         epcgAntRdPntNoiseLevel
7274     }
7275     STATUS current
7276     DESCRIPTION
7277         "This notification is generated when an erase operation
7278         across an antenna fails. To prevent an excessive number
7279         of notifications the epcgAntRdPntSuppressInterval can be
7280         configured to suppress generation"
7281     ::= { epcgReaderNotifs 7 }
7282
7283     epcgReaderAntennaLockFailure NOTIFICATION-TYPE
7284     OBJECTS { sysName,

```

```

7285         epcgRdrDevTimeUtc,
7286         epcgAntRdPntLockFailuresNotifLevel,
7287         epcgReadPointName,
7288         epcgAntRdPntLockFailures,
7289         epcgAntRdPntNoiseLevel
7290     }
7291     STATUS current
7292     DESCRIPTION
7293         "This notification is generated when a lock operation
7294         across an antenna fails. To prevent an excessive number
7295         of notifications the epcgAntRdPntSuppressInterval can be
7296         configured to suppress generation"
7297     ::= { epcgReaderNotifs 8 }
7298
7299     epcgReaderIoPortOperationState NOTIFICATION-TYPE
7300     OBJECTS { sysName,
7301         epcgRdrDevTimeUtc,
7302         epcgIoPortOperStatusNotifLevel,
7303         epcgIoPortName,
7304         epcgIoPortOperStatusPrior,
7305         epcgIoPortOperStatus
7306     }
7307     STATUS current
7308     DESCRIPTION
7309         "This notification is generated when an IO port
7310         undergoes a change in operation status that falls
7311         within the set of notification-triggering state transitions
7312         specified by the epcgIoPortOperStatusNotifFromState and
7313         epcgIoPortOperStatusNotifToState object values. The
7314         epcgIoPortOperStateSuppressInterval object can be
7315         configured to suppress the excessive generation of
7316         notifications."
7317     ::= { epcgReaderNotifs 9 }
7318
7319     epcgReaderSourceOperationState NOTIFICATION-TYPE
7320     OBJECTS { sysName,
7321         epcgRdrDevTimeUtc,
7322         epcgSrcOperStatusNotifyLevel,
7323         epcgSrcName,
7324         epcgSrcOperPriorStatus,
7325         epcgSrcOperStatus
7326     }
7327     STATUS current
7328     DESCRIPTION
7329         "This notification is generated when a source
7330         undergoes a change in operational status that
7331         falls within the set of notification-triggering state
7332         transitions specified by the epcgSrcOperStatusNotifFromState
7333         and epcgSrcOperStatusNotifToState object values. The
7334         notifications may be generated by a from state, a to state,
7335         or some combination of from and to states. The
7336         epcgRdrSourceOperStateSuppressInterval object can be
7337         configured to suppress the excessive generation of
7338         notifications."
7339     ::= { epcgReaderNotifs 10 }
7340
7341     epcgReaderNotificationChanOperState NOTIFICATION-TYPE
7342     OBJECTS { sysName,
7343         epcgRdrDevTimeUtc,
7344         epcgNotifChanOperNotifLevel,
7345         epcgNotifChanName,
7346         epcgNotifChanOperStatusPrior,

```

```

7347         epcgNotifChanOperStatus
7348     }
7349     STATUS current
7350     DESCRIPTION
7351         "This notification is generated when a notification
7352         channel undergoes a change in operational status that
7353         falls within the set of notification-triggering state
7354         transitions specified by the
7355         epcgNotifChanOperNotifFromState and
7356         epcgNotifChanOperNotifToState object values. The
7357         notifications may be generated by a from state, a to state, or some
7358         combination of from and to states. To prevent an excessive number
7359         of notifications the epcgRdrChanOperStateSuppressInterval can be
7360         configured to suppress generation"
7361     ::= { epcgReaderNotifs 11 }
7362
7363
7364     --*****
7365     --** Reader Device Information Objects **
7366     --*****
7367     epcgReaderDeviceInformation
7368         OBJECT IDENTIFIER ::= { epcgReaderDevice 1 }
7369
7370     -- Many of the objects contained within the ReaderDevice class of the
7371     -- EPCglobal Reader Management abstract model are represented, within
7372     -- the SNMP binding, by objects within in the System group (SNMPv2-MIB,
7373     -- see RFC 3418). In particular,
7374     --
7375     --     o System group's sysName object represents ReaderDevice.Name
7376     --     o System group's sysDescr object represents ReaderDevice.Manufacturer
7377     --     and ReaderDevice.Model
7378     --     o System group's sysLocation object represents
7379     --     ReaderDevice.LocationDescription
7380     --     o System group's sysContact object represents ReaderDevice.Contact
7381     --     o System group's sysUpTime object represents ReaderDevice.TimeTicks
7382     --
7383     epcgRdrDevDescription      OBJECT-TYPE
7384         SYNTAX                  SnmpAdminString
7385         MAX-ACCESS               read-only
7386         STATUS                   current
7387         DESCRIPTION
7388             "The operator's textual description of the reader."
7389         ::= { epcgReaderDeviceInformation 1 }
7390
7391     epcgRdrDevRole              OBJECT-TYPE
7392         SYNTAX                  SnmpAdminString
7393         MAX-ACCESS               read-only
7394         STATUS                   current
7395         DESCRIPTION
7396             "Reader functional role; e.g., Pallet Verification Portal."
7397         ::= { epcgReaderDeviceInformation 2 }
7398
7399     epcgRdrDevEpc              OBJECT-TYPE
7400         SYNTAX                  SnmpAdminString
7401         MAX-ACCESS               read-only
7402         STATUS                   current
7403         DESCRIPTION
7404             "Manufacturer assigned EPC URI."
7405         ::= { epcgReaderDeviceInformation 3 }
7406
7407     epcgRdrDevSerialNumber      OBJECT-TYPE
7408         SYNTAX                  SnmpAdminString

```

```

7409     MAX-ACCESS          read-only
7410     STATUS              current
7411     DESCRIPTION
7412         "Manufacturer assigned serial number."
7413     ::= { epcgReaderDeviceInformation 4 }
7414
7415     epcgRdrDevTimeUtc    OBJECT-TYPE
7416     SYNTAX              DateAndTime
7417     MAX-ACCESS          read-only
7418     STATUS              current
7419     DESCRIPTION
7420         "Reader device wall clock time in UTC."
7421     ::= { epcgReaderDeviceInformation 5 }
7422
7423     epcgRdrDevCurrentSource OBJECT-TYPE
7424     SYNTAX              RowPointer
7425     MAX-ACCESS          read-only
7426     STATUS              current
7427     DESCRIPTION
7428         "A pointer to the row in the source table(epcgSourceTable)
7429         that corresponds to the current source"
7430     ::= { epcgReaderDeviceInformation 6 }
7431
7432     epcgRdrDevReboot     OBJECT-TYPE
7433     SYNTAX              TruthValue
7434     MAX-ACCESS          read-write
7435     STATUS              current
7436     DESCRIPTION
7437         "Setting this object to true(1) causes the device to reboot.
7438         Reading this object always returns false(2). Care should be taken
7439         that the reboot is done after the response is sent, to prevent the
7440         management station from resending the request and rebooting the device
7441         again."
7442     ::= { epcgReaderDeviceInformation 7 }
7443
7444     epcgRdrDevResetStatistics OBJECT-TYPE
7445     SYNTAX              TruthValue
7446     MAX-ACCESS          read-write
7447     STATUS              current
7448     DESCRIPTION
7449         " Object used to clear all measurements as follows.
7450         'true' - Causes the device to set all objects defined
7451                 with a syntax of EpcgCurrentCountSinceReset
7452                 to zero.
7453
7454         'false' - read value.
7455
7456         Reading this object always returns 'false'."
7457     ::= { epcgReaderDeviceInformation 8 }
7458
7459     epcgRdrDevResetTimestamp OBJECT-TYPE
7460     SYNTAX              DateAndTime
7461     MAX-ACCESS          read-only
7462     STATUS              current
7463     DESCRIPTION
7464         "This object is set to the current date and time when the
7465         epcgRdrDevResetStatistics object is set to 'true'.
7466         Initially it will be set to null value."
7467     ::= { epcgReaderDeviceInformation 9 }
7468
7469     epcgRdrDevNormalizePowerLevel OBJECT-TYPE
7470     SYNTAX              TruthValue

```

```

7471 MAX-ACCESS          read-only
7472 STATUS             current
7473 DESCRIPTION
7474     " This object is used to indicate how an implementation
7475       presents power levels as follows.
7476
7477       'true' - All power levels will be converted be normalize
7478               to a range from 0-255.
7479
7480       'false' - All power levels are presented in raw format
7481                specific to a particular vendor.
7482
7483       This controls the presentation of the following objects.
7484       -- epcgAntRdPntPowerLevel
7485       "
7486       ::= { epcgReaderDeviceInformation 10 }
7487
7488 epcgRdrDevNormalizeNoiseLevel OBJECT-TYPE
7489     SYNTAX             TruthValue
7490     MAX-ACCESS          read-only
7491     STATUS             current
7492     DESCRIPTION
7493         "This object is used to indicate how an implementation
7494           presents noise levels as follows.
7495
7496           'true' - All noise levels will be converted be normalize
7497                   to a range from 0-255.
7498
7499           'false' - All noise levels are presented in raw format
7500                    specific to a particular vendor.
7501
7502           This controls the presentation of the following objects.
7503           -- epcgAntRdPntNoiseLevel
7504           "
7505           ::= { epcgReaderDeviceInformation 11 }
7506
7507
7508 --*****
7509 --** Optional system wide counters
7510 --*****
7511 epcgGlobalCountersTable OBJECT-TYPE
7512     SYNTAX             SEQUENCE OF EpcgGlobalCountersEntry
7513     MAX-ACCESS          not-accessible
7514     STATUS             current
7515     DESCRIPTION
7516         "The following table aggregates the various counters
7517           that are maintained per entity like read point,
7518           antenna read points, and source."
7519         ::= { epcgReaderDevice 2 }
7520
7521 epcgGlobalCountersEntry OBJECT-TYPE
7522     SYNTAX             EpcgGlobalCountersEntry
7523     MAX-ACCESS          not-accessible
7524     STATUS             current
7525     DESCRIPTION
7526         "Each entry contains information need to communicate with
7527           each type of server. "
7528     INDEX { epcgGlobalCountersIndex }
7529     ::= { epcgGlobalCountersTable 1 }
7530
7531 EpcgGlobalCountersEntry ::= SEQUENCE {
7532     epcgGlobalCountersIndex INTEGER,

```

```

7533     epcgGlobalCountersData    EpcgCurrentCountSinceReset
7534     }
7535
7536 epcgGlobalCountersIndex OBJECT-TYPE
7537     SYNTAX INTEGER { antennaTagsIdentified      (1),
7538                   antennaTagsNotIdentified      (2),
7539                   antennaMemoryReadOperations   (18),
7540                   antennaMemoryReadFailures     (3),
7541                   antennaWriteOperations        (4),
7542                   antennaWriteFailures         (5),
7543                   antennaKillOperations         (6),
7544                   antennaKillFailures          (7),
7545                   antennaEraseOperations        (8),
7546                   antennaEraseFailures         (19),
7547                   antennaLockOperations        (9),
7548                   antennaLockFailures         (10),
7549                   sourceUnknownToGlimpsed      (11),
7550                   sourceGlimpsedToUnknown      (12),
7551                   sourceGlimpsedToObserved     (13),
7552                   sourceObservedToLost        (14),
7553                   sourceLostToGlimpsed        (15),
7554                   sourceLostToUnknown         (16),
7555                   triggerMatches              (17)
7556     }
7557     MAX-ACCESS      not-accessible
7558     STATUS          current
7559     DESCRIPTION
7560         "Index to access global counters as follows.  The units of
7561         each counter will be the same as object indicated by
7562         index.  For example the antennaIdentifier value references
7563         the epcgAntRdPntTagsIdentified and it uses 'tags' as units.
7564
7565         antennaTagsIdentified - Aggregate counter for all instances
7566         of the epcgAntRdPntTagsIdentified object.
7567
7568         antennaTagsNotIdentified - Aggregate counter for all
7569         instances of the epcgAntRdPntTagsNotIdentified object.
7570
7571         antennaMemoryReadOperations - Aggregate counter for all instances
7572         of the epcgAntRdPntMemoryReadOperations object.
7573
7574         antennaMemoryReadFailures - Aggregate counter for all instances
7575         of the epcgAntRdPntMemoryReadFailures object.
7576
7577         antennaWriteOperations - Aggregate counter for all instances
7578         of the epcgAntRdPntWriteOperations object.
7579
7580         antennaWriteFailures - Aggregate counter for all instances
7581         of the epcgAntRdPntWriteFailures object.
7582
7583         antennaKillOperations - Aggregate counter for all instances
7584         of the epcgAntRdPntKillOperations object.
7585
7586         antennaKillFailures - Aggregate counter for all instances
7587         of the epcgAntRdPntKillFailures object.
7588
7589         antennaEraseOperations - Aggregate counter for all instances
7590         of the epcgAntRdPntEraseOperations object.
7591
7592         antennaEraseFailures - Aggregate counter for all instances
7593         of the epcgAntRdPntEraseFailures object.
7594

```

```

7595 antennaLockOperations - Aggregate counter for all instances
7596 of the epcgAntRdPntLockOperations object.
7597
7598 antennaLockFailures - Aggregate counter for all instances
7599 of the epcgAntRdPntLockFailures object.
7600
7601 sourceUnknownToGlimpsed - Aggregate counter for all instances
7602 of the epcgSrcUnknownToGlimpsedTrans object.
7603
7604 sourceGlimpsedToUnknown - Aggregate counter for all instances
7605 of the epcgSrcGlimpsedToUnknownTrans object.
7606
7607 sourceGlimpsedToObserved - Aggregate counter for all instances
7608 of the epcgSrcGlimpsedToObservedTrans object.
7609
7610 sourceObservedToLost - Aggregate counter for all instances
7611 of the epcgSrcObservedToLostTrans object.
7612
7613 sourceLostToGlimpsed - Aggregate counter for all instances
7614 of the epcgSrcLostToGlimpsedTrans object.
7615
7616 sourceLostToUnknown - Aggregate counter for all instances
7617 of the epcgSrcLostToUnknownTrans object.
7618
7619 triggerMatches - Aggregate counter for all instances
7620 of the epcgTriggerMatches object.
7621
7622 source - Aggregate counter for all instances
7623 of the object.
7624 "
7625 ::= { epcgGlobalCountersEntry 1 }
7626
7627 epcgGlobalCountersData OBJECT-TYPE
7628     SYNTAX      EpcgCurrentCountSinceReset
7629     MAX-ACCESS  read-only
7630     STATUS      current
7631     DESCRIPTION
7632         "The object contain aggregate count for all instances
7633         of object as defined by epcgGlobalCountersIndex."
7634     ::= { epcgGlobalCountersEntry 2 }
7635
7636 -----
7637 --** Reader Device - Operation Objects
7638 -----
7639 epcgReaderDeviceOperation
7640     OBJECT IDENTIFIER ::= { epcgReaderDevice 3 }
7641
7642 epcgRdrDevOperStatus OBJECT-TYPE
7643     SYNTAX      EpcgOperationalStatus
7644     MAX-ACCESS  read-only
7645     STATUS      current
7646     DESCRIPTION
7647         "Current status of the reader device."
7648     ::= { epcgReaderDeviceOperation 1}
7649
7650 epcgRdrDevOperStatusPrior OBJECT-TYPE
7651     SYNTAX      EpcgOperationalStatus
7652     MAX-ACCESS  accessible-for-notify
7653     STATUS      current
7654     DESCRIPTION
7655         "Status of reader prior to status change that triggered
7656         the epcgReaderDeviceOperationState notification."

```

```

7657 ::= { epcgReaderDeviceOperation 2 }
7658
7659 epcgRdrDevOperStateEnable OBJECT-TYPE
7660 SYNTAX TruthValue
7661 MAX-ACCESS read-write
7662 STATUS current
7663 DESCRIPTION
7664 "This object controls the generation of the
7665 epcgReaderDeviceOperationState notification
7666 as follows.
7667
7668 'true' Indicates that the notification of
7669 operation state changes has been
7670 enabled.
7671
7672 'false' Indicates that the notification of
7673 operation state changes has been
7674 disabled."
7675 DEFVAL { false }
7676 ::= { epcgReaderDeviceOperation 3 }
7677
7678 epcgRdrDevOperNotifFromState OBJECT-TYPE
7679 SYNTAX EpcgOperationalEnable
7680 MAX-ACCESS read-write
7681 STATUS current
7682 DESCRIPTION
7683 "The source state used to control the generation of
7684 the epcgReaderDeviceOperationState notification."
7685 DEFVAL { { unknown, other, up, down} }
7686 ::= { epcgReaderDeviceOperation 4 }
7687
7688 epcgRdrDevOperNotifToState OBJECT-TYPE
7689 SYNTAX EpcgOperationalEnable
7690 MAX-ACCESS read-write
7691 STATUS current
7692 DESCRIPTION
7693 "The destination state used to control the generation of
7694 the epcgReaderDeviceOperationState notification."
7695 DEFVAL { { unknown, other, up, down} }
7696 ::= { epcgReaderDeviceOperation 5 }
7697
7698 epcgRdrDevOperNotifStateLevel OBJECT-TYPE
7699 SYNTAX EpcgNotifLevel
7700 MAX-ACCESS read-write
7701 STATUS current
7702 DESCRIPTION
7703 "The severity level assigned to the
7704 epcgReaderDeviceOperationStatenotification"
7705 DEFVAL { error }
7706 ::= { epcgReaderDeviceOperation 6 }
7707
7708 epcgRdrDevOperStateSuppressInterval OBJECT-TYPE
7709 SYNTAX Unsigned32 (0 | 1..3600 )
7710 UNITS "seconds"
7711 MAX-ACCESS read-write
7712 STATUS current
7713 DESCRIPTION
7714 "The length of the interval used to suppress generation
7715 of epcgReaderDeviceOperationState change notifications.
7716
7717 Notification will not be suppressed when zero value is
7718 specified."

```

```

7719     DEFVAL { 0 }
7720     ::= { epcgReaderDeviceOperation 7 }
7721
7722 epcgRdrDevOperStateSuppressions OBJECT-TYPE
7723     SYNTAX          Counter32
7724     UNITS            "notifications"
7725     MAX-ACCESS       read-only
7726     STATUS           current
7727     DESCRIPTION
7728         "The number of device operation state change
7729         notifications that have been suppressed."
7730     ::= { epcgReaderDeviceOperation 8 }
7731
7732 --*****
7733 --** Reader Device Memory Objects **
7734 --*****
7735 epcgReaderDeviceMemory
7736     OBJECT IDENTIFIER ::= { epcgReaderDevice 4 }
7737
7738 epcgRdrDevFreeMemory      OBJECT-TYPE
7739     SYNTAX          Gauge32 (0..4294967295)
7740     UNITS            "bytes"
7741     MAX-ACCESS       read-only
7742     STATUS           current
7743     DESCRIPTION
7744         "Reader device free memory, in bytes"
7745     ::= { epcgReaderDeviceMemory 1 }
7746
7747 epcgRdrDevFreeMemoryNotifEnable OBJECT-TYPE
7748     SYNTAX          TruthValue
7749     MAX-ACCESS       read-write
7750     STATUS           current
7751     DESCRIPTION
7752         "This object controls the generation of the
7753         epcgRdrDevMemoryState notification as follows.
7754
7755         'true' Indicates that the notification of
7756         epcgRdrDevMemoryState changes has been enabled.
7757
7758         'false' Indicates that the notification of
7759         epcgRdrDevMemoryState changes has been disabled."
7760     ::= { epcgReaderDeviceMemory 2 }
7761
7762 epcgRdrDevFreeMemoryNotifLevel OBJECT-TYPE
7763     SYNTAX          EpcgNotifLevel
7764     MAX-ACCESS       read-write
7765     STATUS           current
7766     DESCRIPTION
7767         "The severity level assigned to the epcgRdrDevMemoryState
7768         notification"
7769     DEFVAL { critical }
7770     ::= { epcgReaderDeviceMemory 3 }
7771
7772 epcgRdrDevFreeMemoryOnsetThreshold OBJECT-TYPE
7773     SYNTAX          EpcgThreshold
7774     MAX-ACCESS       read-write
7775     STATUS           current
7776     DESCRIPTION
7777         "The threshold value used to detect low memory situations."
7778     ::= { epcgReaderDeviceMemory 4 }
7779
7780 epcgRdrDevFreeMemoryAbateThreshold OBJECT-TYPE

```

```

7781      SYNTAX      EpcgThreshold
7782      MAX-ACCESS  read-write
7783      STATUS      current
7784      DESCRIPTION
7785          "The threshold value used to detect end of low memory
7786          situations."
7787      ::= { epcgReaderDeviceMemory 5 }
7788
7789      epcgRdrDevFreeMemoryStatus OBJECT-TYPE
7790      SYNTAX      INTEGER { shortage (1),
7791                          normal (2)
7792                          }
7793      MAX-ACCESS  read-only
7794      STATUS      current
7795      DESCRIPTION
7796          "The current memory state."
7797      ::= { epcgReaderDeviceMemory 6 }
7798
7799      epcgRdrDevMemStateSuppressInterval OBJECT-TYPE
7800      SYNTAX      Unsigned32 (0 | 1..3600 )
7801      UNITS       "seconds"
7802      MAX-ACCESS  read-write
7803      STATUS      current
7804      DESCRIPTION
7805          "The length of the interval used to suppress generation
7806          of epcgRdrDevMemoryState change notifications.
7807
7808          Notification will not be suppressed when zero value is
7809          specified."
7810      DEFVAL { 0 }
7811      ::= { epcgReaderDeviceMemory 7 }
7812
7813      epcgRdrDevMemStateSuppressions OBJECT-TYPE
7814      SYNTAX      Counter32
7815      UNITS       "notifications"
7816      MAX-ACCESS  read-only
7817      STATUS      current
7818      DESCRIPTION
7819          "The number of device memory state change
7820          notifications that have been suppressed."
7821      ::= { epcgReaderDeviceMemory 8 }
7822
7823      --*****
7824      --** Reader Device - Server Table
7825      --*****
7826
7827      epcgReaderServerTable OBJECT-TYPE
7828      SYNTAX      SEQUENCE OF EpcgReaderServerEntry
7829      MAX-ACCESS  not-accessible
7830      STATUS      current
7831      DESCRIPTION
7832          "The following table provide information about the various
7833          servers used by reader device. Entries are added and removed
7834          from this table via epcgReaderServerRowStatus in accordance
7835          with the RowStatus convention."
7836      ::= { epcgReaderDevice 5 }
7837
7838      epcgReaderServerEntry OBJECT-TYPE
7839      SYNTAX      EpcgReaderServerEntry
7840      MAX-ACCESS  not-accessible
7841      STATUS      current
7842      DESCRIPTION

```

```

7843     "Each entry contains information need to communicate with
7844     each type of server. "
7845     INDEX { epcgReaderServerType,
7846             epcgReaderServerNumber
7847     }
7848     ::= { epcgReaderServerTable 1 }
7849
7850 EpcgReaderServerEntry ::= SEQUENCE {
7851     epcgReaderServerType          INTEGER,
7852     epcgReaderServerNumber        Unsigned32,
7853     epcgReaderServerAddressType   InetAddressType,
7854     epcgReaderServerAddress       InetAddress,
7855     epcgReaderServerRowStatus     RowStatus
7856 }
7857
7858 epcgReaderServerType          OBJECT-TYPE
7859     SYNTAX                     INTEGER { dhcp (1),
7860                                     dns (2),
7861                                     time (3)
7862     }
7863     MAX-ACCESS                 not-accessible
7864     STATUS                     current
7865     DESCRIPTION
7866         "The type of server as follows.
7867
7868         dhcp - DHCP server.
7869         dns  - Domain Name System.
7870         time - Time server used to synchronization Reader's
7871               real-time clock.
7872
7873         "
7874     ::= { epcgReaderServerEntry 1 }
7875
7876 epcgReaderServerNumber        OBJECT-TYPE
7877     SYNTAX                     Unsigned32(1..10)
7878     MAX-ACCESS                 not-accessible
7879     STATUS                     current
7880     DESCRIPTION
7881         "The server number. This index is assigned arbitrarily by
7882         the SNMP engine and is not required to be saved over
7883         restarts."
7884     ::= { epcgReaderServerEntry 2 }
7885
7886 epcgReaderServerAddressType   OBJECT-TYPE
7887     SYNTAX                     InetAddressType
7888     MAX-ACCESS                 read-create
7889     STATUS                     current
7890     DESCRIPTION
7891         "The type of Internet address to be used to
7892         communicate with server to collect."
7893     ::= { epcgReaderServerEntry 3 }
7894
7895 epcgReaderServerAddress       OBJECT-TYPE
7896     SYNTAX                     InetAddress
7897     MAX-ACCESS                 read-create
7898     STATUS                     current
7899     DESCRIPTION
7900         "The address of the server and the format of the address
7901         is specified by the epcgReaderServerAddressType object."
7902     ::= { epcgReaderServerEntry 4 }
7903
7904 epcgReaderServerRowStatus     OBJECT-TYPE
7905     SYNTAX                     RowStatus

```

```

7905     MAX-ACCESS      read-create
7906     STATUS          current
7907     DESCRIPTION
7908         "The status of this row."
7909     ::= { epcgReaderServerEntry 5 }
7910
7911
7912 --*****
7913 --** Read Points **
7914 --*****
7915 epcgReadPointTable OBJECT-TYPE
7916     SYNTAX          SEQUENCE OF EpcgReadPointEntry
7917     MAX-ACCESS      not-accessible
7918     STATUS          current
7919     DESCRIPTION
7920         "This table contains a row entry for each of the Reader's
7921         ReadPoints. A ReadPoint can be any physical entity that is
7922         capable of acquiring (or distributing, in the case of tag
7923         programming) item data. A single RF tag reader antenna is
7924         one example of a ReadPoint. ReadPoints are not limited to
7925         antennas; for example, a bar code scanning device is another
7926         example of a ReadPoint. Hence, ReadPoint is the base class
7927         for all physical elements over which EPC data is read or
7928         written.
7929
7930         The AntennaReadPoint class extends this class and is
7931         represented by the AntennaReadPointTable. (A later revision
7932         of this MIB may introduce a BarCodeReadPointTable that
7933         augments this table.)
7934
7935         Typically the number of rows in this table will be equal to
7936         the number of Antenna ports on a reader"
7937     ::= { epcgReadPoints 1 }
7938
7939
7940 epcgReadPointEntry OBJECT-TYPE
7941     SYNTAX          EpcgReadPointEntry
7942     MAX-ACCESS      not-accessible
7943     STATUS          current
7944     DESCRIPTION
7945         "The properties of a read point. SNMP operations can
7946         neither create nor delete rows of this table"
7947     INDEX { epcgReadPointIndex }
7948     ::= { epcgReadPointTable 1 }
7949
7950 EpcgReadPointEntry ::= SEQUENCE {
7951     epcgReadPointIndex          Unsigned32,
7952     epcgReadPointName           SnmpAdminString,
7953     epcgReadPointDescription    SnmpAdminString,
7954     epcgReadPointAdminStatus    EpcgOperationalStatus,
7955     epcgReadPointOperStatus     EpcgOperationalStatus,
7956     epcgReadPointOperStateNotifyEnable TruthValue,
7957     epcgReadPointOperNotifyFromState EpcgOperationalEnable,
7958     epcgReadPointOperNotifyToState EpcgOperationalEnable,
7959     epcgReadPointOperNotifyStateLevel EpcgNotifLevel
7960 }
7961
7962 epcgReadPointIndex OBJECT-TYPE
7963     SYNTAX          Unsigned32(1..2147483647)
7964     MAX-ACCESS      not-accessible
7965     STATUS          current
7966     DESCRIPTION

```

```

7967         "Index used for uniquely identifying a read point within
7968         the scope of a given RFID Reader. This index is assigned
7969         arbitrarily by the SNMP engine and is not required to be
7970         saved over restarts."
7971         ::= { epcgReadPointEntry 1 }
7972
7973     epcgReadPointName OBJECT-TYPE
7974         SYNTAX      SnmpAdminString
7975         MAX-ACCESS   read-only
7976         STATUS       current
7977         DESCRIPTION
7978             "Unique name assigned to read point by device."
7979         ::= { epcgReadPointEntry 2 }
7980
7981     epcgReadPointDescription OBJECT-TYPE
7982         SYNTAX      SnmpAdminString
7983         MAX-ACCESS   read-only
7984         STATUS       current
7985         DESCRIPTION
7986             "A textual description of the read point."
7987         ::= { epcgReadPointEntry 3 }
7988
7989     epcgReadPointAdminStatus OBJECT-TYPE
7990         SYNTAX      EpcgOperationalStatus
7991         MAX-ACCESS   read-write
7992         STATUS       current
7993         DESCRIPTION
7994             "The administratively specified status of the read point."
7995         ::= { epcgReadPointEntry 4 }
7996
7997     epcgReadPointOperStatus OBJECT-TYPE
7998         SYNTAX      EpcgOperationalStatus
7999         MAX-ACCESS   read-only
8000         STATUS       current
8001         DESCRIPTION
8002             "The read point's current operational status"
8003         ::= { epcgReadPointEntry 5 }
8004
8005     epcgReadPointOperStateNotifyEnable OBJECT-TYPE
8006         SYNTAX      TruthValue
8007         MAX-ACCESS   read-write
8008         STATUS       current
8009         DESCRIPTION
8010             "This object controls the generation of the
8011             epcgReadPointOperationState notification
8012             for this read point as follows.
8013
8014             'true' Indicates that the notification of read point
8015             operation state changes has been enabled.
8016
8017             'false' Indicates that the notification of read point
8018             operation state changes has been disable."
8019
8020         ::= { epcgReadPointEntry 6 }
8021
8022     epcgReadPointOperNotifyFromState OBJECT-TYPE
8023         SYNTAX      EpcgOperationalEnable
8024         MAX-ACCESS   read-write
8025         STATUS       current
8026         DESCRIPTION
8027             "The source state used to control the generation of
8028             the epcgReadPointOperationState notification."

```

```

8029     DEFVAL { { unknown, other, up, down} }
8030     ::= { epcgReadPointEntry 7 }
8031
8032 epcgReadPointOperNotifyToState OBJECT-TYPE
8033     SYNTAX          EpcgOperationalEnable
8034     MAX-ACCESS      read-write
8035     STATUS          current
8036     DESCRIPTION
8037         "The destination state used to control the generation of
8038         the epcgReadPointOperationState notification."
8039     DEFVAL { { unknown, other, up, down} }
8040     ::= { epcgReadPointEntry 8 }
8041
8042 epcgReadPointOperNotifyStateLevel OBJECT-TYPE
8043     SYNTAX          EpcgNotifLevel
8044     MAX-ACCESS      read-write
8045     STATUS          current
8046     DESCRIPTION
8047         "The severity level assigned to the
8048         epcgReadPointOperationState notification."
8049     DEFVAL { error }
8050     ::= { epcgReadPointEntry 9 }
8051
8052 -- Scalars associated with epcgReadPointTable
8053
8054 epcgReadPointPriorOperStatus OBJECT-TYPE
8055     SYNTAX          EpcgOperationalStatus
8056     MAX-ACCESS      accessible-for-notify
8057     STATUS          current
8058     DESCRIPTION
8059         "The past status of read-point prior to the new status change
8060         that triggered the epcgReadPointOperationState
8061         notification."
8062     ::= { epcgReadPoints 2 }
8063
8064 epcgReadPointOperStateSuppressInterval OBJECT-TYPE
8065     SYNTAX          Unsigned32 (0 | 1..3600 )
8066     UNITS           "seconds"
8067     MAX-ACCESS      read-write
8068     STATUS          current
8069     DESCRIPTION
8070         "The length of the interval used to suppress generation
8071         of epcgReadPointOperationState change notifications.
8072
8073         Notification will not be suppressed when zero value is
8074         specified."
8075     DEFVAL { 0 }
8076     ::= { epcgReadPoints 3 }
8077
8078 epcgReadPointOperStateSuppressions OBJECT-TYPE
8079     SYNTAX          Counter32
8080     UNITS           "notifications"
8081     MAX-ACCESS      read-only
8082     STATUS          current
8083     DESCRIPTION
8084         "The number of read point operational state change
8085         notifications that have been suppressed."
8086     ::= { epcgReadPoints 4 }
8087
8088
8089 --*****
8090 --** Antenna Read Points **

```

```

8091  --*****
8092  epcgAntReadPointTable OBJECT-TYPE
8093      SYNTAX      SEQUENCE OF EpcgAntReadPointEntry
8094      MAX-ACCESS  not-accessible
8095      STATUS      current
8096      DESCRIPTION
8097          "This table extends the epcgReadPointTable. It contains
8098           one row for each physical antenna, or antenna port, on
8099           a Reader."
8100      ::= { epcgAntennaReadPoints 1 }
8101
8102  epcgAntReadPointEntry OBJECT-TYPE
8103      SYNTAX      EpcgAntReadPointEntry
8104      MAX-ACCESS  not-accessible
8105      STATUS      current
8106      DESCRIPTION
8107          "The properties of an antenna read point.  SNMP
8108           operations can neither create nor delete
8109           rows of this table"
8110
8111      INDEX { epcgReadPointIndex }
8112      ::= { epcgAntReadPointTable 1 }
8113
8114  EpcgAntReadPointEntry ::= SEQUENCE {
8115      epcgAntRdPntTagsIdentified          EpcgCurrentCountSinceReset,
8116      epcgAntRdPntTagsNotIdentified       EpcgCurrentCountSinceReset,
8117      epcgAntRdPntMemoryReadOperations    EpcgCurrentCountSinceReset,
8118      epcgAntRdPntMemoryReadFailures      EpcgCurrentCountSinceReset,
8119      epcgAntRdPntReadFailureNotifEnable  TruthValue,
8120      epcgAntRdPntReadFailureNotifLevel   EpcgNotifLevel,
8121      epcgAntRdPntWriteOperations          EpcgCurrentCountSinceReset,
8122      epcgAntRdPntWriteFailures           EpcgCurrentCountSinceReset,
8123      epcgAntRdPntWriteFailuresNotifEnable TruthValue,
8124      epcgAntRdPntWriteFailuresNotifLevel EpcgNotifLevel,
8125      epcgAntRdPntKillOperations           EpcgCurrentCountSinceReset,
8126      epcgAntRdPntKillFailures            EpcgCurrentCountSinceReset,
8127      epcgAntRdPntKillFailuresNotifEnable TruthValue,
8128      epcgAntRdPntKillFailuresNotifLevel   EpcgNotifLevel,
8129      epcgAntRdPntEraseOperations          EpcgCurrentCountSinceReset,
8130      epcgAntRdPntEraseFailures           EpcgCurrentCountSinceReset,
8131      epcgAntRdPntEraseFailuresNotifEnable TruthValue,
8132      epcgAntRdPntFailEraseFailuresNotifLevel EpcgNotifLevel,
8133      epcgAntRdPntLockOperations           EpcgCurrentCountSinceReset,
8134      epcgAntRdPntLockFailures            EpcgCurrentCountSinceReset,
8135      epcgAntRdPntLockFailuresNotifEnable  TruthValue,
8136      epcgAntRdPntLockFailuresNotifLevel   EpcgNotifLevel,
8137      epcgAntRdPntPowerLevel               Integer32,
8138      epcgAntRdPntNoiseLevel              Integer32,
8139      epcgAntRdPntTimeEnergized            EpcgCurrentCountSinceReset
8140  }
8141
8142  epcgAntRdPntTagsIdentified OBJECT-TYPE
8143      SYNTAX      EpcgCurrentCountSinceReset
8144      UNITS       "tags"
8145      MAX-ACCESS  read-only
8146      STATUS      current
8147      DESCRIPTION
8148          "Count of tags successfully identified by this antenna read point."
8149      ::= { epcgAntReadPointEntry 1 }
8150
8151  epcgAntRdPntTagsNotIdentified OBJECT-TYPE
8152      SYNTAX      EpcgCurrentCountSinceReset

```

```

8153     UNITS          "tags"
8154     MAX-ACCESS    read-only
8155     STATUS        current
8156     DESCRIPTION
8157         "Count of tags failed to be identified by this antenna
8158         read point. If the device does not implement the technology to
8159         unambiguously determine a failed tag singulation, it is acceptable to
8160         leave this value at zero."
8161     ::= { epcgAntReadPointEntry 2 }
8162
8163     epcgAntRdPntMemoryReadOperations OBJECT-TYPE
8164         SYNTAX      EpcgCurrentCountSinceReset
8165         UNITS        "tags"
8166         MAX-ACCESS  read-only
8167         STATUS      current
8168         DESCRIPTION
8169             "Count of tags successfully having their memory read by this
8170             antenna read point."
8171         ::= { epcgAntReadPointEntry 25 }
8172
8173     epcgAntRdPntMemoryReadFailures OBJECT-TYPE
8174         SYNTAX      EpcgCurrentCountSinceReset
8175         UNITS        "tags"
8176         MAX-ACCESS  read-only
8177         STATUS      current
8178         DESCRIPTION
8179             "Count of tags failed to have their memory read by this
8180             antenna read point."
8181         ::= { epcgAntReadPointEntry 3 }
8182
8183     epcgAntRdPntReadFailureNotifEnable OBJECT-TYPE
8184         SYNTAX      TruthValue
8185         MAX-ACCESS  read-write
8186         STATUS      current
8187         DESCRIPTION
8188             "This object controls the generation of the
8189             epcgReaderAntennaReadFailure notification as follows.
8190
8191             'true' Indicates that the notification of antenna read
8192             failure has been enabled.
8193
8194             'false' Indicates that the notification of antenna read
8195             failure changes has been disabled."
8196         ::= { epcgAntReadPointEntry 4 }
8197
8198     epcgAntRdPntReadFailureNotifLevel OBJECT-TYPE
8199         SYNTAX      EpcgNotifLevel
8200         MAX-ACCESS  read-write
8201         STATUS      current
8202         DESCRIPTION
8203             "The severity level assigned to the
8204             epcgReaderAntennaReadFailure notification."
8205         DEFVAL { error }
8206         ::= { epcgAntReadPointEntry 5 }
8207
8208     epcgAntRdPntWriteOperations OBJECT-TYPE
8209         SYNTAX      EpcgCurrentCountSinceReset
8210         UNITS        "write operations"
8211         MAX-ACCESS  read-only
8212         STATUS      current
8213         DESCRIPTION
8214             "Number of writes successfully done by this antenna read point."

```

```

8215 ::= { epcgAntReadPointEntry 6 }
8216
8217 epcgAntRdPntWriteFailures OBJECT-TYPE
8218 SYNTAX      EpcgCurrentCountSinceReset
8219 UNITS       "write operations"
8220 MAX-ACCESS  read-only
8221 STATUS      current
8222 DESCRIPTION
8223     "Number of writes attempted and failed by this antenna
8224     read point."
8225 ::= { epcgAntReadPointEntry 7 }
8226
8227 epcgAntRdPntWriteFailuresNotifEnable OBJECT-TYPE
8228 SYNTAX      TruthValue
8229 MAX-ACCESS  read-write
8230 STATUS      current
8231 DESCRIPTION
8232     "This object controls the generation of the
8233     epcgReaderAntennaWriteFailure notification as follows.
8234
8235     'true' Indicates that the notification of antenna write
8236     failure changes has been enabled.
8237
8238     'false' Indicates that the notification of antenna write
8239     failure changes has been disabled."
8240 ::= { epcgAntReadPointEntry 8 }
8241
8242 epcgAntRdPntWriteFailuresNotifLevel OBJECT-TYPE
8243 SYNTAX      EpcgNotifLevel
8244 MAX-ACCESS  read-write
8245 STATUS      current
8246 DESCRIPTION
8247     "The severity level assigned to the
8248     epcgReaderAntennaWriteFailure notification."
8249 DEFVAL { notice }
8250 ::= { epcgAntReadPointEntry 9 }
8251
8252 epcgAntRdPntKillOperations OBJECT-TYPE
8253 SYNTAX      EpcgCurrentCountSinceReset
8254 UNITS       "kill operations"
8255 MAX-ACCESS  read-only
8256 STATUS      current
8257 DESCRIPTION
8258     "Number of kill operations successfully done by this antenna read
8259     point."
8260 ::= { epcgAntReadPointEntry 10 }
8261
8262 epcgAntRdPntKillFailures OBJECT-TYPE
8263 SYNTAX      EpcgCurrentCountSinceReset
8264 UNITS       "kill operations"
8265 MAX-ACCESS  read-only
8266 STATUS      current
8267 DESCRIPTION
8268     "Number of kill operations attempted and failed by this
8269     antenna read point."
8270 ::= { epcgAntReadPointEntry 11 }
8271
8272 epcgAntRdPntKillFailuresNotifEnable OBJECT-TYPE
8273 SYNTAX      TruthValue
8274 MAX-ACCESS  read-write
8275 STATUS      current
8276 DESCRIPTION

```

```

8277 "This object controls the generation of the
8278     epcgReaderAntennaKillFailure notification as follows.
8279
8280     'true' Indicates that the notification of antenna kill
8281           failure has been enabled.
8282
8283     'false' Indicates that the notification of antenna kill
8284            failure has been disabled."
8285 ::= { epcgAntReadPointEntry 12 }
8286
8287 epcgAntRdPntKillFailuresNotifLevel OBJECT-TYPE
8288     SYNTAX                      EpcgNotifLevel
8289     MAX-ACCESS                  read-write
8290     STATUS                      current
8291     DESCRIPTION
8292         "The severity level assigned to the
8293         epcgReaderAntennaKillFailure notification."
8294     DEFVAL { notice }
8295     ::= { epcgAntReadPointEntry 13 }
8296
8297 epcgAntRdPntEraseOperations OBJECT-TYPE
8298     SYNTAX                      EpcgCurrentCountSinceReset
8299     UNITS                      "erase operations"
8300     MAX-ACCESS                  read-only
8301     STATUS                      current
8302     DESCRIPTION
8303         "Number of erase operations successfully completed by this antenna
8304         read point. "
8305     ::= { epcgAntReadPointEntry 14 }
8306
8307 epcgAntRdPntEraseFailures OBJECT-TYPE
8308     SYNTAX                      EpcgCurrentCountSinceReset
8309     UNITS                      "erase operations"
8310     MAX-ACCESS                  read-only
8311     STATUS                      current
8312     DESCRIPTION
8313         "Number of erase operations attempted and failed by this
8314         antenna read point."
8315     ::= { epcgAntReadPointEntry 15 }
8316
8317 epcgAntRdPntEraseFailuresNotifEnable OBJECT-TYPE
8318     SYNTAX                      TruthValue
8319     MAX-ACCESS                  read-write
8320     STATUS                      current
8321     DESCRIPTION
8322         "This object controls the generation of the
8323         epcgReaderAntennaEraseFailure notification as follows.
8324
8325         'true' Indicates that the notification of antenna erase
8326               failure changes has been enabled.
8327
8328         'false' Indicates that the notification of antenna erase failure
8329               changes has been disabled."
8330     ::= { epcgAntReadPointEntry 16 }
8331
8332 epcgAntRdPntFailEraseFailuresNotifLevel OBJECT-TYPE
8333     SYNTAX                      EpcgNotifLevel
8334     MAX-ACCESS                  read-write
8335     STATUS                      current
8336     DESCRIPTION
8337         "The severity level assigned to the
8338         epcgReaderAntennaEraseFailure notification."

```

```

8339     DEFVAL { notice }
8340     ::= { epcgAntReadPointEntry 17 }
8341
8342     epcgAntRdPntLockOperations OBJECT-TYPE
8343         SYNTAX          EpcgCurrentCountSinceReset
8344         UNITS            "lock operations"
8345         MAX-ACCESS       read-only
8346         STATUS           current
8347         DESCRIPTION
8348             "Number of lock operations successfully issued by this antenna
8349             read point."
8350     ::= { epcgAntReadPointEntry 18 }
8351
8352     epcgAntRdPntLockFailures OBJECT-TYPE
8353         SYNTAX          EpcgCurrentCountSinceReset
8354         UNITS            "lock operations"
8355         MAX-ACCESS       read-only
8356         STATUS           current
8357         DESCRIPTION
8358             "Number of lock operations attempted and failed by this
8359             antenna read point."
8360     ::= { epcgAntReadPointEntry 19 }
8361
8362     epcgAntRdPntLockFailuresNotifEnable OBJECT-TYPE
8363         SYNTAX          TruthValue
8364         MAX-ACCESS       read-write
8365         STATUS           current
8366         DESCRIPTION
8367             "This object controls the generation of the
8368             epcgReaderAntennaLockFailure notification
8369             as follows.
8370
8371             'true' Indicates that the notification of antenna lock
8372             failure changes has been enabled.
8373
8374             'false' Indicates that the notification of antenna lock
8375             failure changes has been disabled."
8376     ::= { epcgAntReadPointEntry 20 }
8377
8378     epcgAntRdPntLockFailuresNotifLevel OBJECT-TYPE
8379         SYNTAX          EpcgNotifLevel
8380         MAX-ACCESS       read-write
8381         STATUS           current
8382         DESCRIPTION
8383             "The severity level assigned to the
8384             epcgReaderAntennaLockFailure notification."
8385     DEFVAL { notice }
8386     ::= { epcgAntReadPointEntry 21 }
8387
8388     epcgAntRdPntPowerLevel OBJECT-TYPE
8389         SYNTAX          Integer32 (-2147483648..2147483647)
8390         MAX-ACCESS       read-only
8391         STATUS           current
8392         DESCRIPTION
8393             "The power level of this antenna read point.
8394             The information provided by this object will be in
8395             different formats depending on the value of the
8396             epcgRdrDevNormalizePowerLevel object. The different
8397             formats are provide in the describe for the
8398             epcgRdrDevNormalizePowerLevel object."
8399     ::= { epcgAntReadPointEntry 22 }
8400

```

```

8401 epcgAntRdPntNoiseLevel OBJECT-TYPE
8402 SYNTAX Integer32 (-2147483648..2147483647)
8403 MAX-ACCESS read-only
8404 STATUS current
8405 DESCRIPTION
8406 "The noise level of this antenna read point.
8407 The information provided by this object will be in
8408 different formats depending on the value of the
8409 epcgRdrDevNormalizePowerLevel object. The different
8410 formats are provide in the describe for the
8411 epcgRdrDevNormalizePowerLevel object."
8412 ::= { epcgAntReadPointEntry 23 }
8413
8414 epcgAntRdPntTimeEnergized OBJECT-TYPE
8415 SYNTAX EpcgCurrentCountSinceReset
8416 UNITS "milli-seconds"
8417 MAX-ACCESS read-only
8418 STATUS current
8419 DESCRIPTION
8420 "The amount of time ,in milli-seconds, that this antenna
8421 read point since system started or value
8422 was cleared."
8423 ::= { epcgAntReadPointEntry 24 }
8424
8425 -- Scalars associated with the epcgAntReadPointTable.
8426
8427 epcgAntRdPntSuppressInterval OBJECT-TYPE
8428 SYNTAX Unsigned32 (0 | 1..3600 )
8429 UNITS "seconds"
8430 MAX-ACCESS read-write
8431 STATUS current
8432 DESCRIPTION
8433 "The length of the interval used to suppress generation
8434 of the following notifications.
8435
8436 epcgReaderAntennaReadFailure
8437 epcgReaderAntennaWriteFailure
8438 epcgReaderAntennaKillFailure
8439 epcgReaderAntennaEraseFailure
8440 epcgReaderAntennaLockFailure
8441
8442 Each antenna can only generate one of these
8443 notifications per the interval specifed by
8444 this object. Here is an example were interval is
8445 set to 15 seconds.
8446
8447 Time Device Error Type Action
8448 -----
8449 09:00 Antenna[1.1] Read Failure generates notification
8450 09:01 Antenna[1.2] Read Failure generates notification
8451 09:05 Antenna[1.1] Read Failure suppress notification
8452 09:06 Antenna[1.1] Write Failure generates notification
8453 09:16 Antenna[1.1] Read Failure - generates notification
8454
8455 Notification will not be suppressed when zero value is
8456 specified."
8457 DEFVAL { 0 }
8458 ::= { epcgAntennaReadPoints 2 }
8459
8460
8461 --*****
8462 --** IO Ports

```

```

8463  --*****
8464  epcgIoPortTable OBJECT-TYPE
8465      SYNTAX      SEQUENCE OF EpcgIoPortEntry
8466      MAX-ACCESS  not-accessible
8467      STATUS      current
8468      DESCRIPTION
8469          "There is a row entry in this table for each data IO port
8470          that the reader supports. Typically these data IO ports
8471          send and/or receive event triggers."
8472      ::= { epcgIoPorts 1 }
8473
8474  epcgIoPortEntry OBJECT-TYPE
8475      SYNTAX      EpcgIoPortEntry
8476      MAX-ACCESS  not-accessible
8477      STATUS      current
8478      DESCRIPTION
8479          "The properties of a single IO port. SNMP operations
8480          can neither create nor delete rows in the epcgIoPortTable"
8481      INDEX { epcgIoPortIndex }
8482      ::= { epcgIoPortTable 1 }
8483
8484  EpcgIoPortEntry ::= SEQUENCE {
8485      epcgIoPortIndex                Unsigned32,
8486      epcgIoPortName                 SnmpAdminString,
8487      epcgIoPortAdminStatus           EpcgOperationalStatus,
8488      epcgIoPortOperStatus            EpcgOperationalStatus,
8489      epcgIoPortOperStatusNotifEnable TruthValue,
8490      epcgIoPortOperStatusNotifLevel EpcgNotifLevel,
8491      epcgIoPortOperStatusNotifFromState EpcgOperationalEnable,
8492      epcgIoPortOperStatusNotifToState EpcgOperationalEnable,
8493      epcgIoPortDescription           SnmpAdminString
8494  }
8495
8496  epcgIoPortIndex OBJECT-TYPE
8497      SYNTAX      Unsigned32(1..2147483647)
8498      MAX-ACCESS  not-accessible
8499      STATUS      current
8500      DESCRIPTION
8501          "Index used to uniquely identifying an IO port within
8502          the scope of a given RFID Reader. This index is assigned
8503          arbitrarily by the SNMP engine and is not required to
8504          be saved over restarts."
8505      ::= { epcgIoPortEntry 1 }
8506
8507  epcgIoPortName OBJECT-TYPE
8508      SYNTAX      SnmpAdminString
8509      MAX-ACCESS  read-only
8510      STATUS      current
8511      DESCRIPTION
8512          "The name of this IO Port."
8513      ::= { epcgIoPortEntry 2 }
8514
8515  epcgIoPortAdminStatus OBJECT-TYPE
8516      SYNTAX      EpcgOperationalStatus
8517      MAX-ACCESS  read-write
8518      STATUS      current
8519      DESCRIPTION
8520          "The administratively specified status of the IO port."
8521      ::= { epcgIoPortEntry 3 }
8522
8523  epcgIoPortOperStatus OBJECT-TYPE
8524      SYNTAX      EpcgOperationalStatus

```

```

8525     MAX-ACCESS      read-only
8526     STATUS          current
8527     DESCRIPTION
8528         "The read point's current operational status."
8529         ::= { epcgIoPortEntry 4 }
8530
8531     epcgIoPortOperStatusNotifEnable OBJECT-TYPE
8532     SYNTAX          TruthValue
8533     MAX-ACCESS      read-write
8534     STATUS          current
8535     DESCRIPTION
8536         "This object controls the generation of the
8537         epcgReaderIoPortOperationState notification
8538         as follows.
8539
8540         'true' Indicates that the notification of IO port operation
8541         state changes has been enabled.
8542
8543         'false' Indicates that the notification of IO port operation
8544         state changes has been disabled."
8545         ::= { epcgIoPortEntry 5 }
8546
8547     epcgIoPortOperStatusNotifLevel OBJECT-TYPE
8548     SYNTAX          EpcgNotifLevel
8549     MAX-ACCESS      read-write
8550     STATUS          current
8551     DESCRIPTION
8552         "The severity level assigned to the
8553         epcgReaderDeviceOperationState notification"
8554     DEFVAL { error }
8555     ::= { epcgIoPortEntry 6 }
8556
8557     epcgIoPortOperStatusNotifFromState OBJECT-TYPE
8558     SYNTAX          EpcgOperationalEnable
8559     MAX-ACCESS      read-write
8560     STATUS          current
8561     DESCRIPTION
8562         "The source state for state transitions triggering
8563         epcgReaderIoPortOperationState notifications"
8564     DEFVAL { { unknown, other, up, down} }
8565     ::= { epcgIoPortEntry 7 }
8566
8567     epcgIoPortOperStatusNotifToState OBJECT-TYPE
8568     SYNTAX          EpcgOperationalEnable
8569     MAX-ACCESS      read-write
8570     STATUS          current
8571     DESCRIPTION
8572         "The destination state for state transitions triggering
8573         epcgReaderIoPortOperationState notifications"
8574     DEFVAL { { unknown, other, up, down} }
8575     ::= { epcgIoPortEntry 8 }
8576
8577     epcgIoPortDescription OBJECT-TYPE
8578     SYNTAX          SnmpAdminString
8579     MAX-ACCESS      read-only
8580     STATUS          current
8581     DESCRIPTION
8582         "The operator's textual description of the IO Port."
8583         ::= { epcgIoPortEntry 9 }
8584
8585     -- Scalars associated with epcgIoPortTable
8586

```

```

8587 epcgIoPortOperStatusPrior OBJECT-TYPE
8588     SYNTAX          EpcgOperationalStatus
8589     MAX-ACCESS      accessible-for-notify
8590     STATUS          current
8591     DESCRIPTION
8592         "Status of IO port prior to status change that
8593         triggered THIS epcgReaderIoPortOperationState."
8594     ::= { epcgIoPorts 2 }
8595
8596 epcgIoPortOperStateSuppressInterval OBJECT-TYPE
8597     SYNTAX          Unsigned32 (0 | 1..3600 )
8598     UNITS           "seconds"
8599     MAX-ACCESS      read-write
8600     STATUS          current
8601     DESCRIPTION
8602         "The length of the interval used to suppress generation
8603         of epcgReaderIoPortOperationState change notifications.
8604
8605         Notification will not be suppressed when zero value is
8606         specified."
8607     DEFVAL { 0 }
8608     ::= { epcgIoPorts 3 }
8609
8610 epcgIoPortOperStateSuppressions OBJECT-TYPE
8611     SYNTAX          Counter32
8612     UNITS           "notifications"
8613     MAX-ACCESS      read-only
8614     STATUS          current
8615     DESCRIPTION
8616         "The number of IO port state change
8617         notifications that have been suppressed."
8618     ::= { epcgIoPorts 4 }
8619
8620 --*****
8621 --** (Logical) Sources (of tag data)
8622 --*****
8623 epcgSourceTable OBJECT-TYPE
8624     SYNTAX          SEQUENCE OF EpcgSourceEntry
8625     MAX-ACCESS      not-accessible
8626     STATUS          current
8627     DESCRIPTION
8628         "This table contains a row for each Source configured
8629         within a Reader. A Source is a logical entity with
8630         zero, one, or more ReadPoints assigned to it. A
8631         Source object may be associated with zero, one or
8632         more ReadTriggers and zero, one or more Notification
8633         Channels."
8634     ::= { epcgSources 1 }
8635
8636 epcgSourceEntry OBJECT-TYPE
8637     SYNTAX          EpcgSourceEntry
8638     MAX-ACCESS      not-accessible
8639     STATUS          current
8640     DESCRIPTION
8641         "The properties of a tag data source. SNMP operations can
8642         neither create nor delete rows of this table."
8643
8644     INDEX { epcgSrcIndex }
8645     ::= { epcgSourceTable 1 }
8646
8647 EpcgSourceEntry ::= SEQUENCE {
8648     epcgSrcIndex          Unsigned32,

```

```

8649     epcgSrcName                               SnmpAdminString,
8650     epcgSrcReadCyclesPerTrigger                Unsigned32,
8651     epcgSrcReadDutyCycle                       Gauge32,
8652     epcgSrcReadTimeout                        Unsigned32,
8653     epcgSrcGlimpsedTimeout                    Unsigned32,
8654     epcgSrcObservedThreshold                  Unsigned32,
8655     epcgSrcObservedTimeout                    Unsigned32,
8656     epcgSrcLostTimeout                        Unsigned32,
8657     epcgSrcUnknownToGlimpsedTrans             EpcgCurrentCountSinceReset,
8658     epcgSrcGlimpsedToUnknownTrans             EpcgCurrentCountSinceReset,
8659     epcgSrcGlimpsedToObservedTrans            EpcgCurrentCountSinceReset,
8660     epcgSrcObservedToLostTrans                EpcgCurrentCountSinceReset,
8661     epcgSrcLostToGlimpsedTrans               EpcgCurrentCountSinceReset,
8662     epcgSrcLostToUnknownTrans                EpcgCurrentCountSinceReset,
8663     epcgSrcAdminStatus                       EpcgOperationalStatus,
8664     epcgSrcOperStatus                        EpcgOperationalStatus,
8665     epcgSrcOperStatusNotifEnable             TruthValue,
8666     epcgSrcOperStatusNotifFromState          EpcgOperationalEnable,
8667     epcgSrcOperStatusNotifToState            EpcgOperationalEnable,
8668     epcgSrcOperStatusNotifyLevel             EpcgNotifLevel,
8669     epcgSrcSupportsWriteOperations            TruthValue
8670 }
8671
8672 epcgSrcIndex OBJECT-TYPE
8673     SYNTAX             Unsigned32 (1..2147483647)
8674     MAX-ACCESS         not-accessible
8675     STATUS             current
8676     DESCRIPTION
8677         "Index used for uniquely identifying a source within the
8678         scope of a given RFID reader. This index is assigned
8679         arbitrarily by the SNMP engine and is not required to
8680         be saved over restarts."
8681     ::= { epcgSourceEntry 1 }
8682
8683 epcgSrcName OBJECT-TYPE
8684     SYNTAX             SnmpAdminString
8685     MAX-ACCESS         read-only
8686     STATUS             current
8687     DESCRIPTION
8688         "Unique name assigned to a source."
8689     ::= { epcgSourceEntry 2 }
8690
8691 epcgSrcReadCyclesPerTrigger OBJECT-TYPE
8692     SYNTAX             Unsigned32
8693     UNITS              "cycles"
8694     MAX-ACCESS         read-write
8695     STATUS             current
8696     DESCRIPTION
8697         "Read cycles attempted per trigger."
8698     ::= { epcgSourceEntry 3 }
8699
8700 epcgSrcReadDutyCycle OBJECT-TYPE
8701     SYNTAX             Gauge32 (0..100)
8702     UNITS              "percentage"
8703     MAX-ACCESS         read-write
8704     STATUS             current
8705     DESCRIPTION
8706         "Duty cycle for this read source."
8707     ::= { epcgSourceEntry 4 }
8708
8709 epcgSrcReadTimeout OBJECT-TYPE
8710     SYNTAX             Unsigned32

```

```

8711     UNITS           "milli-seconds"
8712     MAX-ACCESS      read-write
8713     STATUS          current
8714     DESCRIPTION
8715         "Read timeout in milli-seconds for this source."
8716     ::= { epcgSourceEntry 5 }
8717
8718     epcgSrcGlimpsedTimeout OBJECT-TYPE
8719         SYNTAX      Unsigned32
8720         UNITS       "milli-seconds"
8721         MAX-ACCESS  read-write
8722         STATUS      current
8723         DESCRIPTION
8724             "Glimpsed timeout in milli-seconds for this source."
8725         ::= { epcgSourceEntry 6 }
8726
8727     epcgSrcObservedThreshold OBJECT-TYPE
8728         SYNTAX      Unsigned32
8729         UNITS       "milli-seconds"
8730         MAX-ACCESS  read-write
8731         STATUS      current
8732         DESCRIPTION
8733             "Number of milli-seconds required for tag to be considered
8734             observed for this source"
8735         ::= { epcgSourceEntry 7 }
8736
8737     epcgSrcObservedTimeout OBJECT-TYPE
8738         SYNTAX      Unsigned32
8739         UNITS       "milli-seconds"
8740         MAX-ACCESS  read-write
8741         STATUS      current
8742         DESCRIPTION
8743             "Number of milli-seconds required for tag to be considered
8744             observed for this source"
8745         ::= { epcgSourceEntry 8 }
8746
8747     epcgSrcLostTimeout OBJECT-TYPE
8748         SYNTAX      Unsigned32
8749         UNITS       "milli-seconds"
8750         MAX-ACCESS  read-write
8751         STATUS      current
8752         DESCRIPTION
8753             "Number of milli-seconds required for tag to be considered lost
8754             for this source"
8755         ::= { epcgSourceEntry 9 }
8756
8757     epcgSrcUnknownToGlimpsedTrans OBJECT-TYPE
8758         SYNTAX      EpcgCurrentCountSinceReset
8759         UNITS       "transitions"
8760         MAX-ACCESS  read-only
8761         STATUS      current
8762         DESCRIPTION
8763             "Number of tags transitioning from 'unknown' status to
8764             'glimpsed' status for this source."
8765         ::= { epcgSourceEntry 10 }
8766
8767     epcgSrcGlimpsedToUnknownTrans OBJECT-TYPE
8768         SYNTAX      EpcgCurrentCountSinceReset
8769         UNITS       "transitions"
8770         MAX-ACCESS  read-only
8771         STATUS      current
8772         DESCRIPTION

```

```

8773         "Number of tags transitioning from 'glimpsed' status to
8774         'unknown' status for this source."
8775         ::= { epcgSourceEntry 11 }
8776
8777     epcgSrcGlimpsedToObservedTrans OBJECT-TYPE
8778         SYNTAX      EpcgCurrentCountSinceReset
8779         UNITS        "transitions"
8780         MAX-ACCESS   read-only
8781         STATUS       current
8782         DESCRIPTION
8783             "Number of tags transitioning from 'glimpsed' status
8784             to 'observed' status for this source."
8785         ::= { epcgSourceEntry 12 }
8786
8787     epcgSrcObservedToLostTrans OBJECT-TYPE
8788         SYNTAX      EpcgCurrentCountSinceReset
8789         UNITS        "transitions"
8790         MAX-ACCESS   read-only
8791         STATUS       current
8792         DESCRIPTION
8793             "Number of tags transitioning from 'observed' status
8794             to 'lost' status for this source."
8795         ::= { epcgSourceEntry 13 }
8796
8797     epcgSrcLostToGlimpsedTrans OBJECT-TYPE
8798         SYNTAX      EpcgCurrentCountSinceReset
8799         UNITS        "transitions"
8800         MAX-ACCESS   read-only
8801         STATUS       current
8802         DESCRIPTION
8803             "Number of tags transitioning from 'lost' status to
8804             'glimpsed' status for this source"
8805         ::= { epcgSourceEntry 14 }
8806
8807     epcgSrcLostToUnknownTrans OBJECT-TYPE
8808         SYNTAX      EpcgCurrentCountSinceReset
8809         MAX-ACCESS   read-only
8810         STATUS       current
8811         DESCRIPTION
8812             "Number of tags transitioning from 'lost' status to
8813             'unknown' status for this source"
8814         ::= { epcgSourceEntry 15 }
8815
8816     epcgSrcAdminStatus OBJECT-TYPE
8817         SYNTAX      EpcgOperationalStatus
8818         MAX-ACCESS   read-write
8819         STATUS       current
8820         DESCRIPTION
8821             "The administratively specified status of the source."
8822         ::= { epcgSourceEntry 16 }
8823
8824     epcgSrcOperStatus OBJECT-TYPE
8825         SYNTAX      EpcgOperationalStatus
8826         MAX-ACCESS   read-only
8827         STATUS       current
8828         DESCRIPTION
8829             "The source's current operational status."
8830         ::= { epcgSourceEntry 17 }
8831
8832     epcgSrcOperStatusNotifEnable OBJECT-TYPE
8833         SYNTAX      TruthValue
8834         MAX-ACCESS   read-write

```

```

8835 STATUS          current
8836 DESCRIPTION
8837     "This object controls the generation of the
8838     epcgReaderSourceOperationState notification
8839     as follows.
8840
8841     'true'  Indicates that the notification of Source state
8842             changes has been enabled.
8843
8844     'false' Indicates that the notification of Source state
8845             changes has been disabled."
8846 ::= { epcgSourceEntry 18 }
8847
8848 epcgSrcOperStatusNotifFromState OBJECT-TYPE
8849 SYNTAX          EpcgOperationalEnable
8850 MAX-ACCESS      read-write
8851 STATUS          current
8852 DESCRIPTION
8853     "The source state for state transitions triggering
8854     epcgReaderSourceOperationState notifications"
8855 DEFVAL { { unknown, other, up, down} }
8856 ::= { epcgSourceEntry 19 }
8857
8858 epcgSrcOperStatusNotifToState OBJECT-TYPE
8859 SYNTAX          EpcgOperationalEnable
8860 MAX-ACCESS      read-write
8861 STATUS          current
8862 DESCRIPTION
8863     "The destination state for state transitions triggering
8864     epcgReaderSourceOperationState notifications"
8865 DEFVAL { { unknown, other, up, down} }
8866 ::= { epcgSourceEntry 20 }
8867
8868 epcgSrcOperStatusNotifyLevel OBJECT-TYPE
8869 SYNTAX          EpcgNotifLevel
8870 MAX-ACCESS      read-write
8871 STATUS          current
8872 DESCRIPTION
8873     "The severity level assigned to the
8874     epcgReaderSourceOperationState notification."
8875 DEFVAL { error }
8876 ::= { epcgSourceEntry 21 }
8877
8878 epcgSrcSupportsWriteOperations OBJECT-TYPE
8879 SYNTAX          TruthValue
8880 MAX-ACCESS      read-only
8881 STATUS          current
8882 DESCRIPTION
8883     "This object provides an indication of the capability of the
8884     source as follows.
8885     'true' - source supports read and write operations.
8886     'false' - source supports only read operations."
8887 ::= { epcgSourceEntry 22 }
8888
8889
8890 -- Scalars associated with the epcgSourceTable.
8891
8892 epcgSrcOperPriorStatus OBJECT-TYPE
8893 SYNTAX          EpcgOperationalStatus
8894 MAX-ACCESS      accessible-for-notify
8895 STATUS          current
8896 DESCRIPTION

```

```

8897         "Status of EPC data source prior to status change that
8898         triggered the epcgReaderSourceOperationState
8899         notification."
8900     ::= { epcgSources 2 }
8901
8902 epcgSrcOperStateSuppressInterval OBJECT-TYPE
8903     SYNTAX      Unsigned32 (0 | 1..3600 )
8904     UNITS       "seconds"
8905     MAX-ACCESS  read-write
8906     STATUS      current
8907     DESCRIPTION
8908         "The length of the interval used to suppress generation
8909         of epcgReaderSourceOperationState change notifications.
8910
8911         Notification will not be suppressed when zero value is
8912         specified."
8913     DEFVAL { 0 }
8914     ::= { epcgSources 5 }
8915
8916 epcgSrcOperStateSuppressions OBJECT-TYPE
8917     SYNTAX      Counter32
8918     UNITS       "notifications"
8919     MAX-ACCESS  read-only
8920     STATUS      current
8921     DESCRIPTION
8922         "The number of source operational state change
8923         notifications that have been suppressed."
8924     ::= { epcgSources 6 }
8925
8926 -----
8927 --** Notification Channels
8928 -----
8929 epcgNotificationChannelTable OBJECT-TYPE
8930     SYNTAX      SEQUENCE OF EpcgNotificationChannelEntry
8931     MAX-ACCESS  not-accessible
8932     STATUS      current
8933     DESCRIPTION
8934         "This table contains one row per notification channel.
8935         A notification channel is associated with zero, one
8936         or more Sources, and zero, one Notification Triggers"
8937     ::= { epcgNotificationChannels 1 }
8938
8939 epcgNotificationChannelEntry OBJECT-TYPE
8940     SYNTAX      EpcgNotificationChannelEntry
8941     MAX-ACCESS  not-accessible
8942     STATUS      current
8943     DESCRIPTION
8944         "The properties of a notification channel.  SNMP operations
8945         can neither create nor delete rows of this table"
8946     INDEX { epcgNotifChanIndex }
8947     ::= { epcgNotificationChannelTable 1 }
8948
8949 EpcgNotificationChannelEntry ::= SEQUENCE {
8950     epcgNotifChanIndex      Unsigned32,
8951     epcgNotifChanName       SnmpAdminString,
8952     epcgNotifChanAddressType InetAddressType,
8953     epcgNotifChanAddress    InetAddress,
8954     epcgNotifChanLastAttempt DateAndTime,
8955     epcgNotifChanLastSuccess DateAndTime,
8956     epcgNotifChanAdminStatus EpcgOperationalStatus,
8957     epcgNotifChanOperStatus EpcgOperationalStatus,
8958     epcgNotifChanOperNotifEnable TruthValue,

```

```

8959         epcgNotifChanOperNotifLevel      EpcgNotifLevel,
8960         epcgNotifChanOperNotifFromState   EpcgOperationalEnable,
8961         epcgNotifChanOperNotifToState     EpcgOperationalEnable
8962     }
8963
8964 epcgNotifChanIndex OBJECT-TYPE
8965     SYNTAX      Unsigned32(1..2147483647)
8966     MAX-ACCESS  not-accessible
8967     STATUS      current
8968     DESCRIPTION
8969         "Index used for uniquely identifying a notification channel
8970         within the scope of a given RFID Reader. This index is
8971         assigned arbitrarily by the SNMP engine and is not required
8972         to be saved over restarts."
8973     ::= { epcgNotificationChannelEntry 1 }
8974
8975 epcgNotifChanName OBJECT-TYPE
8976     SYNTAX      SnmpAdminString
8977     MAX-ACCESS  read-only
8978     STATUS      current
8979     DESCRIPTION
8980         "Unique name assigned to Notification Channel."
8981     ::= { epcgNotificationChannelEntry 2 }
8982
8983 epcgNotifChanAddressType OBJECT-TYPE
8984     SYNTAX      InetAddressType
8985     MAX-ACCESS  read-only
8986     STATUS      current
8987     DESCRIPTION
8988         "The type of Internet address used to identify the
8989         notification channel's destination"
8990     ::= { epcgNotificationChannelEntry 3 }
8991
8992 epcgNotifChanAddress OBJECT-TYPE
8993     SYNTAX      InetAddress
8994     MAX-ACCESS  read-only
8995     STATUS      current
8996     DESCRIPTION
8997         "The Internet address identifying the notification
8998         channel's destination."
8999     ::= { epcgNotificationChannelEntry 4 }
9000
9001 epcgNotifChanLastAttempt OBJECT-TYPE
9002     SYNTAX      DateAndTime
9003     MAX-ACCESS  read-only
9004     STATUS      current
9005     DESCRIPTION
9006         "Wall-clock time of last attempted notification by this
9007         notification channel."
9008     ::= { epcgNotificationChannelEntry 5 }
9009
9010 epcgNotifChanLastSuccess OBJECT-TYPE
9011     SYNTAX      DateAndTime
9012     MAX-ACCESS  read-only
9013     STATUS      current
9014     DESCRIPTION
9015         "Wall-clock time of last successful notification sent by
9016         this notification channel."
9017     ::= { epcgNotificationChannelEntry 6 }
9018
9019 epcgNotifChanAdminStatus OBJECT-TYPE
9020     SYNTAX      EpcgOperationalStatus

```

```

9021     MAX-ACCESS      read-write
9022     STATUS          current
9023     DESCRIPTION
9024         "The administratively specified status of the
9025         notification channel."
9026     ::= { epcgNotificationChannelEntry 7 }
9027
9028     epcgNotifChanOperStatus OBJECT-TYPE
9029         SYNTAX          EpcgOperationalStatus
9030         MAX-ACCESS      read-only
9031         STATUS          current
9032         DESCRIPTION
9033             "The notification channel's current operational status."
9034         ::= { epcgNotificationChannelEntry 8 }
9035
9036     epcgNotifChanOperNotifEnable OBJECT-TYPE
9037         SYNTAX          TruthValue
9038         MAX-ACCESS      read-write
9039         STATUS          current
9040         DESCRIPTION
9041             "This object controls the generation of the
9042             epcgReaderNotificationChanOperState notification
9043             as follows.
9044
9045             'true' - Indicates that the sending of notification channel
9046                     operation state changes has been enabled.
9047
9048             'false' - Indicates that the sending of notification channel
9049                     operation state changes has been disabled."
9050         ::= { epcgNotificationChannelEntry 9 }
9051
9052     epcgNotifChanOperNotifLevel OBJECT-TYPE
9053         SYNTAX          EpcgNotifLevel
9054         MAX-ACCESS      read-write
9055         STATUS          current
9056         DESCRIPTION
9057             "The severity level assigned to the
9058             epcgReaderNotificationChanOperState notification."
9059         DEFVAL { error }
9060         ::= { epcgNotificationChannelEntry 10 }
9061
9062     epcgNotifChanOperNotifFromState OBJECT-TYPE
9063         SYNTAX          EpcgOperationalEnable
9064         MAX-ACCESS      read-write
9065         STATUS          current
9066         DESCRIPTION
9067             "The source state for the state transitions triggering
9068             epcgReaderNotificationChanOperState notifications"
9069         DEFVAL { { unknown, other, up, down} }
9070         ::= { epcgNotificationChannelEntry 11 }
9071
9072     epcgNotifChanOperNotifToState OBJECT-TYPE
9073         SYNTAX          EpcgOperationalEnable
9074         MAX-ACCESS      read-write
9075         STATUS          current
9076         DESCRIPTION
9077             "The destination state for the state transitions triggering
9078             epcgReaderNotificationChanOperState notifications"
9079         DEFVAL { { unknown, other, up, down} }
9080         ::= { epcgNotificationChannelEntry 12 }
9081
9082     -- Scalars associated with epcgNotificationChannelTable

```

```

9083
9084 epcgNotifChanOperStatusPrior OBJECT-TYPE
9085     SYNTAX      EpcgOperationalStatus
9086     MAX-ACCESS  accessible-for-notify
9087     STATUS      current
9088     DESCRIPTION
9089         "The status of notification channel prior to the current
9090         status change that triggered the
9091         epcgReaderNotificationChanOperState notification"
9092     ::= { epcgNotificationChannels 2 }
9093
9094 epcgNotifChanOperStateSuppressInterval OBJECT-TYPE
9095     SYNTAX      Unsigned32 (0 | 1..3600 )
9096     UNITS       "seconds"
9097     MAX-ACCESS  read-write
9098     STATUS      current
9099     DESCRIPTION
9100         "The length of the interval used to suppress generation
9101         of epcgReaderNotificationChanOperState change notifications.
9102
9103         Notification will not be suppressed when zero value is
9104         specified."
9105     DEFVAL { 0 }
9106     ::= { epcgNotificationChannels 3 }
9107
9108 epcgNotifChanOperStateSuppressions OBJECT-TYPE
9109     SYNTAX      Counter32
9110     UNITS       "notifications"
9111     MAX-ACCESS  read-only
9112     STATUS      current
9113     DESCRIPTION
9114         "The number of channel operational state change
9115         notifications that have been suppressed."
9116     ::= { epcgNotificationChannels 4 }
9117
9118 --*****
9119 --** Triggers
9120 --*****
9121 epcgTriggerTable OBJECT-TYPE
9122     SYNTAX      SEQUENCE OF EpcgTriggerEntry
9123     MAX-ACCESS  not-accessible
9124     STATUS      current
9125     DESCRIPTION
9126         "This table contains one row per trigger. A
9127         Trigger is associated with zero or one IO Port.
9128         A Trigger may be associated with zero or more Sources
9129         and zero or more Notification Channels"
9130     ::= { epcgTriggers 1 }
9131
9132 epcgTriggerEntry OBJECT-TYPE
9133     SYNTAX      EpcgTriggerEntry
9134     MAX-ACCESS  not-accessible
9135     STATUS      current
9136     DESCRIPTION
9137         "The properties of a trigger. SNMP operations can neither
9138         create nor delete rows in the epcgTriggerTable."
9139     INDEX { epcgTrigIndex }
9140     ::= { epcgTriggerTable 1 }
9141
9142 EpcgTriggerEntry ::= SEQUENCE {
9143     epcgTrigIndex      Unsigned32,
9144     epcgTrigName       SnmpAdminString,

```

```

9145         epcgTrigType          EpcgTriggerType,
9146         epcgTrigParameters     SnmpAdminString,
9147         epcgTriggerMatches     EpcgCurrentCountSinceReset,
9148         epcgTrigIoPort         RowPointer
9149     }
9150
9151     epcgTrigIndex      OBJECT-TYPE
9152     SYNTAX              Unsigned32(1..2147483647)
9153     MAX-ACCESS          not-accessible
9154     STATUS              current
9155     DESCRIPTION
9156         "Index uniquely identifying this trigger. This index is
9157         assigned arbitrarily by the SNMP engine and is not
9158         required to be saved over restarts."
9159     ::= { epcgTriggerEntry 1 }
9160
9161     epcgTrigName        OBJECT-TYPE
9162     SYNTAX              SnmpAdminString
9163     MAX-ACCESS          read-only
9164     STATUS              current
9165     DESCRIPTION
9166         "Name of this trigger."
9167     ::= { epcgTriggerEntry 2 }
9168
9169     epcgTrigType        OBJECT-TYPE
9170     SYNTAX              EpcgTriggerType
9171     MAX-ACCESS          read-only
9172     STATUS              current
9173     DESCRIPTION
9174         "The trigger type."
9175     ::= { epcgTriggerEntry 3 }
9176
9177     epcgTrigParameters  OBJECT-TYPE
9178     SYNTAX              SnmpAdminString
9179     MAX-ACCESS          read-only
9180     STATUS              current
9181     DESCRIPTION
9182         "The epcgTrigParameters value must comply with the constraints
9183         defined for the 'triggervalue' datatype in the EPCglobal
9184         Reader Protocol Specification, 1.1 or greater."
9185     REFERENCE
9186         "EPCglobal Reader Protocol Specification section 6.4"
9187     ::= { epcgTriggerEntry 4 }
9188
9189     epcgTriggerMatches  OBJECT-TYPE
9190     SYNTAX              EpcgCurrentCountSinceReset
9191     UNITS               "occurrences"
9192     MAX-ACCESS          read-only
9193     STATUS              current
9194     DESCRIPTION
9195         "Number of times this trigger has matched."
9196     ::= { epcgTriggerEntry 5 }
9197
9198     epcgTrigIoPort      OBJECT-TYPE
9199     SYNTAX              RowPointer
9200     MAX-ACCESS          read-only
9201     STATUS              current
9202     DESCRIPTION
9203         "A pointer to an entry in the IoPort table. A
9204         trigger need not correspond to an IO port
9205         (e.g., a trigger of type timer has not IO port
9206         association), in which case this

```

```

9207         RowPointer object has the value zeroDotzero"
9208         ::= { epcgTriggerEntry 6 }
9209
9210 -----
9211 --** Notification Trigger Table
9212 -----
9213 epcgNotifTrigTable OBJECT-TYPE
9214     SYNTAX          SEQUENCE OF EpcgNotifTrigEntry
9215     MAX-ACCESS      not-accessible
9216     STATUS          current
9217     DESCRIPTION
9218         "This table associates triggers with notification channels.
9219         Multiple triggers may be associated with the same
9220         notification channel and a single trigger may be associated
9221         with multiple notification channels. There will be one
9222         row entry for each unique pairing of trigger and notification
9223         channel. "
9224     ::= { epcgTriggers 2 }
9225
9226 epcgNotifTrigEntry OBJECT-TYPE
9227     SYNTAX          EpcgNotifTrigEntry
9228     MAX-ACCESS      not-accessible
9229     STATUS          current
9230     DESCRIPTION
9231         "Identifies a single pairing of a trigger and notification
9232         channel. Entries can be added and removed from this
9233         table via epcgNotifTrigRowStatus in accordance with the
9234         RowStatus convention."
9235     INDEX { epcgNotifChanIndex,
9236            epcgTrigIndex }
9237     ::= { epcgNotifTrigTable 1 }
9238
9239 EpcgNotifTrigEntry ::= SEQUENCE {
9240     epcgNotifTrigRowStatus      RowStatus
9241 }
9242
9243 epcgNotifTrigRowStatus OBJECT-TYPE
9244     SYNTAX          RowStatus
9245     MAX-ACCESS      read-create
9246     STATUS          current
9247     DESCRIPTION
9248         "The status of this row."
9249     ::= { epcgNotifTrigEntry 1 }
9250
9251 -----
9252 --** Read Trigger Table
9253 -----
9254 epcgReadTrigTable OBJECT-TYPE
9255     SYNTAX          SEQUENCE OF EpcgReadTrigEntry
9256     MAX-ACCESS      not-accessible
9257     STATUS          current
9258     DESCRIPTION
9259         "This table associates Sources with Triggers. Multiple
9260         triggers may be associated with the same Source and a
9261         single trigger may be associated with multiple Sources.
9262         There will be one row entry for each unique pairing of
9263         Trigger and Source. Entries can be added and removed from
9264         this table via epcgReadTrigRowStatus in accordance with the
9265         RowStatus convention."
9266     ::= { epcgTriggers 3 }
9267
9268 epcgReadTrigEntry OBJECT-TYPE

```

```

9269     SYNTAX          EpcgReadTrigEntry
9270     MAX-ACCESS      not-accessible
9271     STATUS          current
9272     DESCRIPTION
9273         "Identifies a single pairing of a Source and Trigger."
9274         INDEX { epcgSrcIndex,
9275                epcgTrigIndex }
9276     ::= { epcgReadTrigTable 1 }
9277
9278 EpcgReadTrigEntry ::= SEQUENCE {
9279     epcgReadTrigRowStatus          RowStatus
9280 }
9281
9282 epcgReadTrigRowStatus OBJECT-TYPE
9283     SYNTAX          RowStatus
9284     MAX-ACCESS      read-create
9285     STATUS          current
9286     DESCRIPTION
9287         "The status of this row."
9288     ::= { epcgReadTrigEntry 1 }
9289
9290 -----
9291 --** ReadPoint-Source Table
9292 -----
9293 epcgRdPntSrcTable OBJECT-TYPE
9294     SYNTAX          SEQUENCE OF EpcgRdPntSrcEntry
9295     MAX-ACCESS      not-accessible
9296     STATUS          current
9297     DESCRIPTION
9298         "This table associates ReadPoints with (tag) Sources.
9299         Multiple ReadPoints may be associated with the same
9300         Source and a single ReadPoint may be associated with
9301         multiple Sources. There will be one row entry for
9302         each unique pairing of ReadPoint and Source"
9303     ::= { epcgSources 3 }
9304
9305 epcgRdPntSrcEntry OBJECT-TYPE
9306     SYNTAX          EpcgRdPntSrcEntry
9307     MAX-ACCESS      not-accessible
9308     STATUS          current
9309     DESCRIPTION
9310         "Identifies a single pairing of a Read Point and Source."
9311         INDEX { epcgReadPointIndex,
9312                epcgSrcIndex }
9313     ::= { epcgRdPntSrcTable 1 }
9314
9315 EpcgRdPntSrcEntry ::= SEQUENCE {
9316     epcgRdPntSrcRowStatus          RowStatus
9317 }
9318
9319 epcgRdPntSrcRowStatus OBJECT-TYPE
9320     SYNTAX          RowStatus
9321     MAX-ACCESS      read-create
9322     STATUS          current
9323     DESCRIPTION
9324         "The status of this row."
9325     ::= { epcgRdPntSrcEntry 1 }
9326
9327 -----
9328 --** Notification Channel - Source Table
9329 -----
9330

```

```

9331 epcgNotifChanSrcTable OBJECT-TYPE
9332     SYNTAX          SEQUENCE OF EpcgNotifChanSrcEntry
9333     MAX-ACCESS      not-accessible
9334     STATUS          current
9335     DESCRIPTION
9336         "This table associates Notification Channels with
9337         (tag) Sources. Multiple Notification Channels may be
9338         associated with the same Source and a single Notification
9339         Channel may be associated with multiple Sources. There
9340         will be one row entry for each unique pairing of
9341         Notification Channel and Source."
9342     ::= { epcgSources 4 }
9343
9344 epcgNotifChanSrcEntry OBJECT-TYPE
9345     SYNTAX          EpcgNotifChanSrcEntry
9346     MAX-ACCESS      not-accessible
9347     STATUS          current
9348     DESCRIPTION
9349         "Identifies a single pairing of a Notificaiton
9350         Channel and Source.
9351
9352         Entries can be added and removed from
9353         this table via epcgNotifChanSrcRowStatus in accordance with the
9354         RowStatus convention."
9355     INDEX { epcgNotifChanIndex,
9356             epcgSrcIndex }
9357     ::= { epcgNotifChanSrcTable 1 }
9358
9359 EpcgNotifChanSrcEntry ::= SEQUENCE {
9360     epcgNotifChanSrcRowStatus      RowStatus
9361 }
9362
9363 epcgNotifChanSrcRowStatus OBJECT-TYPE
9364     SYNTAX          RowStatus
9365     MAX-ACCESS      read-create
9366     STATUS          current
9367     DESCRIPTION
9368         "The status of this row."
9369     ::= { epcgNotifChanSrcEntry 1 }
9370
9371 --*****
9372 --* Conformance definitions
9373 --*****
9374 epcgReaderGroups          OBJECT IDENTIFIER ::= { epcgReaderConformance 1 }
9375 epcgReaderCompliances     OBJECT IDENTIFIER ::= { epcgReaderConformance 2 }
9376
9377 --*****
9378 --* Compliances statements
9379 --*****
9380 epcgReaderStandardCompliance MODULE-COMPLIANCE
9381     STATUS          current
9382     DESCRIPTION
9383         "This define the initial standard implementation requirements
9384         for an EPCglobal-compliant Reader.
9385
9386         The following groups are optional.
9387         epcgReaderGlobalMeasurementsGroup
9388         epcgReaderAntennaSuppressNotifyGroup
9389         epcgReaderNotifGroup
9390         epcgReaderMemoryGroup
9391         epcgReaderServerInfoGroup
9392

```

```

9393         epcgReaderIOPortGroup
9394         epcgReaderNotifChannelGroup
9395         epcgReaderTriggerGroup
9396         epcgReaderTableAssocGroup
9397     "
9398 MODULE      -- This module
9399     MANDATORY-GROUPS { epcgReaderDeviceInformationGroup,
9400                        epcgReaderOperationGroup,
9401                        epcgReadPointGroup,
9402                        epcgReaderAntennaReadPointGroup,
9403                        epcgReaderSourceGroup,
9404                        epcgReaderMandatoryTableAssocGroup
9405                        }
9406
9407 -- *** epcgSourceTable *****
9408 OBJECT      epcgSrcReadCyclesPerTrigger
9409 MIN-ACCESS  read-only
9410 DESCRIPTION "Write access is not required."
9411
9412 OBJECT      epcgSrcReadDutyCycle
9413 MIN-ACCESS  read-only
9414 DESCRIPTION "Write access is not required."
9415
9416 OBJECT      epcgSrcReadTimeout
9417 MIN-ACCESS  read-only
9418 DESCRIPTION "Write access is not required."
9419
9420 OBJECT      epcgSrcGlimpsedTimeout
9421 MIN-ACCESS  read-only
9422 DESCRIPTION "Write access is not required."
9423
9424 OBJECT      epcgSrcObservedThreshold
9425 MIN-ACCESS  read-only
9426 DESCRIPTION "Write access is not required."
9427
9428 OBJECT      epcgSrcObservedTimeout
9429 MIN-ACCESS  read-only
9430 DESCRIPTION "Write access is not required."
9431
9432 OBJECT      epcgSrcLostTimeout
9433 MIN-ACCESS  read-only
9434 DESCRIPTION "Write access is not required."
9435
9436 ::= { epcgReaderCompliances 1 }
9437
9438 -----
9439 --** Units of conformance
9440 -----
9441 epcgReaderNotifGroup NOTIFICATION-GROUP
9442     NOTIFICATIONS { epcgReaderDeviceOperationState,
9443                    epcgRdrDevMemoryState,
9444                    epcgReadPointOperationState,
9445                    epcgReaderAntennaReadFailure,
9446                    epcgReaderAntennaWriteFailure,
9447                    epcgReaderAntennaKillFailure,
9448                    epcgReaderAntennaEraseFailure,
9449                    epcgReaderAntennaLockFailure,
9450                    epcgReaderIoPortOperationState,
9451                    epcgReaderSourceOperationState,
9452                    epcgReaderNotificationChanOperState
9453                    }
9454     STATUS      current

```

```

9455 DESCRIPTION
9456     "Reader device notifications."
9457 ::= { epcgReaderGroups 1 }
9458
9459
9460 epcgReaderGlobalMeasurementsGroup OBJECT-GROUP
9461     OBJECTS      { epcgGlobalCountersData }
9462     STATUS        current
9463     DESCRIPTION
9464         "Reader device global measurements objects."
9465     ::= { epcgReaderGroups 2 }
9466
9467
9468 epcgReaderDeviceInformationGroup OBJECT-GROUP
9469     OBJECTS      { epcgRdrDevDescription,
9470                   epcgRdrDevRole,
9471                   epcgRdrDevEpc,
9472                   epcgRdrDevSerialNumber,
9473                   epcgRdrDevTimeUtc,
9474                   epcgRdrDevCurrentSource,
9475                   epcgRdrDevReboot,
9476                   epcgRdrDevResetStatistics,
9477                   epcgRdrDevResetTimestamp,
9478                   epcgRdrDevNormalizePowerLevel,
9479                   epcgRdrDevNormalizeNoiseLevel
9480                 }
9481     STATUS        current
9482     DESCRIPTION
9483         "Reader device information objects"
9484     ::= { epcgReaderGroups 3 }
9485
9486
9487 epcgReaderOperationGroup OBJECT-GROUP
9488     OBJECTS      { epcgRdrDevOperStatus,
9489                   epcgRdrDevOperStatusPrior,
9490                   epcgRdrDevOperStateEnable,
9491                   epcgRdrDevOperNotifFromState,
9492                   epcgRdrDevOperNotifToState,
9493                   epcgRdrDevOperNotifStateLevel,
9494                   epcgRdrDevOperStateSuppressInterval,
9495                   epcgRdrDevOperStateSuppressions
9496                 }
9497     STATUS        current
9498     DESCRIPTION
9499         "Reader device operation objects."
9500     ::= { epcgReaderGroups 4 }
9501
9502
9503 epcgReaderMemoryGroup OBJECT-GROUP
9504     OBJECTS      { epcgRdrDevFreeMemory,
9505                   epcgRdrDevFreeMemoryNotifEnable,
9506                   epcgRdrDevFreeMemoryNotifLevel,
9507                   epcgRdrDevFreeMemoryOnsetThreshold,
9508                   epcgRdrDevFreeMemoryAbateThreshold,
9509                   epcgRdrDevFreeMemoryStatus,
9510                   epcgRdrDevMemStateSuppressInterval,
9511                   epcgRdrDevMemStateSuppressions
9512                 }
9513     STATUS        current
9514     DESCRIPTION
9515         "This mandatory group defines Device-level objects"
9516     ::= { epcgReaderGroups 5 }

```

```

9517
9518
9519 epcgReaderServerInfoGroup OBJECT-GROUP
9520     OBJECTS      { epcgReaderServerAddressType,
9521                   epcgReaderServerAddress,
9522                   epcgReaderServerRowStatus
9523                   }
9524     STATUS        current
9525     DESCRIPTION
9526         "Reader device Server objects."
9527     ::= { epcgReaderGroups 6 }
9528
9529
9530 epcgReadPointGroup OBJECT-GROUP
9531     OBJECTS      { epcgReadPointName,
9532                   epcgReadPointDescription,
9533                   epcgReadPointAdminStatus,
9534                   epcgReadPointOperStatus,
9535                   epcgReadPointOperStateNotifyEnable,
9536                   epcgReadPointOperNotifyFromState,
9537                   epcgReadPointOperNotifyToState,
9538                   epcgReadPointOperNotifyStateLevel,
9539                   epcgReadPointPriorOperStatus,
9540                   epcgReadPointOperStateSuppressInterval,
9541                   epcgReadPointOperStateSuppressions
9542                   }
9543     STATUS        current
9544     DESCRIPTION
9545         "Reader - Read Point Objects."
9546     ::= { epcgReaderGroups 7 }
9547
9548 epcgReaderAntennaReadPointGroup OBJECT-GROUP
9549     OBJECTS      { epcgAntRdPntTagsIdentified,
9550                   epcgAntRdPntTagsNotIdentified,
9551                   epcgAntRdPntMemoryReadFailures,
9552                   epcgAntRdPntReadFailureNotifEnable,
9553                   epcgAntRdPntReadFailureNotifLevel,
9554                   epcgAntRdPntWriteOperations,
9555                   epcgAntRdPntWriteFailures,
9556                   epcgAntRdPntWriteFailuresNotifEnable,
9557                   epcgAntRdPntWriteFailuresNotifLevel,
9558                   epcgAntRdPntKillOperations,
9559                   epcgAntRdPntKillFailures,
9560                   epcgAntRdPntKillFailuresNotifEnable,
9561                   epcgAntRdPntKillFailuresNotifLevel,
9562                   epcgAntRdPntEraseOperations,
9563                   epcgAntRdPntEraseFailures,
9564                   epcgAntRdPntEraseFailuresNotifEnable,
9565                   epcgAntRdPntFailEraseFailuresNotifLevel,
9566                   epcgAntRdPntLockOperations,
9567                   epcgAntRdPntLockFailures,
9568                   epcgAntRdPntLockFailuresNotifEnable,
9569                   epcgAntRdPntLockFailuresNotifLevel,
9570                   epcgAntRdPntPowerLevel,
9571                   epcgAntRdPntNoiseLevel,
9572                   epcgAntRdPntTimeEnergized,
9573                   epcgAntRdPntMemoryReadOperations
9574                   }
9575     STATUS        current
9576     DESCRIPTION
9577         "Reader - Antenna Read Point objects."
9578     ::= { epcgReaderGroups 8 }

```

```

9579
9580
9581 epcgReaderAntennaSuppressNotifyGroup OBJECT-GROUP
9582     OBJECTS      { epcgAntRdPntSuppressInterval }
9583     STATUS       current
9584     DESCRIPTION
9585         "Reader - Antenna Read Point suppress notifications."
9586     ::= { epcgReaderGroups 9 }
9587
9588
9589 epcgReaderIOPortGroup OBJECT-GROUP
9590     OBJECTS      { epcgIoPortName,
9591                   epcgIoPortAdminStatus,
9592                   epcgIoPortOperStatus,
9593                   epcgIoPortOperStatusNotifEnable,
9594                   epcgIoPortOperStatusNotifFromState,
9595                   epcgIoPortOperStatusNotifToState,
9596                   epcgIoPortOperStatusNotifLevel,
9597                   epcgIoPortDescription,
9598                   epcgIoPortOperStatusPrior,
9599                   epcgIoPortOperStateSuppressInterval,
9600                   epcgIoPortOperStateSuppressions
9601                 }
9602     STATUS       current
9603     DESCRIPTION
9604         "Reader IO Port objects."
9605     ::= { epcgReaderGroups 10 }
9606
9607 epcgReaderSourceGroup OBJECT-GROUP
9608     OBJECTS      { epcgSrcName,
9609                   epcgSrcReadCyclesPerTrigger,
9610                   epcgSrcReadDutyCycle,
9611                   epcgSrcReadTimeout,
9612                   epcgSrcGlimpsedTimeout,
9613                   epcgSrcObservedThreshold,
9614                   epcgSrcObservedTimeout,
9615                   epcgSrcLostTimeout,
9616                   epcgSrcUnknownToGlimpsedTrans,
9617                   epcgSrcGlimpsedToUnknownTrans,
9618                   epcgSrcGlimpsedToObservedTrans,
9619                   epcgSrcObservedToLostTrans,
9620                   epcgSrcLostToGlimpsedTrans,
9621                   epcgSrcLostToUnknownTrans,
9622                   epcgSrcAdminStatus,
9623                   epcgSrcOperStatus,
9624                   epcgSrcOperStatusNotifEnable,
9625                   epcgSrcOperStatusNotifFromState,
9626                   epcgSrcOperStatusNotifToState,
9627                   epcgSrcOperStatusNotifyLevel,
9628                   epcgSrcSupportsWriteOperations,
9629                   epcgSrcOperPriorStatus,
9630                   epcgSrcOperStateSuppressInterval,
9631                   epcgSrcOperStateSuppressions
9632                 }
9633     STATUS       current
9634     DESCRIPTION
9635         "Reader (Tag Data) Source objects."
9636     ::= { epcgReaderGroups 11 }
9637
9638
9639 epcgReaderNotifChannelGroup OBJECT-GROUP
9640     OBJECTS      { epcgNotifChanName,

```

```

9641         epcgNotifChanAddressType,
9642         epcgNotifChanAddress,
9643         epcgNotifChanLastAttempt,
9644         epcgNotifChanLastSuccess,
9645         epcgNotifChanAdminStatus,
9646         epcgNotifChanOperStatus,
9647         epcgNotifChanOperNotifEnable,
9648         epcgNotifChanOperNotifLevel,
9649         epcgNotifChanOperNotifFromState,
9650         epcgNotifChanOperNotifToState,
9651         epcgNotifChanOperStatusPrior,
9652         epcgNotifChanOperStateSuppressInterval,
9653         epcgNotifChanOperStateSuppressions
9654     }
9655     STATUS          current
9656     DESCRIPTION
9657         "Notification Channel objects"
9658     ::= { epcgReaderGroups 12 }
9659
9660     epcgReaderTriggerGroup OBJECT-GROUP
9661     OBJECTS          { epcgTrigName,
9662                       epcgTrigType,
9663                       epcgTrigParameters,
9664                       epcgTriggerMatches,
9665                       epcgTrigIoPort
9666                     }
9667     STATUS          current
9668     DESCRIPTION
9669         "Reader Trigger objects."
9670     ::= { epcgReaderGroups 13 }
9671
9672     epcgReaderTableAssocGroup OBJECT-GROUP
9673     OBJECTS          { epcgNotifTrigRowStatus,
9674                       epcgReadTrigRowStatus,
9675                       epcgNotifChanSrcRowStatus
9676                     }
9677     STATUS          current
9678     DESCRIPTION
9679         "Contains objects that maintain associations between rows
9680         in different tables."
9681     ::= { epcgReaderGroups 14 }
9682
9683     epcgReaderMandatoryTableAssocGroup OBJECT-GROUP
9684     OBJECTS          { epcgRdPntSrcRowStatus
9685                     }
9686     STATUS          current
9687     DESCRIPTION
9688         "Contains objects that maintain associations between rows
9689         in different tables."
9690     ::= { epcgReaderGroups 15 }
9691
9692     END
9693

```

11 Acknowledgements

The authors thank the many contributions of members of the Working Group during the definition and review of this document.

12 References

- [RP1] “EPCglobal Reader Protocol, Version 1.1,” EPCglobal, <http://XXX>.
- [MIBII] “Management Information Base for Network Management of TCP/IP-based internets: MIB-II,” <http://www.ietf.org/rfc/rfc1213.txt?number=1213>
- [ISODir2] ISO, “Rules for the structure and drafting of International Standards (ISO/IEC Directives, Part2, 2001, 4th edition),” July 2002.
- [SYSLOG] “The syslog Protocol” IETF draft.
<http://www.ietf.cnri.reston.va.us/internet-drafts/draft-ietf-syslog-protocol-15.txt>