Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2017

S. Vallin
Stefan Vallin AB
M. Bjorklund
Cisco

October 5, 2016

YANG Alarm Module
draft-vallin-netmod-alarm-module-00

Abstract

This document defines a YANG module for alarm management. It includes functions for alarm list management, alarm shelving and notifications to inform management systems. There are also RPCs to manage the operator state of an alarm and administrative alarm procedures. The module carefully maps to relevant alarm standards.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 8, 2017.

Table of Contents

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

This document defines a YANG [RFC7950] module for alarm management. The purpose is to define a standardised alarm interface for network devices that can be easily integrated into management applications. The model is also applicable as a northbound alarm interface in the management applications.

Alarm monitoring is a fundamental part of monitoring the network. Raw alarms from devices do not always tell the status of the network services or necessarily point to the root cause. However, being able to feed alarms to the network management system in a standardised format is a starting point for performing higher level network assurance tasks.

The telecommunication domain has standardised an alarm interface in ITU-T X.733 [X.733]. This continued in mobile networks within the 3GPP organisation [ALARMIRP]. Although SNMP is the dominant mechanism for monitoring devices, IETF did not early on standardise an alarm MIB. Instead, management systems interpreted the enterprise specific traps per MIB and device to build an alarm list. When finally The Alarm MIB [RFC3877] was published, it had to address the existence of enterprise traps and map these into alarms. This requirement led to a MIB that is not easy to use.

This document defines a standardised YANG module for alarm management. The design of the module is based on experience from using and implementing the above mentioned alarm standards.

2.1. Terminology

The following terms are defined in [RFC7950]:

- action
- client
- data tree
- RPC
- server

The following terms are used within this document:

- Alarm (the general concept): An alarm signifies an undesirable state in a resource that requires corrective action.
- Alarm Instance: The alarm state for a specific resource and alarm type. For example (GigabitEthernet0/15, link-alarm). An entry in the alarm list.
- Alarm Inventory: A list of all possible alarm types on a system.
- Alarm Shelving: Blocking alarms according to specific criteria.
- Alarm Type: An alarm type identifies a possible unique alarm state for a resource. Alarm types are names to identify the state like 'link-alarm', 'jitter-violation', 'high-disk-utilization'.
- Management System: The alarm management application that consumes the alarms, i.e., acts as a client.
- Resource: A fine-grained identification of the alarming resource, for example: an interface, a process.
- System: The system that implements this YANG alarm module, i.e., acts as a server. This corresponds to a network device or a management application that provides a north-bound alarm interface.

3. Objectives

The objectives for the design of the Alarm Module are:

- Simple to use. If a system supports this module, it shall be straight-forward to integrate this into a YANG based alarm manager.
- View alarms as states on resources and not as discrete notifications.
- Clear definition of "alarm" in order to exclude general events that should not be forwarded as alarm notifications.
- Clear and precise identification of alarm types and alarm instances.
- A management system should be able to pull all available alarm types from a system, i.e., read the alarm inventory from a system.

This makes it possible to prepare alarm operators with corresponding alarm instructions.

- Address alarm usability requirements. While IETF has not really addressed alarm management, telecom standards has addressed it purely from a protocol perspective. The process industry has published several relevant standards addressing requirements for a useful alarm interface; [EEMUA], [ISA182]. This alarm module defines usability requirements as well as a YANG data model.

- Mapping to X.733, which is a requirement for many alarm systems. Still, keep some of the X.733 concepts out of the core model in order to make the model small and easy to understand.

4. Background and Usability Requirements

Common alarm problems and the cause of the problems are summarised in Table 1. This summary is adopted to networking based on the ISA [ISA182] and EEMUA [EEMUA] standards.

Problem	Cause	How this module address the cause
Alarms are generated but they are ignored by the operator.	"Nuisance" alarms (chattering alarms and fleeting alarms), faulty hardware, redundant alarms, cascading alarms, incorrect alarm settings, alarms have not been rationalised, the alarms represent log information rather than true alarms.	Strict definition of alarms requiring corrective response. Alarm requirements in Table 2.
When alarms occur, operators do not know how to respond.	Insufficient alarm response procedures and not well defined alarm types.	The alarm inventory lists all alarm types and corrective actions. Alarm requirements in Table 2.
The alarm display is full of alarms, even when there is nothing wrong.	Nuisance alarms, stale alarms, alarms from equipment not in service.	The alarm definition and alarm shelving.
During a failure, operators are flooded with so many alarms that they do not know which ones are the most important.	Incorrect prioritization of alarms. Not using advanced alarm techniques (e.g. state-based alarming).	State-based alarm model, alarm rate requirements in Table 3 and Table 4

Table 1: Alarm Problems and Causes

Based upon the above problems EEMUA gives the following definition of a good alarm:

Characteristic	Explanation
Relevant	Not spurious or of low operational value.
Unique	Not duplicating another alarm.
Timely	Not long before any response is needed or too late to do anything.
Prioritised	Indicating the importance that the operator deals with the problem.
Understandable	Having a message which is clear and easy to understand.
Diagnostic	Identifying the problem that has occurred.
Advisory	Indicative of the action to be taken.
Focusing	Drawing attention to the most important issues.

Table 2: Definition of a Good Alarm

Vendors SHOULD rationalise all alarms according to above. Another crucial requirement is acceptable alarm rates. Vendors SHOULD make sure that they do not exceed the recommendations from EEMUA below:

Long Term Alarm Rate in Steady Operation	Acceptability
More than one per minute	Very likely to be unacceptable.
One per 2 minutes	Likely to be over-demanding.
One per 5 minutes	Manageable.
Less than one per 10 minutes	Very likely to be acceptable.

Table 3: Acceptable Alarm Rates, Steady State

Number of alarms displayed in 10 minutes following a major network problem	Acceptability
More than 100	Definitely excessive and very likely to lead to the operator to abandon the use of the alarm system.
20-100	Hard to cope with.
Under 10	Should be manageable - but may be difficult if several of the alarms require a complex operator response.

Table 4: Acceptable Alarm Rates, Burst

The numbers in Table 3 and Table 4 are the sum of all alarms for a network being managed from one alarm console. So every individual system or NMS contributes to these numbers.

Vendors SHOULD make sure that the following rules are used in designing the alarm interface:

1. Rationalize the alarms in the system to ensure that every alarm is necessary, has a purpose, and follows the cardinal rule - that it requires an operator response. Adheres to the rules of Table 2

2. Audit the quality of the alarms. Talk with the operators about how well the alarm information support them. Do they know what to do in the event of an alarm? Are they able to quickly diagnose the problem and determine the corrective action? Does the alarm text adhere to the requirements in Table 2?

3. Analyze and benchmark the performance of the system and compare it to the recommended metrics in Table 3 and Table 4. Start by identifying nuisance alarms, standing alarms at normal state and startup.

5. Alarm Concepts

This section defines the fundamental concepts behind the data model. This section is rooted in the works of Vallin et. al [ALARMSEM].

5.1. What is an Alarm?

There are two misconceptions regarding alarms and alarm interfaces that are important to sort out. The first problem is that alarms are mixed with events in general. Alarms MUST correspond to an undesirable state that needs corrective action. Many implementations of alarm interfaces do not adhere to this principle and just send events in general. In order to qualify as an alarm, there must exist a corrective action. If that is not true, it is an event that can go into logs.

The other misconception is that the term alarm refers to the notification itself. Rather, an alarm is a state of a resource in the system. The alarm notifications report state changes of the alarm, such as alarm raise and alarm clear.

Based upon the above, we will use the following alarm definition:

> An alarm signifies an undesirable state in a resource that requires corrective action.

"One of the most important principles of alarm management is that an alarm requires an action. This means that if the operator does not need to respond to an alarm (because unacceptable consequences do not occur), then it is not an alarm. Following this cardinal rule will help eliminate many potential alarm management issues." [ISA182]

5.2. What is an Alarm Type?

One of the fundamental requirements stated in the previous section is that every alarm must have a corresponding corrective action. This means that every vendor should be able to prepare a list of available alarms and their corrective actions. We use the term 'alarm type' to refer to every possible alarm that could be active in the system.

Alarm types are also fundamental in order to provide a state-based alarm list. The alarm list correlates alarm state changes for the same alarm type and the same resource into one alarm.

Different alarm interfaces use different mechanisms to define alarm types, ranging from simple error numbers to more advanced mechanisms like the X.733 triplet of event type, probable cause and specific problem.

This document defines an alarm type with an alarm type id and an alarm type qualifier.

The alarm type id is modeled as a YANG identity. With YANG identities, new alarm types can be defined in a distributed fashion. YANG identities are hierarchical, which means that an hierarchy of alarm types can be defined.

The primary goal for the alarm module has been to provide a simple but extensible mechanism. YANG identities is a good mechanism for enumerated values that are easy to extend.

This means that every possible alarm type that can appear in a system exists as a well defined hierarchical identity along with a description. Tools can provide a list of possible alarms by parsing the YANG identities rather than reading user guides.

Standards and vendors should define their own alarm type identities based on this definition.

The use of YANG identities means that all possible alarms are identified at design time. This explicit declaration of alarm types makes it easier to allow for alarm qualification reviews and preparation of alarm actions and documentation.

There are occasions where the alarm types are not known at design time. For example, a system with digital inputs that allows users to connects detectors (e.g., smoke detector) to the inputs. In this case it is a configuration action that says that certain connectors are fire alarms for example. The drawback of this is that there is a big risk that alarm operators will receive alarm types as a surprise, they do not know how to resolve the problem since a defined alarm procedure does not necessarily exist.

In order to allow for dynamic addition of alarm types the alarm module also allows for further qualification of the identity based alarm type using a string.

A common misunderstanding is that individual alarm notifications are alarm types. This is not correct; e.g., "link-up" and "link-down" are two notifications reporting different states for the same alarm type, "link-alarm".

A vendor or standard can then define their own alarm-type hierarchy. The example below shows a hierarchy based on X.733 event types:

```
  import ietf-alarms {
    prefix al;
  }
  identity vendor-alarms {
    base al:alarm-type;
  }
  identity communications-alarm {
    base vendor-alarms;
  }
  identity link-alarm {
    base communications-alarm;
  }
```

Alarm types can be abstract. An abstract alarm type is used as a base for defining hierarchical alarm types. Concrete alarm types are used for alarm states and appear in the alarm inventory. There are two kinds of concrete alarm types:

1. The last subordinate identity in the 'alarm-type-id' hierarchy is concrete, for example: "alarm-identity.environmental-alarm.smoke". In this example "alarm-identity" and "environmental-alarm" are abstract YANG identities, whereas "smoke" is a concrete YANG identity.

2. The YANG identity hierarchy is abstract and the concrete alarm type is defined by the dynamic alarm qualifier string, for example: "alarm-identity.environmental-alarm.external-detector" with alarm-type-qualifier "smoke".

For example:

```
// Alternative 1: concrete alarm type identity
import ietf-alarms {
  prefix al;
}
identity environmental-alarm {
  base al:alarm-type;
  description "Abstract alarm type";
}
identity smoke {
  base environmental-alarm;
  description "Concrete alarm type";
}

// Alternative 2: concrete alarm type qualifier
import ietf-alarms {
  prefix al;
}
identity environmental-alarm {
  base al:alarm-type;
  description "Abstract alarm type";
}
identity external-detector {
  base environmental-alarm;
  description
    "Abstract alarm type, a run-time configuration
     procedure sets the type of alarm detected. This will
     be reported in the alarm-qualifier.";
}
```

5.3. How are Resources Identified?

It is of vital importance to be able to refer to the alarming resource. This reference must be as fine-grained as possible. If the alarming resource exists in the data tree then an instance-identifier MUST be used with the full path to the object.

This module also allows for alternate naming of the alarming resource if it is not available in the data tree.

5.4. How are Alarm Instances Identified?

A primary goal of this alarm module is to remove any ambiguity in how alarm notifications are mapped to an update of an alarm instance. X.733 and especially 3GPP was not really clear on this point. This YANG alarm module states that the tuple (resource, alarm type identifier, alarm type qualifier) corresponds to the same alarm instance. This means that alarm notifications for the same resource

and same alarm type are matched to update the same alarm instance. These three leafs are therefore used as the key in the alarm list:

```
list alarm {
  key "resource alarm-type-id alarm-type-qualifier";
  ...
}
```

5.5. What is the Life-Cycle of an Alarm?

The alarm model clearly separates the resource alarm life-cycle from the operator and administrative life-cycles of an alarm.

- resource alarm life-cycle: the alarm instrumentation that controls alarm raise, clearance, and severity changes.

- operator alarm life-cycle: operators acting upon alarms with actions like acknowledgment and closing. Closing an alarm implies that the operator considers the corrective action performed. Operators can also shelf alarms in order to avoid nuisance alarms.

- administrative alarm life-cycle: deleting (purging) alarms and compressing the alarm status change list. This module exposes operations to manage the administrative life-cycle. The server may also perform these operations based on other policies, but how that is done is out of scope for this document.

5.5.1. Resource Alarm Life-Cycle

From a resource perspective, an alarm can have the following life-cycle: raise, change severity, change severity, clear, being raised again etc. All of these status changes can have different alarm texts generated by the instrumentation. Two important things to note:

1. Alarms are not deleted when they are cleared. Deleting alarms is an administrative process. The alarm module defines an rpc "purge" that deletes alarms.

2. Alarms are not cleared by operators, only the underlying instrumentation can clear an alarm. Operators can close alarms.

The YANG tree representation below illustrates the resource oriented life-cycle:

```
 +--ro alarm* [resource alarm-type-id alarm-type-qualifier]
       ...
    +--ro is-cleared                boolean
    +--ro last-status-change        yang:date-and-time
    +--ro last-perceived-severity   severity
    +--ro last-alarm-text           alarm-text
    +--ro status-change* [time]
       +--ro time            yang:date-and-time
       +--ro perceived-severity    severity
       +--ro alarm-text            alarm-text
```

For every status change from the resource perspective a row is added to the 'status-change' list. The last status values are also represented at leafs for the alarm. Note well that the alarm severity does not include 'cleared', alarm clearance is a flag.

An alarm can therefore look like this: ((GigabitEthernet0/25, link-alarm,""), false, T, major, "Interface GigabitEthernet0/25 down")

5.5.2. Operator Alarm Life-cycle

Operators can also act upon alarms using the set-operator-state action:

```
 +--ro alarm* [resource alarm-type-id alarm-type-qualifier]
       ...
  +--ro last-operator-state    operator-state {operator-actions}?
  +--ro last-operator?         string {operator-actions}?
  +--ro last-operator-text?    alarm-text {operator-actions}?
  +--ro last-operator-action?  yang:date-and-time {operator-actions}?
  +--ro operator-action* [time] {operator-actions}?
     +--ro time     yang:date-and-time
     +--ro state    operator-state
     +--ro name     string
     +--ro text?    string
     |     +--ro last-operator-state-change?   yang:date-and-time
     |     +--ro last-operator?                string
     |             {operator-actions}?
     |     +--ro last-operator-state           operator-state
     |             {operator-actions}?
     |     +--ro last-operator-text?           alarm-text
     |             {operator-actions}?
     |     +--ro operator-state-change* [time] {operator-actions}?
     |     |  +--ro time        yang:date-and-time
     |     |  +--ro operator    string
     |     |  +--ro state       operator-state
     |     |  +--ro text?       string
```

The operator state for an alarm can be: 'none', 'ack', 'shelved', and 'closed'. Alarm deletion, 'rpc purge', can use this state as a criteria. A closed alarm is an alarm where the operator has performed any required corrective actions. Closed alarms are good candidates for being deleted.

5.5.3. Administrative Alarm Life-Cycle

Deleting alarms from the alarm list is considered an administrative action. This is supported by the "purge-alarms" rpc. The "purge-alarms" rpc takes a filter as input. The filter selects alarms based on the operator and resource life-cycle such as "all closed cleared alarms older than a time specification". The server may also perform these operations based on other policies, but how that is done is out of scope for this document.

Alarms can be compressed. Compressing an alarm deletes all entries in the alarm's "status-change" list except for the last status change. A client can perform this using the "compress-alarms" rpc. The server may also perform these operations based on other policies, but how that is done is out of scope for this document.

5.6. Alarm Shelving

Alarm shelving is an important function in order for alarm management applications and operators to stop superfluous alarms. A shelved alarm implies that any alarms fulfilling this criteria are ignored. Shelved alarms appear in a dedicated shelved alarm list in order not to disturb the relevant alarms. Shelved alarms do not generate notifications.

6. Alarm Data Model

Alarm shelving and operator actions are YANG features so that a server can select not to support these.

The data model has the following overall structure:

```
 +--rw alarms
    +--rw control
    |  +--rw max-alarm-status-changes?   union
    |  +--rw notify-status-changes?      boolean
    |  +--rw alarm-shelving {alarm-shelving}?
    |     +--rw shelf* [shelf-name]
    |        +--rw shelf-name              string
    |        +--rw resource?               resource
    |        +--rw alarm-type-id?          alarm-type-id
    |        +--rw alarm-type-qualifier?   alarm-type-qualifier
```

```
       |        +--rw description?            string
       +--ro alarm-inventory
       |  +--ro alarm-type*
       |     +--ro alarm-type-id           alarm-type-id
       |     +--ro alarm-type-qualifier?   alarm-type-qualifier
       |     +--ro has-clear               union
       |     +--ro description             string
       +--ro summary
       |  +--ro alarm-summary* [severity]
       |  |  +--ro severity                  severity
       |  |  +--ro total?                    yang:gauge32
       |  |  +--ro cleared?                  yang:gauge32
       |  |  +--ro cleared-not-closed?       yang:gauge32
       |  |  |       {operator-actions}?
       |  |  +--ro cleared-closed?           yang:gauge32
       |  |  |       {operator-actions}?
       |  |  +--ro not-cleared-closed?       yang:gauge32
       |  |  |       {operator-actions}?
       |  |  +--ro not-cleared-not-closed?   yang:gauge32
       |  |          {operator-actions}?
       |  +--ro shelves-active?   empty {alarm-shelving}?
       +--ro alarm-list
       |  +--ro number-of-alarms?   yang:gauge32
       |  +--ro last-changed?       yang:date-and-time
       |  +--ro alarm* [resource alarm-type-id alarm-type-qualifier]
       |     +--ro resource                     resource
       |     +--ro alarm-type-id                alarm-type-id
       |     +--ro alarm-type-qualifier         alarm-type-qualifier
       |     +--ro alt-resource*                resource
       |     +--ro related-alarm*
       |     |       [resource alarm-type-id alarm-type-qualifier]
       |     |  +--ro resource
       |     |  |       -> /alarms/alarm-list/alarm/resource
       |     |  +--ro alarm-type-id           leafref
       |     |  +--ro alarm-type-qualifier    leafref
       |     +--ro impacted-resources*           resource
       |     +--ro root-cause-resources*         resource
       |     +--ro is-cleared                    boolean
       |     +--ro last-status-change            yang:date-and-time
       |     +--ro last-perceived-severity       severity
       |     +--ro last-alarm-text               alarm-text
       |     +--ro status-change* [time]
       |     |  +--ro time                   yang:date-and-time
       |     |  +--ro perceived-severity     severity-with-clear
       |     |  +--ro alarm-text             alarm-text
       |     +--ro last-operator-state-change?   yang:date-and-time
       |     |       {operator-actions}?
       |     +--ro last-operator?                string
```

```
|     |       {operator-actions}?
|     +--ro last-operator-state           operator-state
|     |       {operator-actions}?
|     +--ro last-operator-text?           alarm-text
|     |       {operator-actions}?
|     +--ro operator-state-change* [time] {operator-actions}?
|     |  +--ro time        yang:date-and-time
|     |  +--ro operator    string
|     |  +--ro state       operator-state
|     |  +--ro text?       string
|     +---x set-operator-state {operator-actions}?
|        +---w input
|           +---w state    operator-state
|           +---w text?    string
+--ro shelved-alarms {alarm-shelving}?
   +--ro number-of-shelved-alarms?   yang:gauge32
   +--ro alarm-shelf-last-changed?   yang:date-and-time
   +--ro shelved-alarm*
           [resource alarm-type-id alarm-type-qualifier]
      +--ro resource                      resource
      +--ro alarm-type-id                 alarm-type-id
      +--ro alarm-type-qualifier          alarm-type-qualifier
      +--ro alt-resource*                 resource
      +--ro related-alarm*
      |       [resource alarm-type-id alarm-type-qualifier]
      |  +--ro resource
      |  |       -> /alarms/alarm-list/alarm/resource
      |  +--ro alarm-type-id           leafref
      |  +--ro alarm-type-qualifier    leafref
      +--ro impacted-resources*           resource
      +--ro root-cause-resources*         resource
      +--ro is-cleared                    boolean
      +--ro last-status-change            yang:date-and-time
      +--ro last-perceived-severity       severity
      +--ro last-alarm-text               alarm-text
      +--ro status-change* [time]
      |  +--ro time                  yang:date-and-time
      |  +--ro perceived-severity    severity-with-clear
      |  +--ro alarm-text            alarm-text
      +--ro last-operator-state-change?   yang:date-and-time
      |       {operator-actions}?
      +--ro last-operator?                string
      |       {operator-actions}?
      +--ro last-operator-state           operator-state
      |       {operator-actions}?
      +--ro last-operator-text?           alarm-text
      |       {operator-actions}?
      +--ro operator-state-change* [time] {operator-actions}?
```

```
              +--ro time        yang:date-and-time
              +--ro operator    string
              +--ro state       operator-state
              +--ro text?       string
```

6.1. Alarm Control

The "/alarms/control/notify-status-changes" leaf controls if notifications are sent for all state changes, severity change and alarm text change, or just for new and cleared alarms.

Every alarm has a list of status changes, this is a circular list. The length of this list is controlled by "/alarms/control/max-alarm-status-changes".

6.1.1. Alarm Shelving

The shelving control tree is shown below:

```
+--rw alarm-shelving {alarm-shelving}?
      +--rw shelf* [shelf-name]
            +--rw shelf-name              string
            +--rw resource?               resource
            +--rw alarm-type-id?          alarm-type-id
            +--rw alarm-type-qualifier?   alarm-type-qualifier
```

Shelved alarms are shown in a dedicated shelved alarm list. The instrumentation MUST move shelved alarms from the alarm list (/alarms/alarm-list) to the shelved alarm list (/alarms/shelved-alarms/). Shelved alarms do not generate any notifications. When the shelving criteria is removed or changed the alarm list MUST be updated to the correct actual state of the alarms.

A leaf (/alarms/summary/shelfs-active) in the alarm summary indicates if there are shelved alarms.

A system can select to not support the shelving feature.

6.2. Alarm Inventory

The alarm inventory represents all possible alarm types that may occur in the system. A management system may use this to build alarm procedures. The alarm inventory is relevant for several reasons:

The system might not instrument all alarm type identities.

The system has configured dynamic alarm types using the alarm qualifier. The inventory makes it possible for the management system to discover these.

Note that the mechanism whereby dynamic alarm types are added using the alarm type qualifier MUST populate this list.

The alarm inventory tree is shown below:

```
 +--ro alarm-inventory
       +--ro alarm-type*
             +--ro alarm-type-id           alarm-type-id
             +--ro alarm-type-qualifier?   alarm-type-qualifier
             +--ro has-clear               union
             +--ro description             string
```

6.3. Alarm Summary

The alarm summary list summarises alarms per severity; how many cleared, cleared and closed, and closed. It also gives an indication if there are shelved alarms.

6.4. The Alarm List

The alarm list (/alarms/alarm-list) is a function from (resource, alarm type, alarm type qualifier) to the current alarm state.

```
  +--ro alarm-list
     +--ro number-of-alarms?   yang:gauge32
     +--ro last-changed?       yang:date-and-time
     +--ro alarm* [resource alarm-type-id alarm-type-qualifier]
        +--ro resource                      resource
        +--ro alarm-type-id                 alarm-type-id
        +--ro alarm-type-qualifier          alarm-type-qualifier
        +--ro alt-resource*                 resource
        +--ro related-alarm*
        |       [resource alarm-type-id alarm-type-qualifier]
        |  +--ro resource
        |  |       -> /alarms/alarm-list/alarm/resource
        |  +--ro alarm-type-id           leafref
        |  +--ro alarm-type-qualifier    leafref
        +--ro impacted-resources*           resource
        +--ro root-cause-resources*         resource
        +--ro is-cleared                    boolean
        +--ro last-status-change            yang:date-and-time
        +--ro last-perceived-severity       severity
        +--ro last-alarm-text               alarm-text
        +--ro status-change* [time]
        |  +--ro time                  yang:date-and-time
        |  +--ro perceived-severity    severity-with-clear
        |  +--ro alarm-text            alarm-text
        +--ro last-operator-state-change?   yang:date-and-time
        |       {operator-actions}?
        +--ro last-operator?                string
        |       {operator-actions}?
        +--ro last-operator-state           operator-state
        |       {operator-actions}?
        +--ro last-operator-text?           alarm-text
        |       {operator-actions}?
        +--ro operator-state-change* [time] {operator-actions}?
        |  +--ro time        yang:date-and-time
        |  +--ro operator    string
        |  +--ro state       operator-state
        |  +--ro text?       string
        +---x set-operator-state {operator-actions}?
           +---w input
              +---w state    operator-state
              +---w text?    string
```

Every alarm has three important states, the resource clearance state "is-cleared", the operator state "last-operator-state" and the severity "last-perceived-severity".

In order to see the alarm history the resource state changes are available in the "status-change" list and the operator history is available in the "operator-state-change" list.

6.5. The Shelved Alarms List

The shelved alarm list has the same structure as the alarm list above. It shows all the alarms that matches the shelving criteria (/alarms/control/alarm-shelving).

6.6. RPCs

The alarm module supports rpcs/actions to manage the alarms:

"purge-alarms" (RPC): delete alarms according to specific criteria, for example all cleared alarms older then a specific date.

"compress" and "compress-alarms" rpcs: compress the status-change list for the alarms.

"set-operator-state" action: change the operator state for an alarm: for example acknowledge.

6.7. Notifications

The alarm module supports a general notification to report alarm state changes. It carries all relevant parameters for the alarm management application.

There is also a notification to report that an operator changed the operator state on an alarm, like acknowledge.

7. Alarm YANG Module

```
<CODE BEGINS> file "ietf-alarms.yang"
module ietf-alarms {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-alarms";
  prefix al;

  import ietf-yang-types {
    prefix yang;
  }

  organization
```

```
       "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        Editor:   Stefan Vallin
                  <mailto:stefan@wallan.se>

        Editor:   Martin Bjorklund
                  <mailto:mbj@tail-f.com>";

     description
       "This module is an interface for managing alarms.  Main inputs to
        the module design are the 3GPP Alarm IRP, ITU-T X.733 and
        ANSI/ISA-18.2 alarm standards.

        Main features of this module include:

          * Alarm list:
                    A list of all alarms.  Cleared alarms stay in
                    the list until explicitly removed.

          * Operator actions on alarms:
                    Acknowledging and closing alarms.

          * Administrative actions on alarms:
                    Purging alarms from the list according to specific
                    criteria.

          * Alarm inventory:
                    A management application can read all
                    alarm types implemented by the system.

          * Alarm shelving:
                    Shelving (blocking) alarms according
                    to specific criteria.

        This module uses a stateful view on alarms.  An alarm is a state
        for a specific resource (note that an alarm is not a
        notification).  An alarm type is a possible alarm state for a
        resource.  For example, the tuple:

          ('link-alarm', 'GigabitEthernet0/25')

        is an alarm of type 'link-alarm' on the resource
        'GigabitEthernet0/25'.
```

```
        Alarm types are identified using YANG identities and an optional
        string-based qualifier.  The string-based qualifier allows for
        dynamic extension of the statically defined alarm types.  Alarm
        types identify a possible alarm state and not the individual
        notifications.  For example, the traditional 'link-down' and
        'link-up' notifications are two notifications referring to the
        same alarm type 'link-alarm'.

        With this design there is no ambiguity about how alarm and alarm
        clear correlation should be performed: notifications that report
        the same resource and alarm type are considered updates of the
        same alarm, such as clearing an active alarm or changing the
        severity of an alarm.

        The instrumentation can update 'severity' and 'alarm-text' on an
        existing alarm.  The above alarm example can therefore look
        like:

          (('link-alarm', 'GigabitEthernet0/25'),
           warning,
           'interface down while interface admin state is up')

        There is a clear separation between updates on the alarm from
        the underlying resource, like clear, and updates from an
        operator like acknowledge or closing an alarm:

          (('link-alarm', 'GigabitEthernet0/25'),
           warning,
           'interface down while interface admin state is up',
           cleared,
           closed)

        Administrative actions like removing closed alarms older than a
        given time is supported.";

      revision 2016-10-05 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: YANG Alarm Module";
      }

      /*
       * Features
       */

      feature operator-actions {
        description
```

```
        "This feature means that the systems supports operator states
         on alarms.";
    }

    feature alarm-shelving {
      description
        "This feature means that the system supports shelving
         (blocking) alarms.";
    }

    /*
     * Identities
     */

    identity alarm-identity {
      description
        "Base identity for alarm types.  A unique identification of the
         alarm, not including the resource.  Different resources can
         share alarm types.  If the resource reports the same alarm
         type, it is to be considered to be the same alarm.  The alarm
         type is a simplification of the different X.733 and 3GPP alarm
         IRP alarm correlation mechanisms and it allows for
         hierarchical extensions.

         A string-based qualifier can be used in addition to the
         identity in order to have different alarm types based on
         information not known at design-time, such as values in
         textual SNMP Notification var-binds.

         Standards and vendors can define sub-identities to clearly
         identify specific alarm types.

         This identity is abstract and shall not be used for alarms.";
    }

    /*
     * Common types
     */

    typedef resource {
      type union {
        type instance-identifier {
          require-instance false;
        }
        type yang:object-identifier;
        type string;
      }
      description
```

```
        "This is an identification of the alarming resource, such as an
         interface.  It should be as fine-grained as possible both to
         guide the operator and to guarantee uniqueness of the
         alarms.  If a resource has both a config and a state tree
         normally this should identify the state tree,
         (e.g., /interfaces-state/interface/name).
         But if the instrumentation can detect a broken config, this
         should be identified as the resource.
         If the alarming resource is modelled in YANG, this
         type will be an instance-identifier.  If the resource is an
         SNMP object, the type will be an object-identifier.  If the
         resource is anything else, for example a distinguished name or
         a CIM path, this type will be a string.";
    }

    typedef alarm-text {
      type string;
      description
        "The string used to inform operators about the alarm.  This
         MUST contain enough information for an operator to be able
         to understand the problem and how to resolve it.  If this
         string contains structure, this format should be clearly
         documented for programs to be able to parse that
         information.";
    }

    typedef severity  {
      type enumeration {
        enum indeterminate {
          value 2;
          description
            "Indicates that the severity level could not be
             determined. This level SHOULD be avoided.";
        }
        enum minor {
          value 3;
          description
            "The 'minor' severity level indicates the existence of a
             non-service affecting fault condition and that corrective
             action should be taken in order to prevent a more serious
             (for example, service affecting) fault. Such a severity
             can be reported, for example, when the detected alarm
             condition is not currently degrading the capacity of the
             resource.";
        }
        enum warning {
          value 4;
          description
```

```
            "The 'warning' severity level indicates the detection of
             a potential or impending service affecting fault, before
             any significant effects have been felt. Action should be
             taken to further diagnose (if necessary) and correct the
             problem in order to prevent it from becoming a more
             serious service affecting fault.";
        }
        enum major {
          value 5;
          description
            "The 'major' severity level indicates that a service
             affecting condition has developed and an urgent
             corrective action is required. Such a severity can be
             reported, for example, when there is a severe
             degradation in the capability of the resource
             and its full capability must be restored.";
        }
        enum critical {
          value 6;
          description
            "The 'critical' severity level indicates that a service
             affecting condition has occurred and an immediate
             corrective action is required. Such a severity can be
             reported, for example, when a resource becomes totally
             out of service and its capability must be restored.";
        }
      }
      description
        "The severity level of the alarm.  Note well that value 'clear'
         is not included. If an alarm is cleared or not is a separate
         boolean flag.";
      reference
        "ITU Recommendation X.733: Information Technology
           - Open Systems Interconnection
           - System Management: Alarm Reporting Function";
    }

    typedef severity-with-clear  {
      type union {
        type enumeration {
          enum cleared {
            value 1;
            description
              "The alarm is cleared by the instrumentation.";
          }
        }
        type severity;
      }
```

```
      description
        "The severity level of the alarm including clear.
         This is used *only* in notifications reporting state changes
         for an alarm.";
    }

    typedef operator-state  {
      type enumeration {
        enum none {
          value 1;
          description
            "The alarm is not being taken care of.";
        }
        enum ack {
          value 2;
          description
            "The alarm is being taken care of.  Corrective action not
             taken yet, or failed";
        }
        enum closed {
          value 3;
          description
            "Corrective action taken successfully.";
        }
        enum shelved {
          value 4;
          description
            "Alarm shelved. Alarms in alarms/shelved-alarms/
             MUST be assigned this state by the server.";
        }
        enum un-shelved {
          value 5;
          description
            "Alarm moved back to alarm-list from shelf.
             Alarms 'moved' from /alarms/shelved-alarms/
             to /alarms/alarm-list MUST be assigned this
             state by the server.";
        }

      }
      description
        "Operator states on an alarm.  The 'closed' state indicates
         that an operator considers the alarm being resolved.  This
         is separate from the resource alarm clear flag.";
    }

    /* Alarm type */
```

```
    typedef alarm-type-id {
      type identityref {
        base alarm-identity;
      }
      description
        "Identifies an alarm type. The description of the alarm type id
         MUST indicate if the alarm type is abstract or not. An
         abstract alarm type is used as a base for other alarm type ids
         and will not be used as a value for an alarm or be present in
         the alarm inventory.";
    }

    typedef alarm-type-qualifier {
      type string;
      description
        "If an alarm type can not be fully specified at design time by
         alarm-type-id, this string qualifier is used in addition to
         fully define a unique alarm type.

         The definition of alarm qualifiers is considered being part
         of the instrumentation and out of scope for this module.
         An empty string is used when this is part of a key.";
    }

    /*
     * Groupings
     */

    grouping common-alarm-parameters {
      description
        "Common parameters for an alarm.

         This grouping is used both in the alarm list and in the
         notification representing an alarm state change.";

      leaf resource {
        type resource;
        mandatory true;
        description
          "The alarming resource.  See also 'alt-resource'.
           This could for example be a reference to the alarming
           interface";
      }

      leaf alarm-type-id {
        type alarm-type-id;
        mandatory true;
        description
```

```
          "This leaf and the leaf 'alarm-type-qualifier' together
           provides a unique identification of the alarm type.";
        }

        leaf alarm-type-qualifier {
          type alarm-type-qualifier;
          description
            "This leaf is used when the 'alarm-type-id' leaf cannot
             uniquely identify the alarm type.  Normally, this is not
             the case, and this leaf is the empty string.";
        }

        leaf-list alt-resource {
          type resource;
          description
            "Used if the alarming resource is available over other
             interfaces.  This field can contain SNMP OID's, CIM paths or
             3GPP Distinguished names for example.";
        }

        list related-alarm {
          key "resource alarm-type-id alarm-type-qualifier";

          description
            "References to related alarms.  Note that the related alarm
             might have been removed from the alarm list.";

          leaf resource {
            type leafref {
              path "/alarms/alarm-list/alarm/resource";
              require-instance false;
            }
            description
              "The alarming resource for the related alarm.";
          }
          leaf alarm-type-id {
            type leafref {
              path "/alarms/alarm-list/alarm"
                 + "[resource=current()/../resource]"
                 + "/alarm-type-id";
              require-instance false;
            }
            description
              "The alarm type identifier for the related alarm.";
          }
          leaf alarm-type-qualifier {
            type leafref {
              path "/alarms/alarm-list/alarm"
```

```
              + "[resource=current()/../resource]"
              + "[alarm-type-id=current()/../alarm-type-id]"
              + "/alarm-type-qualifier";
            require-instance false;
          }
          description
            "The alarm qualifier for the related alarm.";
        }
      }
      leaf-list impacted-resources {
        type resource;
        description
          "Resources that might be affected by this alarm.";
      }
      leaf-list root-cause-resources {
        type resource;
        description
          "Resources that are candidates for causing the alarm.";
      }
    }

    grouping alarm-state-change-parameters {
      description
        "Parameters for an alarm state change.

         This grouping is used both in the alarm list's
         status-change list and in the notification representing an
         alarm state change.";

      leaf time {
        type yang:date-and-time;
        mandatory true;
        description
          "The time the status of the alarm changed.  The value
           represents the time the real alarm state change appeared
           in the resource and not when it was added to the
           alarm list.";
      }
      leaf perceived-severity {
        type severity-with-clear;
        mandatory true;
        description
          "The severity of the alarm as defined by X.733.  Note
           that this may not be the original severity since the alarm
           may have changed severity.";
        reference
          "ITU Recommendation X.733: Information Technology
             - Open Systems Interconnection
```

```
            - System Management: Alarm Reporting Function";
      }
      leaf alarm-text {
        type alarm-text;
        mandatory true;
        description
          "A user friendly text describing the alarm state change.";
        reference
          "ITU Recommendation X.733: Information Technology
             - Open Systems Interconnection
             - System Management: Alarm Reporting Function";
      }
    }

    grouping operator-parameters {
      description
        "This grouping defines parameters that can
         be changed by an operator";
      leaf time {
        type yang:date-and-time;
        mandatory true;
        description
          "Timestamp for operator action on alarm.";
      }
      leaf operator {
        type string;
        mandatory true;
        description
          "The name of the operator that has acted on this
           alarm.";
      }
      leaf state {
        type operator-state;
        mandatory true;
        description
          "The operator's view of the alarm state.";
      }
      leaf text {
        type string;
        description
          "Additional optional textual information provided by
           the operator.";
      }
    }

    grouping resource-alarm-parameters {
      description
        "Alarm parameters that originates from the resource view.";
```

```
      leaf is-cleared {
        type boolean;
        mandatory true;
        description
          "Indicates the clearance state of the alarm.  An alarm
           might toggle from active alarm to cleared alarm and back
           to active again.  This leaf reflects the perceived
           severity in the latest entry in the status-change
           list.";
      }

      leaf last-status-change {
        type yang:date-and-time;
        mandatory true;
        description
          "A timestamp when the status-change list was last
           changed. This value equals the latest 'when' leaf in the
           status-change list.  The value can be used by a manager
           to read the last status change without iterating the
           status-change list below. Alarms can be deleted by the
           operator using the 'purge' rpc, the status change  list of
           an alarm can be compressed by the 'compress-alarms' rpc.
           The server may also purge and compress alarms based on
           locally configured policies, but how this is done is out
           of scope for this document.";
      }

      leaf last-perceived-severity {
        type severity;
        mandatory true;
        description
          "The severity of the last status-change that
           reported a severity that is not equal to cleared.";
      }

      leaf last-alarm-text {
        type alarm-text;
        mandatory true;
        description
          "The alarm-text of the last status-change that
           reported a severity that is not equal to cleared.";
      }

      list status-change {
        key time;
        min-elements 1;
        description
          "A list of status change events for this alarm.
```

```
            This list is ordered according to the timestamps of
            alarm state changes. The last item corresponds to the
            latest state change.

            The following state changes creates an entry in this
            list:
            - changed severity (warning, minor, major, critical)
            - clearance status, this also updates the is-cleared
              leaf
            - alarm text update";

        uses alarm-state-change-parameters;
      }
    }

    grouping operator-alarm-parameters {
      description
        "Alarm parameters that originates from the operator view.";

      leaf last-operator-state-change {
        if-feature operator-actions;
        type yang:date-and-time;
        description
          "A timestamp when the operator-state-change list was last
           changed.";
      }
      leaf last-operator {
        if-feature operator-actions;
        type string;
        description
          "The last operator that acted upon the alarm.";
      }

      leaf last-operator-state {
        if-feature operator-actions;
        type operator-state;
        mandatory true;
        description
          "The state of the alarm as set by the operator. When the
           alarm is first raised by the instrumentation it has the
           'none' state. After initial alarm raise this leaf
           represents the state in the latest entry in the
           'operator-state-change' list.
           The 'closed' state indicates that the alarm is
           considered resolved by the operator.";
      }

      leaf last-operator-text {
```

```
        if-feature operator-actions;
        type alarm-text;
        description
          "The alarm-text of the last status change.";
      }


      list operator-state-change {
        if-feature operator-actions;
        key time;
        description
          "This list is used by operators to indicate
           the state of human intervention on an alarm.
           For example, if an operator has seen an alarm,
           the operator can add a new item to this list indicating
           that the alarm is acknowledged.";
        uses operator-parameters;
      }

    }

    /*
     * The /alarms data tree
     */

    container alarms {
      description
        "The top container for this module";
      container control {
        description
          "Configuration to control the alarm behaviour.";
        leaf max-alarm-status-changes {
          type union {
            type uint16;
            type enumeration {
              enum infinite {
                description
                  "The status change entries are accumulated
                   infinitely.";
              }
            }
          }
          default 32;
          description
            "The status-change entries are kept in a circular list
             per alarm.  When this number is exceeded, the oldest
             status change entry is automatically removed.  If the
             value is 'infinite', the status change entries are
```

```
          accumulated infinitely.";
      }

      leaf notify-status-changes {
        type boolean;
        default false;
        description
          "This leaf controls whether notifications are sent on all
           alarm status updates, e.g., updated perceived-severity or
           alarm-text.  By default the notifications are only sent
           when a new alarm is raised, re-raised after being cleared
           and when an alarm is cleared.";
      }
      container alarm-shelving {
        if-feature alarm-shelving;
        description
          "This list is used to shelf alarms.  The server will move
           any alarms corresponding to the shelving criteria from the
           alarms/alarm-list/alarm list to the
           alarms/shelved-alarms/shelved-alarm list.  It will also
           stop sending notifications for the shelved alarms.  The
           conditions in the shelf criteria are logically ANDed. When
           the shelving criteria is deleted or changed, the
           non-matching alarms MUST appear in the
           alarms/alarm-list/alarm list according to the real state.
           This means that the instrumentation MUST maintain states
           for the shelved alarms.  Alarms that match the criteria
           shall have an operator-state 'shelved'.";
        list shelf {
          key shelf-name;
          leaf shelf-name {
            type string;
            description
              "An arbitrary name for the alarm shelf.";
          }
          description
            "Each entry defines the criteria for shelving alarms.
             Criterias are ANDed.";

          leaf resource {
            type resource;
            description
              "Shelv alarms for this resource.";
          }
          leaf alarm-type-id {
            type alarm-type-id;
            description
              "Shelv alarms for this alarm type identifier.";
```

```
            }
            leaf alarm-type-qualifier {
              type alarm-type-qualifier;
              description
                "Shelv alarms for this alarm type qualifier.";
            }
            leaf description {
              type string;
              description
                "An optional textual description of the shelf.  This
                 description should include the reason for shelving
                 these alarms.";
            }
          }
        }
      }

      container alarm-inventory {
        config false;
        description
          "This list contains all possible alarm types for the system.
           The list also tells if each alarm type has a corresponding
           clear state. The inventory shall only contain concrete alarm
           types.";
        list alarm-type {
          description
            "An entry in this list defines a possible alarm.";
          leaf alarm-type-id {
            type alarm-type-id;
            mandatory true;
            description
              "The statically defined alarm type identifier for this
               possible alarm.";
          }
          leaf alarm-type-qualifier {
            type alarm-type-qualifier;
            description
              "The optionally dynamically defined alarm type identifier
               for this possible alarm.";
          }
          leaf has-clear {
            type union {
              type boolean;
            }
            mandatory true;
            description
              "This leaf tells the operator if the alarm will be
               cleared when the correct corrective action has been
```

```
            taken.  Implementations SHOULD strive for detecting the
            cleared state for all alarm types.  If this leaf is
            true, the operator can monitor the alarm until it
            becomes cleared after the corrective action has been
            taken.  If this leaf is false the operator needs to
            validate that the alarm is not longer active using other
            mechanisms.  Alarms can lack a corresponding clear due
            to missing instrumentation or that there is no logical
            corresponding clear state.";
        }
        leaf description {
          type string;
          mandatory true;
          description
            "A description of the possible alarm.  It SHOULD include
             information on possible underlying root causes and
             corrective actions.";
        }
      }
    }

    container summary {
      config false;
      description
        "This container gives a summary of number of alarms
         and shelved alarms";
      list alarm-summary {
        key severity;
        description
          "A global summary of all alarms in the system.";
        leaf severity {
          type severity;
          description
            "Alarm summary for this severity level.";
        }
        leaf total {
          type yang:gauge32;
          description
            "Total number of alarms of this severity level.";
        }
        leaf cleared {
          type yang:gauge32;
          description
            "For this severity level, the number of alarms that are
          cleared.";
        }
        leaf cleared-not-closed {
          if-feature operator-actions;
```

```
            type yang:gauge32;
            description
              "For this severity level, the number of alarms that are
               cleared but not closed.";
          }
          leaf cleared-closed {
            if-feature operator-actions;
            type yang:gauge32;
            description
              "For this severity level, the number of alarms that are
               cleared and closed.";
          }
          leaf not-cleared-closed {
            if-feature operator-actions;
            type yang:gauge32;
            description
              "For this severity level, the number of alarms that are
               not cleared but closed.";
          }
          leaf not-cleared-not-closed {
            if-feature operator-actions;
            type yang:gauge32;
            description
              "For this severity level, the number of alarms that are
               not cleared and not closed.";
          }
        }
        leaf shelves-active {
          if-feature alarm-shelving;
          type empty;
          description
            "This is a hint to the operator that there are active
             alarm shelves. This leaf MUST exist if the
             alarms/shelved-alarms/number-of-shelved-alarms is > 0.";
        }
      }

      container alarm-list {
        config false;
        description
          "The alarms in the system.";
        leaf number-of-alarms {
          type yang:gauge32;
          description
            "This object shows the total number of
             alarms in the system, i.e., the total number
             of entries in the alarm list.";
        }
```

```
        leaf last-changed {
          type yang:date-and-time;
          description
            "A timestamp when the alarm list was last
             changed. The value can be used by a manager to
             initiate an alarm resynchronization procedure.";
        }

        list alarm {
          key "resource alarm-type-id alarm-type-qualifier";

          description
            "The list of alarms.  Each entry in the list holds one
             alarm for a given alarm type and resource.
             An alarm can be updated from the underlying resource or
             by the user. These changes are reflected in different
             lists below the corresponding alarm.";

          uses common-alarm-parameters;
          uses resource-alarm-parameters;
          uses operator-alarm-parameters;

          action set-operator-state {
            if-feature operator-actions;
            description
              "This is a means for the operator to indicate
               the level of human intervention on an alarm.";
            input {
              leaf state {
                type operator-state;
                mandatory true;
                description
                  "Set this operator state.";
              }
              leaf text {
                type string;
                description
                  "Additional optional textual information.";
              }
            }
          }

        }
      }

      container shelved-alarms {
        if-feature alarm-shelving;
        config false;
```

```
        description
          "The shelved alarms. Alarms appear here if they match the
           criterias in /alarms/control/alarm-shelving. This list does
           not generate any notifications. The list represents alarms
           that are considered not relevant by the operator. Alarms in
           this list have an operator-state of 'shelved'. This can not
           be changed.";
        leaf number-of-shelved-alarms {
          type yang:gauge32;
          description
            "This object shows the total number of currently
             alarms, i.e., the total number of entries
             in the alarm list.";
        }

        leaf alarm-shelf-last-changed {
          type yang:date-and-time;
          description
            "A timestamp when the shelved alarm list was last
             changed. The value can be used by a manager to
             initiate an alarm resynchronization procedure.";
        }

        list shelved-alarm {
          key "resource alarm-type-id alarm-type-qualifier";

          description
            "The list of shelved alarms.  Each entry in the list holds
             one alarm for a given alarm type and resource.  An alarm
             can be updated from the underlying resource or by the
             user. These changes are reflected in different lists below
             the corresponding alarm.";

          uses common-alarm-parameters;
          uses resource-alarm-parameters;
          uses operator-alarm-parameters;

        }
      }
    }

    /*
     * Operations
     */

    rpc compress-alarms {
      description
        "This operation requests the server to compress entries in the
```

```
          alarm list by removing all but the latest state change for all
          alarms.  Conditions in the input are logically ANDed.  If no
          input condition is given, all alarms are compressed.";
        input {
          leaf resource {
            type leafref {
              path "/alarms/alarm-list/alarm/resource";
              require-instance false;
            }
            description
              "Compress the alarms with this resource.";
          }
          leaf alarm-type-id {
            type leafref {
              path "/alarms/alarm-list/alarm/alarm-type-id";
            }
            description
              "Compress alarms with this alarm-type-id.";
          }
          leaf alarm-type-qualifier {
            type leafref {
              path "/alarms/alarm-list/alarm/alarm-type-qualifier";
            }
            description
              "Compress the alarms with this alarm-type-qualifier.";
          }
        }
        output {
          leaf compressed-alarms {
            type uint32;
            description
              "Number of compressed alarm entries.";
          }
        }
      }

      grouping filter-input {
        description
          "Grouping to specify a filter construct on alarm information.";
        leaf alarm-status {
          type enumeration {
            enum any {
              description
                "Ignore alarm clearance status.";
            }
            enum cleared {
              description
                "Filter cleared alarms.";
```

```
          }
          enum not-cleared {
            description
              "Filter not cleared alarms.";
          }
        }
        mandatory true;
        description
          "The clearance status of the alarm.";
      }

      container older-than {
        presence "Age specification";
        description
          "Matches the 'last-status-change' leaf in the alarm.";
        choice age-spec {
          description
            "Filter using date and time age.";
          case seconds {
            leaf seconds {
              type uint16;
              description
                "Seconds part";
            }
          }
          case minutes {
            leaf minutes {
              type uint16;
              description
                "Minute part";
            }
          }
          case hours {
            leaf hours {
              type uint16;
              description
                "Hours part.";
            }
          }
          case days {
            leaf days {
              type uint16;
              description
                "Day part";
            }
          }
          case weeks {
            leaf weeks {
```

```
            type uint16;
            description
              "Week part";
          }
        }
      }
    }
    container severity {
      presence "Severity filter";
      choice sev-spec {
        description
          "Filter based on severity level.";
        leaf below {
          type severity;
          description
            "Severity less than this leaf.";
        }
        leaf is {
          type severity;
          description
            "Severity level equal this leaf.";
        }
        leaf above {
          type severity;
          description
            "Severity level higher than this leaf.";
        }
      }
      description
        "Filter based on severity.";
    }
    container operator-state-filter {
      if-feature operator-actions;
      presence "Operator state filter";
      leaf state {
        type operator-state;
        description
          "Filter on operator state.";
      }
      leaf user {
        type string;
        description
          "Filter based on which operator.";
      }
      description
        "Filter based on operator state.";
    }
  }
```

```
    rpc purge-alarms {
      description
        "This operation requests the server to delete entries from the
         alarm list according to the supplied criteria.  Typically it
         can be used to delete alarms that are in closed operator state
         and older than a specified time.  The number of purged alarms
         is returned as an output parameter";
      input {
        uses filter-input;
      }
      output {
        leaf purged-alarms {
          type uint32;
          description
            "Number of purged alarms.";
        }
      }
    }

    /*
     * Notifications
     */

    notification alarm-notification {
      description
        "This notification is used to report a state change for an
         alarm.  The same notification is used for sending a newly
         raised alarm, a cleared alarm or changing the text and/or
         severity of an existing alarm.";

      uses common-alarm-parameters;
      uses alarm-state-change-parameters;
    }

    notification operator-action {
      if-feature operator-actions;
      description
        "This notification is used to report that an operator
         acted upon an alarm.";

      leaf resource {
        type leafref {
          path "/alarms/alarm-list/alarm/resource";
          require-instance false;
        }
        description
          "The alarming resource.";
      }
```

```
      leaf alarm-type-id {
        type leafref {
          path "/alarms/alarm-list/alarm"
             + "[resource=current()/../resource]"
             + "/alarm-type-id";
          require-instance false;
        }
        description
          "The alarm type identifier for the alarm.";
      }
      leaf alarm-type-qualifier {
        type leafref {
          path "/alarms/alarm-list/alarm"
             + "[resource=current()/../resource]"
             + "[alarm-type-id=current()/../alarm-type-id]"
             + "/alarm-type-qualifier";
          require-instance false;
        }
        description
          "The alarm qualifier for the alarm.";
      }

      uses operator-parameters;
    }
  }

  <CODE ENDS>
```

8. X.733 Alarm Mapping Data Model

Many alarm management systems are based on the X.733 alarm standard. This YANG module allows a mapping from alarm types to X.733 event-type and probable-cause.

The module augments the alarm inventory, the alarm list and the alarm notification with X.733 parameters.

The module also supports a feature whereby the alarm manager can configure the mapping. This might be needed when the default mapping provided by the system is in conflict with other systems or not considered good.

9. X.733 Alarm Mapping YANG Module

This YANG module references [X.736].

```
<CODE BEGINS> file "ietf-alarms-x733.yang"
module ietf-alarms-x733 {
```

```
yang-version 1.1;
namespace "urn:ietf:params:xml:ns:yang:ietf-alarms-x733";
prefix x733;

import ietf-alarms {
  prefix al;
}

organization
  "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

contact
  "WG Web:   <http://tools.ietf.org/wg/netmod/>
   WG List:  <mailto:netmod@ietf.org>

   Editor:   Stefan Vallin
             <mailto:stefan@wallan.se>

   Editor:   Martin Bjorklund
             <mailto:mbj@tail-f.com>";

description
  "This module augments the ietf-alarms module with X.733 mapping
    information.  The following structures are augemented with
    event type and probable cause:

    1) alarm inventory: all possible alarms.
    2) alarm: every alarm in the system.
    3) alarm notification: notifications indicating alarm state
       changes.

    The module also optionally allows the alarm management system
    to configure the mapping.  The mapping does not include a
    a corresponding specific problem value.  The recommendation is
    to use alarm-type-qualifier which serves the same purpose.";
reference
  "ITU Recommendation X.733: Information Technology
     - Open Systems Interconnection
     - System Management: Alarm Reporting Function";

revision 2016-10-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Alarm Module";
}

/*
```

```
      * Features
      */

     feature configure-x733-mapping {
       description
         "The system supports configurable X733 mapping from
          alarm type to event type and probable cause.";
     }

     /*
      * Typedefs
      */

     typedef event-type  {
       type enumeration {
         enum other {
           value 1;
           description
             "None of the below.";
         }
         enum communications-alarm {
           value 2;
           description
             "An alarm of this type is principally associated with the
              procedures and/or processes required to convey
              information from one point to another.";
           reference
             "ITU Recommendation X.733: Information Technology
                - Open Systems Interconnection
                - System Management: Alarm Reporting Function";
         }
         enum quality-of-service-alarm {
           value 3;
           description
             "An alarm of this type is principally associated with a
              degradation in the quality of a service.";
           reference
             "ITU Recommendation X.733: Information Technology
                - Open Systems Interconnection
                - System Management: Alarm Reporting Function";
         }
         enum processing-error-alarm {
           value 4;
           description
             "An alarm of this type is principally associated with a
              software or processing fault.";
           reference
             "ITU Recommendation X.733: Information Technology
```

```
            - Open Systems Interconnection
            - System Management: Alarm Reporting Function";
        }
        enum equipment-alarm {
          value 5;
          description
            "An alarm of this type is principally associated with an
             equipment fault.";
          reference
            "ITU Recommendation X.733: Information Technology
               - Open Systems Interconnection
               - System Management: Alarm Reporting Function";
        }
        enum environmental-alarm {
          value 6;
          description
            "An alarm of this type is principally associated with a
            condition relating to an enclosure in which the equipment
            resides.";
          reference
            "ITU Recommendation X.733: Information Technology
               - Open Systems Interconnection
               - System Management: Alarm Reporting Function";
        }
        enum integrity-violation {
          value 7;
          description
            "An indication that information may have been illegally
             modified, inserted or deleted.";
          reference
            "ITU Recommendation X.736: Information Technology
               - Open Systems Interconnection
               - System Management: Security Alarm Reporting Function";
        }
        enum operational-violation {
          value 8;
          description
            "An indication that the provision of the requested service
             was not possible due to the unavailability, malfunction or
             incorrect invocation of the service.";
          reference
            "ITU Recommendation X.736: Information Technology
               - Open Systems Interconnection
               - System Management: Security Alarm Reporting Function";
        }
        enum physical-violation {
          value 9;
          description
```

```
            "An indication that a physical resource has been violated
             in a way that suggests a security attack.";
          reference
            "ITU Recommendation X.736: Information Technology
               - Open Systems Interconnection
               - System Management: Security Alarm Reporting Function";
        }
        enum security-service-or-mechanism-violation {
          value 10;
          description
            "An indication that a security attack has been detected by
             a security service or mechanism.";
          reference
            "ITU Recommendation X.736: Information Technology
               - Open Systems Interconnection
               - System Management: Security Alarm Reporting Function";
        }
        enum time-domain-violation {
          value 11;
          description
            "An indication that an event has occurred at an unexpected
             or prohibited time.";
          reference
            "ITU Recommendation X.736: Information Technology
               - Open Systems Interconnection
               - System Management: Security Alarm Reporting Function";
        }
      }
      description
        "The event types as defined by X.733 and X.736.  The use of the
         term 'event' is a bit confusing.  In an alarm context these
         are top level alarm types.";
    }

    /*
     * Groupings
     */

    grouping x733-alarm-parameters {
      description
        "Common X.733 parameters for alarms.";

      leaf event-type {
        type event-type;
        description
          "The X.733/X.736 event type for this alarm.";
      }
      leaf probable-cause {
```

```
        type uint32;
        description
          "The X.733 probable cause for this alarm.";
      }
    }

    grouping x733-alarm-definition-parameters {
      description
        "Common X.733 parameters for alarm definitions.";

      leaf event-type {
        type event-type;
        description
          "The alarm type has this X.733/X.736 event type.";
      }
      leaf probable-cause {
        type uint32;
        description
          "The alarm type has this X.733 probable cause value.
           This module defines probable cause as an integer
           and not as an enumeration.  The reason being that the
           primary use of probable cause is in the management
           application if it is based on the X.733 standard.
           However, most management applications have their own
           defined enum definitions and merging enums from
           different systems might create conflicts.  By using
           a configurable uint32 the system can be configured
           to match the enum values in the manager.";
      }
    }

    /*
     * Add X.733 parameters to the alarm defintions, alarms,
     * and notification.
     */

    augment "/al:alarms/al:alarm-inventory/al:alarm-type" {
      description
        "Augment X.733 mapping information to the alarm inventory.";

      uses x733-alarm-definition-parameters;
    }

    augment "/al:alarms/al:control" {
      description
        "Add X.733 mapping capabilities. ";
      list x733-mapping {
        if-feature configure-x733-mapping;
```

```
        key "alarm-type-id alarm-type-qualifier-match";
        description
          "This list allows a management application to control the
           X.733 mapping for all alarm types in the system. Any entry
           in this list will allow the alarm manager to over-ride the
           default X.733 mapping in the system and the final mapping
           will be shown in the alarm-inventory";

        leaf alarm-type-id {
          type al:alarm-type-id;
          description
            "Map the alarm type with this alarm type identifier.";
        }
        leaf alarm-type-qualifier-match {
          type string;
          description
            "A W3C regular expression that is used when mapping an
             alarm type and alarm-type-qualifier to X.733 parameters.";
        }

        uses x733-alarm-definition-parameters;
      }
    }

    augment "/al:alarms/al:alarm-list/al:alarm" {
      description
        "Augment X.733 information to the alarm.";

      uses x733-alarm-parameters;
    }

    augment "/al:alarms/al:shelved-alarms/al:shelved-alarm" {
      description
        "Augment X.733 information to the alarm.";

      uses x733-alarm-parameters;
    }

    augment "/al:alarm-notification" {
      description
        "Augment X.733 information to the alarm notification.";

      uses x733-alarm-parameters;
    }
  }

  <CODE ENDS>
```

10. Security Considerations

None.

11. Acknowledgements

The author wishes to thank Viktor Leijon and Johan Nordlander for their valuable input on forming the alarm model.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

12.2. Informative References

[ALARMIRP] 3GPP, "Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)", 3GPP TS 32.111-2 3.4.0, March 2005.

[ALARMSEM] Wallin, S., Leijon, V., Nordlander, J., and N. Bystedt, "The semantics of alarm definitions: enabling systematic reasoning about alarms. International Journal of Network Management, Volume 22, Issue 3, John Wiley and Sons, Ltd, http://dx.doi.org/10.1002/nem.800", March 2012.

[EEMUA] EEMUA Publication No. 191 Engineering Equipment and Materials Users Association, London, 2 edition., "Alarm Systems: A Guide to Design, Management and Procurement.", 2007.

[ISA182] International Society of Automation,ISA, "ANSI/ISA-18.2-2009 Management of Alarm Systems for the Process Industries", 2009.

[RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <http://www.rfc-editor.org/info/rfc3877>.

[X.733] International Telecommunications Union, "Information Technology - Open Systems Interconnection - Systems Management: Alarm Reporting Function", ITU-T Recommendation X.733, 1992.

[X.736] International Telecommunications Union, "Information Technology - Open Systems Interconnection - Systems Management: Security alarm reporting function", ITU-T Recommendation X.736, 1992.

Appendix A. Enterprise-specific Alarm-Types Example

This example shows how to define alarm-types in an enterprise specific module. In this case "xyz" has chosen to define top level identities according to X.733 event types.

```
module example-xyz-alarms {
  namespace "urn:example:xyz-alarms";
  prefix xyz-al;

  import ietf-alarms {
    prefix al;
  }

  identity xyz-alarms {
    base al:alarm-identity;
  }

  identity communications-alarm {
    base xyz-alarms;
  }
  identity quality-of-service-alarm {
    base xyz-alarms;
  }
  identity processing-error-alarm {
    base xyz-alarms;
  }
  identity equipment-alarm {
    base xyz-alarms;
  }
  identity environmental-alarm {
    base xyz-alarms;
  }

  // communications alarms
  identity link-alarm {
    base communications-alarm;
  }

  // QoS alarms
  identity high-jitter-alarm {
    base quality-of-service-alarm;
  }
}
```

Appendix B. Alarm Inventory Example

This shows an alarm inventory, it shows one alarm type defined only with the identifier, and another dynamically configured. In the latter case a digital input has been connected to a smoke-detector, therefore the 'alarm-type-qualifier' is set to "smoke-detector" and the 'alarm-type-identity' to "environmental-alarm".

```
<alarms xmlns="urn:ietf:params:xml:ns:yang:ietf-alarms"
        xmlns:xyz-al="urn:example:xyz-alarms">
  <alarm-inventory>
    <alarm-type>
      <description>
        Link failure, operational state down but admin state up
      </description>
      <alarm-type-id>xyz-al:link-alarm</alarm-type-id>
      <has-clear>true</has-clear>
    </alarm-type>
    <alarm-type>
      <description>
        Connected smoke detector to digital input
      </description>
      <alarm-type-id>xyz-al:environmental-alarm</alarm-type-id>
      <alarm-type-qualifier>smoke-alarm</alarm-type-qualifier>
      <has-clear>true</has-clear>
    </alarm-type>
  </alarm-inventory>
</alarms>
```

Appendix C. Alarm List Example

In this example we show an alarm that has toggled [major, clear, major]. An operator has acknowledged the alarm.

```
<alarms xmlns="urn:ietf:params:xml:ns:yang:ietf-alarms"
        xmlns:xyz-al="urn:example:xyz-alarms"
        xmlns:dev="urn:example:device">
  <alarm-list>
    <number-of-alarms>1</number-of-alarms>
    <last-changed>2015-04-08T08:39:50.00Z</last-changed>

    <alarm>
      <resource>
        /dev:interfaces/dev:interface[name='FastEthernet1/0']
      </resource>
      <alarm-type-id>xyz-al:link-alarm</alarm-type-id>
      <alarm-type-qualifier></alarm-type-qualifier>

      <is-cleared>false</is-cleared>
      <alt-resource>1.3.6.1.2.1.2.2.1.1.17</alt-resource>
      <last-status-change>
        2015-04-08T08:39:40.00Z
      </last-status-change>
      <last-perceived-severity>major</last-perceived-severity>
      <last-alarm-text>
```

```
          Link operationally down but administratively up
        </last-alarm-text>
        <status-change>
          <time>2015-04-08T08:39:40.00Z</time>
          <perceived-severity>major</perceived-severity>
          <alarm-text>
            Link operationally down but administratively up
          </alarm-text>
        </status-change>
        <status-change>
          <time>2015-04-08T08:30:00.00+00:00</time>
          <perceived-severity>cleared</perceived-severity>
          <alarm-text>
            Link operationally up and administratively up
          </alarm-text>
        </status-change>
        <status-change>
          <time>2015-04-08T08:20:10.00+00:00</time>
          <perceived-severity>major</perceived-severity>
          <alarm-text>
            Link operationally down but administratively up
          </alarm-text>
        </status-change>

        <last-operator-state>ack</last-operator-state>
        <last-operator-text>
          Will investigate, ticket TR764999
        </last-operator-text>
        <last-operator-state-change>
          2015-04-08T08:39:50.00Z
        </last-operator-state-change>
        <operator-state-change>
          <time>2015-04-08T08:39:50.00Z</time>
          <state>ack</state>
          <operator>joe</operator>
          <text>Will investigate, ticket TR764999</text>
        </operator-state-change>
      </alarm>

    </alarm-list>
  </alarms>
```

Appendix D. Alarm Shelving Example

This example shows how to shelf alarms. We shelf alarms related to the smoke-detectors since they are being installed and tested. We also shelf all alarms from FastEthernet1/0.

```
<alarms xmlns="urn:ietf:params:xml:ns:yang:ietf-alarms"
        xmlns:xyz-al="urn:example:xyz-alarms"
        xmlns:dev="urn:example:device">
  <control>
    <alarm-shelving>
      <shelf>
        <shelf-name>FE10</shelf-name>
        <resource>
          /dev:interfaces/dev:interface[name='FastEthernet1/0']
        </resource>
      </shelf>
      <shelf>
        <shelf-name>detectortest</shelf-name>
        <alarm-type-id>xyz-al:environmental-alarm</alarm-type-id>
        <alarm-type-qualifier>smoke-alarm</alarm-type-qualifier>
      </shelf>
    </alarm-shelving>
  </control>
</alarms>
```

Appendix E. X.733 Mapping Example

This example shows how to map a dynamic alarm type (alarm-type-identity=environmental-alarm, alarm-type-qualifier=smoke-alarm) to the corresponding X.733 event-type and probable cause parameters.

```
<alarms xmlns="urn:ietf:params:xml:ns:yang:ietf-alarms"
        xmlns:xyz-al="urn:example:xyz-alarms">
  <control>
    <x733-mapping
       xmlns="urn:ietf:params:xml:ns:yang:ietf-alarms-x733">
      <alarm-type-id>xyz-al:environmental-alarm</alarm-type-id>
      <alarm-type-qualifier-match>
        smoke-alarm
      </alarm-type-qualifier-match>
      <event-type>quality-of-service-alarm</event-type>
      <probable-cause>777</probable-cause>
    </x733-mapping>
  </control>
</alarms>
```

Authors' Addresses

Stefan Vallin
Stefan Vallin AB

Email: stefan@wallan.se

Martin Bjorklund
Cisco

Email: mbj@tail-f.com