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 operations 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 May 26, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
   1.1. Terminology and Notation .................................. 3
2. Objectives ....................................................... 4
3. Alarm Module Concepts ........................................ 5
   3.1. Alarm Definition ........................................... 5
   3.2. Alarm Type ................................................ 5
   3.3. Identifying the Alarming Resource ......................... 7
   3.4. Identifying Alarm Instances ............................... 8
   3.5. Alarm Life-Cycle .......................................... 8
      3.5.1. Resource Alarm Life-Cycle .............................. 9
      3.5.2. Operator Alarm Life-cycle ............................ 10
      3.5.3. Administrative Alarm Life-Cycle ........................ 10
   3.6. Root Cause, Impacted Resources and Related Alarms ....... 10
   3.7. Alarm Shelving ........................................... 11
   3.8. Alarm Profiles ........................................... 12
4. Alarm Data Model .............................................. 12
   4.1. Alarm Control ............................................. 14
      4.1.1. Alarm Shelving ....................................... 14
   4.2. Alarm Inventory ........................................... 14
   4.3. Alarm Summary ............................................. 15
   4.4. The Alarm List ............................................ 16
   4.5. The Shelved Alarms List .................................. 18
   4.6. Alarm Profiles ............................................. 18
   4.7. Operations ................................................ 19
   4.8. Notifications ............................................. 19
5. Relationship to the ietf-hardware YANG module ............... 19
6. Alarm YANG Module ............................................. 20
7. X.733 Extensions .............................................. 50
8. The X.733 Mapping Module .................................... 51
9. IANA Considerations ........................................... 62
10. Security Considerations ....................................... 63
11. Acknowledgements ............................................. 64
12. References ................................................... 64
   12.1. Normative References .................................... 64
   12.2. Informative References ................................... 65
Appendix A. Vendor-specific Alarm-Types Example ............... 66
Appendix B. Alarm Inventory Example ............................. 67
Appendix C. Alarm List Example .................................. 68
Appendix D. Alarm Shelving Example .............................. 69
Appendix E. X.733 Mapping Example .............................. 70
Appendix F. Relationship to other alarm standards ............. 71
   F.1. Alarm definition ........................................ 71
   F.2. Data model .............................................. 73
1. Introduction

This document defines a YANG [RFC7950] module for alarm management. The purpose is to define a standardized 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 alarm management application in a standardized format is a starting point for performing higher level network assurance tasks.

The design of the module is based on experience from using and implementing available alarm standards from ITU [X.733], 3GPP [ALARMIRP] and ANSI [ISA182].

1.1. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC7950]:

- action
- client
- data tree
- 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.
2. 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.

Tree diagrams used in this document follow the notation defined in [RFC8340].
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, see Appendix G. 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 some 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.

3. Alarm Module Concepts

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

3.1. Alarm Definition

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

There are two main things to remember from this definition:

1. the definition focuses on leaving out events and logging information in general. Alarms should only be used for undesired states that require action.

2. the definition also focus on alarms as a state on a resource, not the notifications that report the state changes.

See Appendix F for information how this definition relates to other alarm standards.

3.2. Alarm Type

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.
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 connect 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.

In order to allow for dynamic addition of alarm types the alarm module allows for further qualification of the identity based alarm type using a string. A potential 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. To avoid this risk the system MUST publish all possible alarm types in the alarm inventory, see Section 4.2.

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

```yaml
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-type-qualifier.";
}
```

A server SHOULD strive to minimize the number of dynamically defined alarm types.

3.3. Identifying the Alarming Resource

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.
When the module is used in a controller/orchestrator/manager the original device resource identification can be modified to include the device in the path. The details depend on how devices are identified, and are out of scope for this specification.

Example:

The original device alarm might identify the resource as 
"/dev:interfaces/dev:interface[dev:name='FastEthernet1/0']".

The resource identification in the manager could look something like: "/mgr:devices/mgr:device[mgr:name='xyz123']/dev:interfaces/
dev:interface[dev:name='FastEthernet1/0']"

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

3.4. Identifying Alarm Instances

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 were not really clear on this point. This YANG alarm module states that the tuple (resource, alarm type identifier, alarm type qualifier) corresponds to a single 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 leaves are therefore used as the key in the alarm list:

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

3.5. Alarm Life-Cycle

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 (block/filter) alarms in order to avoid nuisance alarms.
A server SHOULD describe how long it retains cleared/closed alarms: until manually purged or if it has an automatic removal policy.

### 3.5.1. Resource Alarm Life-Cycle

From a resource perspective, an alarm can for example 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 action "purge-alarms" 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:

```yang
++-ro alarm* [resource alarm-type-id alarm-type-qualifier]
    
    +-ro is-cleared boolean
    +-ro last-changed yang:date-and-time
    +-ro perceived-severity severity
    +-ro alarm-text alarm-text
    +-ro status-change* [time] {alarm-history}?
        
        +-ro time yang:date-and-time
        +-ro perceived-severity severity-with-clear
        +-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 as leafs for the alarm. Note well that the alarm severity does not include "cleared", alarm clearance is a boolean flag.

An alarm can therefore look like this: `((GigabitEthernet0/25, link-alarm,""), false, T, major, "Interface GigabitEthernet0/25 down")`
3.5.2. 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 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    writable-operator-state
        +----w text?    string
```

The operator state for an alarm can be: "none", "ack", "shelved", and "closed". Alarm deletion (using the action "purge-alarms"), 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 purged.

3.5.3. Administrative Alarm Life-Cycle

Deleting alarms from the alarm list is considered an administrative action. This is supported by the "purge-alarms" action. The "purge-alarms" action 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.

Purged alarms are removed from the alarm list. Note well, if the alarm resource state changes after a purge, the alarm will reappear in the alarm list.

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" action. The server may also perform these operations based on other policies, but how that is done is out of scope for this document.

3.6. Root Cause, Impacted Resources and Related Alarms

The general principle of this alarm module is to limit the amount of alarms. In many cases several resources are affected for a given underlying problem. A full disk will of course impact databases and
applications as well. The recommendation is not have a single alarm for the underlying problem an list the affected resources in the alarm, rather than having separate alarms for each resource.

The alarm has one leaf-list to identify possible "impacted-resources" and a leaf-list to identify possible "root-cause-resources". These serves as hints only. It is up to the client application to use this information to present the overall status. Using the the disk full example, a "good" alarm would be to use the hard disk partition as the alarming resource and add the database and applications into the impacted-resources leaf-list.

A system should always strive to identify the resource that can be acted upon as the "resource" leaf. The "impacted-resource" leaf-list shall be used to identify any side-effects of the alarm. The impacted resources can not be acted upon to fix the problem. The disk full example above illustrates the principle; you can not fix the underlying issue by database operations. However, you need to pay attention to the database to perform any operations that limits the impact of problem.

In some occasions the system might not be capable of detecting the root cause, the resource that can be acted upon. The instrumentation in this case only monitors the side-effect and needs to represent an alarm that indicates a situation that needs acting upon. The instrumentation still might identify possible candidates for the root-cause resource. In this case the "root-cause-resource" leaf-list can be used to indicate the candidate root-cause resources. An example of this kind of alarm might be an active test tool that detects an SLA violation on a VPN connection and identifies the devices along the chain as candidate root causes.

The alarm module also supports a way to associate different alarms to each other with the "related-alarm" list. This list enables the server to inform the client that certain alarms are related to other alarms.

Note well that this module does not prescribe any dependencies or preference between the above alarm correlation mechanisms. Different systems have different capabilities and the above described mechanisms are available to support the instrumentation features.

### 3.7. 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 (blocked/filtered). Shelved alarms appear in a dedicated shelved
alarm list in order not to disturb the relevant alarms. Shelved alarms do not generate notifications.

3.8. Alarm Profiles

Alarm profiles are used to configure further information to an alarm type. This module supports configuring severity levels overriding the system default levels. This corresponds to the Alarm Assignment Profile, ASAP, functionality in M.3100 [M.3100] and M.3160 [M.3160]. Other standard or enterprise modules can augment this list with further alarm type information.

4. Alarm Data Model

The fundamental parts of the data model are the "alarm-list" with associated notifications and the "alarm-inventory" list of all possible alarm types. These MUST be implemented by a system. The rest of the data model are made conditional with YANG the features "operator-actions", "alarm-shelving", "alarm-history", "alarm-summary", "alarm-profile", and "severity-assignment".

The data model has the following overall structure:
---rw control
|   ---rw max-alarm-status-changes? union
|   |   ---rw (notify-status-changes)? ...
|   |   ---rw alarm-shelving {alarm-shelving}? ...
---ro alarm-inventory
|   ---ro alarm-type* [alarm-type-id alarm-type-qualifier]
|   ...
---ro summary {alarm-summary}?
|   ---ro alarm-summary* [severity]
|   |   ...
|   ---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]
|   |   ...
|   |   ---x purge-alarms ...
|   |   ---x compress-alarms {alarm-history}?
|   ...
---ro shelved-alarms {alarm-shelving}?
|   ---ro number-of-shelved-alarms? yang:gauge32
|   ---ro shelved-alarms-last-changed? yang:date-and-time
|   ---ro shelved-alarm*
|   |   [resource alarm-type-id alarm-type-qualifier]
|   |   ...
|   |   ---x purge-shelved-alarms ...
|   |   ---x compress-shelved-alarms {alarm-history}?
|   ...
---rw alarm-profile*
|   [alarm-type-id alarm-type-qualifier-match resource]
|   {alarm-profile}?
|   ---rw alarm-type-id alarm-type-id
|   ---rw alarm-type-qualifier-match string
|   ---rw resource resource-match
|   ---rw description string
|   ---rw alarm-severity-assignment-profile
|       {severity-assignment}?
4.1. Alarm Control

The "/alarms/control/notify-status-changes" choice controls if notifications are sent for all state changes, only raise and clear, or only notifications more severe than a configured level. This feature in combination with alarm shelving corresponds to the ITU Alarm Report Control functionality.

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".

4.1.1. Alarm Shelving

The shelving control tree is shown below:

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

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.

Shelving and unshelving can only be performed by editing the shelf configuration. It cannot be performed on individual alarms. The server will add an operator state indicating that the alarm was shelved/unshelved.

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.

4.2. 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 defined alarm type identities, and some alarm identities are abstract.

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 optional leaf-list "resource" in the alarm inventory enables the system to publish for which resources a given alarm type may appear.

A server MUST implement the alarm inventory in order to enable controlled alarm procedures in the client.

A server implementer may want to document the alarm inventory for off-line processing by clients. The file format defined in [I-D.ietf-netmod-yang-instance-file-format] can be used for this purpose.

The alarm inventory tree is shown below:

```
+--ro alarm-inventory
    +--ro alarm-type* [alarm-type-id alarm-type-qualifier]
        +--ro alarm-type-id           alarm-type-id
        +--ro alarm-type-qualifier    alarm-type-qualifier
        +--ro resource*               resource-match
        +--ro has-clear               boolean
        +--ro severity-levels*        severity
        +--ro description             string
```

4.3. Alarm Summary

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

The alarm summary tree is shown below:
---ro summary {alarm-summary}?
  +--ro alarm-summary* [severity]
    |   +--ro severity                  severity
    |   +--ro total?                    yang:gauge32
    |   +--ro not-cleared?              yang:gauge32
    |   +--ro cleared?                  yang:gauge32
    |   |       {operator-actions}?
    |   |       +--ro cleared-closed?     yang:gauge32
    |   |       |       {operator-actions}?
    |   |       +--ro cleared-not-closed?  yang:gauge32
    |   |       |       {operator-actions}?
    |   +--ro not-cleared-closed?       yang:gauge32
    |   |       {operator-actions}?
    +--ro shelves-active?   empty {alarm-shelving}?

4.4. The Alarm List

The alarm list (/alarms/alarm-list) is a function from (resource, alarm type, alarm type qualifier) to the current composite alarm state. The composite state includes states for the resource life-cycle such as severity, clearance flag and operator states such as acknowledgment. This means that for a given resource and alarm-type the alarm list shows the current states of the alarm such as acknowledged and cleared status.

---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-resource*         resource
    +--ro root-cause-resource*       resource
    +--ro time-created               yang:date-and-time
    +--ro is-cleared                 boolean
    +--ro last-raised                yang:date-and-time
    +--ro last-changed               yang:date-and-time
    +--ro perceived-severity         severity
```yang
++--ro alarm-text
++--ro status-change* [time] {alarm-history}?
    +--ro time      yang:date-and-time
    +--ro perceived-severity severity-with-clear
    +--ro alarm-text alarm-text
++--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  writable-operator-state
        +---w text? string
++--n operator-action {operator-actions}?
    +-- time      yang:date-and-time
    +-- operator  string
    +-- state     operator-state
    +-- text?     string
++--x purge-alarms
    +---w input
        +---w alarm-status  enumeration
        +---w older-than!
        |    +---w (age-spec)?
        |        +--:(seconds)
        |            |    +---w seconds?  uint16
        |        +--:(minutes)
        |            |    +---w minutes?  uint16
        |        +--:(hours)
        |            |    +---w hours?  uint16
        |        +--:(days)
        |            |    +---w days?  uint16
        |        +--:(weeks)
        |            |    +---w weeks?  uint16
        +---w severity!
        |    +---w (sev-spec)?
        |        +--:(below)
        |            |    +---w below?  severity
        |        +--:(is)
        |            |    +---w is?  severity
        |        +--:(above)
        |            |    +---w above?  severity
        +---w operator-state-filter! {operator-actions}?
    +---w state?  operator-state
    +---w user?   string
++--ro output
    +--ro purged-alarms?  uint32
++--x compress-alarms {alarm-history}?
```
Every alarm has three important states, the resource clearance state "is-cleared", the severity "perceived-severity" and the operator state available in the operator state change list.

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.

4.5. 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).

4.6. Alarm Profiles

Alarm profiles (/alarms/alarm-profile/) is a list of configurable alarm types. The list supports configurable alarm severity levels in the container "alarm-severity-assignment-profile". If an alarm matches the configured alarm type it MUST use the configured severity level(s) instead of the system default. This configuration MUST also be represented in the alarm inventory.
4.7. Operations

The alarm module supports the following actions to manage the alarms:

/alarms/alarm-list/purge-alarms: Delete alarms from the "alarm-list" according to specific criteria, for example all cleared alarms older than a specific date.

/alarms/alarm-list/compress-alarms: Compress the "status-change" list for the alarms.

/alarms/alarm-list/alarm/set-operator-state: Change the operator state for an alarm. For example, an alarm can be acknowledged by setting the operator state to "ack".

/alarms/shelved-alarm-list/purge-shelved-alarms: Delete alarms from the "shelved-alarm-list" according to specific criteria, for example all alarms older than a specific date.

/alarms/shelved-alarm-list/compress-shelved-alarms: Compress the "status-change" list for the alarms.

4.8. 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.

If the alarm inventory is changed, for example a new card type is inserted, a notification will tell the management application that new alarm types are available.

5. Relationship to the ietf-hardware YANG module

RFC 8348 [RFC8348] defines the "ietf-hardware" YANG data model for the management of hardware. The "alarm-state" in RFC 8348 is a summary of the alarm severity levels that may be active on the specific hardware component. It does not say anything about how alarms are reported, and it doesn’t provide any details of the alarms.

The mapping between the alarm YANG data model and the "alarm-state" in RFC 8348 is as follows:
6. Alarm YANG Module

This YANG module references [RFC6991].

<CODE BEGINS> file "ietf-alarms@2018-11-22.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;
        reference "RFC 6991: Common YANG Data Types.";
    }

    organization "IETF CCAMP Working Group";
    contact
        "WG Web:  <http://tools.ietf.org/wg/ccamp>
        WG List:  <mailto:ccamp@ietf.org>
        Editor:  Stefan Vallin
            <mailto:stefan@wallan.se>
        Editor:  Martin Bjorklund
            <mailto:mbj@tail-f.com>
        // RFC Ed.: replace XXXX with actual RFC number and
        // remove this note.

    description "This module defines 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 purged.

* 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.

* Alarm profiles:
  A management system can attach further information to alarm types, for example overriding system default severity levels.

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, e.g., 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:

```yang
(('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:

```yang
(('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.

This alarm module does not define how the underlying instrumentation detects and clears the specific alarms. That belongs to the SDO or enterprise that owns that specific technology.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info).

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in the module text are to be interpreted as described in BCP 14 [RFC 2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This version of this YANG module is part of RFC XXXX (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for full legal notices.";

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
revision 2018-11-22 {
description
"Initial revision.";
reference "RFC XXXX: YANG Alarm Module";
}

*/
* Features
*/

feature operator-actions {
    description
    "This feature indicates that the system supports operator
    states on alarms.";
}

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

feature alarm-history {
    description
    "This feature indicates that server maintains a history of
    state changes for each alarm. For example, if an alarm
    toggles between cleared and active 10 times, these state
    changes are present in a separate list in the alarm.";
}

feature alarm-summary {
    description
    "This feature indicates that the server summarizes the number
    of alarms per severity and operator state.";
}

feature alarm-profile {
    description
    "The system supports clients to configure further information
    to each alarm type.";
}

feature severity-assignment {
    description
    "The system supports configurable alarm severity levels.";
    reference
    "M.3160/M.3100 Alarm Severity Assignment Profile, ASAP";
}
identity alarm-type-id {
  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 MUST NOT be used for alarms."
};

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

If the alarming object is identified by a UUID use the uuid type. Be cautious when using this type, since a UUID is hard to use for an operator.

If the server supports several models, the precedence should be in the order as given in the union definition.

```yaml
typedef resource-match {
  type union {
    type yang:xpath1.0;
    type yang:object-identifier;
    type string;
  }
  description
  "This type is used to match resources of type ‘resource’. Since the type ‘resource’ is a union of different types, the ‘resource-match’ type is also a union of corresponding types.

  If the type is given as an XPath 1.0 expression, a resource of type ‘instance-identifier’ matches if the instance is part of the node set that is the result of evaluating the XPath 1.0 expression. For example, the XPath 1.0 expression:

  `/ietf-interfaces:interfaces/ietf-interfaces:interface
    [ietf-interfaces:type='ianaift:ethernetCsmacd']`

  would match the resource instance-identifier:

  `/if:interfaces/if:interface[if:name='eth1'],

  assuming that the interface ‘eth1’ is of type ‘ianaift:ethernetCsmacd’.

  If the type is given as an object identifier, a resource of type ‘object-identifier’ matches if the match object identifier is a prefix of the resource’s object identifier. For example, the value:

  1.3.6.1.2.1.2.2

  would match the resource object identifier:

  1.3.6.1.2.1.2.2.1.1.5
```
If the type is given as an UUID or a string, it is interpreted as a W3C regular expression, which matches a resource of type 'yang:uuid' or 'string' if the given regular expression matches the resource string.

If the type is given as an XPath expression it is evaluated in the following XPath context:

- The set of namespace declarations is the set of prefix and namespace pairs for all YANG modules implemented by the server, where the prefix is the YANG module name and the namespace is as defined by the 'namespace' statement in the YANG module.

If a leaf of this type is encoded in XML, all namespace declarations in scope on the leaf element are added to the set of namespace declarations. If a prefix found in the XML is already present in the set of namespace declarations, the namespace in the XML is used.

- The set of variable bindings is empty.

- The function library is the core function library and the functions defined in Section 10 of RFC 7950.

- The context node is the root node in the data tree.

```yml
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
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 writable-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.";
  }
}
description
"Operator states on an alarm.  The ‘closed’ state indicates
that an operator considers the alarm being resolved.  This is
separate from the alarm’s ‘is-cleared’ leaf.";
}

typedef operator-state {
type union {
  type writable-operator-state;
type enumeration {
    enum shelved {
      value 4;
      description
      "The alarm is not being taken care of. Corrective action not
taken yet, or failed";
    }
  }
  type severity;
}
description
"The severity level of the alarm including clear.  This is used
only in notifications reporting state changes for an alarm.";
}
"The alarm is shelved. Alarms in /alarms/shelved-alarms/ MUST be assigned this operator state by the server as the last entry in the operator-state-change list. The text for that entry SHOULD include the shelf name.";
}

enum un-shelved {
  value 5;
  description
  "The alarm is moved back to 'alarm-list' from a shelf. Alarms that are moved from /alarms/shelved-alarms/ to /alarms/alarm-list MUST be assigned this state by the server as the last entry in the 'operator-state-change' list. The text for that entry SHOULD include the shelf name.";
}
}

description
"Operator states on an alarm. The 'closed' state indicates that an operator considers the alarm being resolved. This is separate from the alarm’s 'is-cleared' leaf.";

/* Alarm type */

typedef alarm-type-id {
  type identityref {
    base alarm-type-id;
  }
  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.";
}
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 purged 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-resource {
    type resource;
    description
    "Resources that might be affected by this alarm. If the system creates an alarm on a resource and also has a mapping to other resources that might be impacted, these resources can be listed in this leaf-list. In this way the system can create one alarm instead of several. For example, if an interface has an alarm, the ‘impacted-resource’ can reference the aggregated port channels."
}
leaf-list root-cause-resource {
    type resource;
    description
    "Resources that are candidates for causing the alarm. If the system has a mechanism to understand the candidate root causes of an alarm, this leaf-list can be used to list the root cause candidate resources. In this way the system can create one alarm instead of several. An example might be a logging system (alarm resource) that fails, the alarm can reference the file-system in the ‘root-cause-resource’ leaf-list. Note that the intended use is not to also send an an alarm with the root-cause-resource as alarming
resource. The root-cause-resource leaf list is a hint and should not also generate an alarm for the same problem.
}

} 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. The /alarm-list/alarm/last-changed MUST be set to the same value."

  }

  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

  }

  leaf alarm-text {
    type alarm-text;
    mandatory true;
    description
    "A user friendly text describing the alarm state change."

    reference

  }
}

} 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 current clearance state of the alarm. An alarm might toggle from active alarm to cleared alarm and back to active again.";
  }
  leaf last-raised {
    type yang:date-and-time;
    mandatory true;
    description
      "An alarm may change severity level and toggle between active and cleared during its life-time. This leaf indicates the last time it was last raised (is-cleared = false).";
  }
  leaf last-changed {

type yang:date-and-time;
mandatory true;
description "A timestamp when the alarm status was last changed. Status changes are changes to 'is-cleared', 'perceived-severity', and 'alarm-text'.";
}
leaf perceived-severity {
    type severity;
    mandatory true;
    description "The last severity of the alarm. If an alarm was raised with severity 'warning', but later changed to 'major', this leaf will show 'major'.";
}
leaf alarm-text {
    type alarm-text;
    mandatory true;
    description "The last reported alarm text. This text should contain information for an operator to be able to understand the problem and how to resolve it.";
}
list status-change {
    if-feature "alarm-history";
    key "time";
    min-elements 1;
    description "A list of status change events for this alarm.

    The entry with latest time-stamp in this list MUST correspond to the leafs 'is-cleared', 'perceived-severity' and 'alarm-text' for the alarm. The time-stamp for that entry MUST be equal to the 'last-changed' leaf.

    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 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 "Hour part";
        }
      }
    }
  }
}

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 {
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.";
    }
    choice notify-status-changes {
        description "This leaf controls the notifications sent for alarm status updates. There are three options:

1. Notifications are sent for all updates, severity level changes and alarm text changes

2. Notifications are only sent for alarm raise and clear

3. Notifications are sent for status changes equal to or above the specified severity level. Clear notifications shall always be sent. Notifications shall also be sent for state changes that makes an alarm less severe than the specified level.

For example, in option 3, assuming the severity level is set to major and that the alarm has the following state changes:

\[(\text{Time, severity, clear})\]:
\[(T1, \text{major}, -), (T2, \text{minor}, -), (T3, \text{warning, -}), (T4, \text{minor, -}), (T5, \text{major, -}), (T6, \text{critical, -}), (T7, \text{major. -}), (T8, \text{major, clear})\]

In that case, notifications will be sent at times T1, T2, T5, T6, T7 and T8.

leaf notify-all-state-changes {
  type empty;
  description
  "Send notifications for all status changes.";
}
leaf notify-raise-and-clear {
  type empty;
  description
  "Send notifications only for raise, clear, and re-raise. Notifications for severity level changes or alarm text changes are not sent.";
}
leaf notify-severity-level {
  type severity;
  description
  "Only send notifications for alarm state changes crossing the specified level. Always send clear notifications.";
}
} container alarm-shelving {
  if-feature "alarm-shelving";
  description
  "The alarm-shelving/shelf list is used to shelve (block/filter) alarms. The server will move any alarms corresponding to the shelving criteria from the alarms/alarms-list/alarms 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'. When the shelf configuration removes an alarm from the shelf the server shall add an operator state 'unshelved'.

```yaml
list shelf {
  key "name";
  leaf name {
    type string;
    description
    "An arbitrary name for the alarm shelf.";
  }
  description
  "Each entry defines the criteria for shelving alarms. Criteria are ANDed. If no criteria are specified, all alarms will be shelved.";
  leaf-list resource {
    type resource-match;
    description
    "Shelve alarms for matching resources.";
  }
  leaf alarm-type-id {
    type alarm-type-id;
    description
    "Shelve all alarms that have an alarm-type-id that is equal to or derived from the given alarm-type-id.";
  }
  leaf alarm-type-qualifier-match {
    type string;
    description
    "A W3C regular expression that is used to match an alarm type qualifier. Shelve all alarms that matches this regular expression for the 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 alarm-inventory/alarm-type list contains all possible alarm types for the system.

    If the system knows for which resources a specific alarm type can appear, this is also identified in the inventory.
    The list also tells if each alarm type has a corresponding clear state. The inventory shall only contain concrete alarm types.

    The alarm inventory MUST be updated by the system when new alarms can appear. This can be the case when installing new software modules or inserting new card types. A notification ‘alarm-inventory-changed’ is sent when the inventory is changed."
}

list alarm-type {
    key "alarm-type-id alarm-type-qualifier";
    description "An entry in this list defines a possible alarm.";
    leaf alarm-type-id {
        type alarm-type-id;
        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-list resource {
        type resource-match;
        description "Optionally, specifies for which resources the alarm type is valid.";
    }
    leaf has-clear {
        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-list severity-levels {
  type severity;
  description
  "This leaf-list indicates the possible severity levels of this alarm type. Note well that ‘clear’ is not part of the severity type. In general, the severity level should be defined by the instrumentation based on dynamic state and not defined statically by the alarm type in order to provide relevant severity level based on dynamic state and context. However most alarm types have a defined set of possible severity levels and this should be provided here."
}

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 {
  if-feature "alarm-summary";
  config false;
  description
  "This container gives a summary of number of alarms."
  list alarm-summary {
    key "severity";
    description
    "A global summary of all alarms in the system. The summary does not include shelved alarms."
    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 not-cleared {
  type yang:gauge32;
  description
    "Total number of alarms of this severity level that are not cleared.";
}
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.
             The following leafs are maintained by the resource:
             is-cleared, last-change, perceived-severity, and
             alarm-text. An operator can change: operator-state and
             operator-text.

             Entries appear in the alarm list the first time an alarm
             becomes active for a given alarm-type and resource.
             Entries do not get deleted when the alarm is cleared, this
             is a boolean state in the alarm.

             Alarm entries are removed, purged, from the list by an
             explicit purge action. For example, purge all alarms that
             are cleared and in closed operator-state that are older
than 24 hours. Purged alarms are removed from the alarm list. If the alarm resource state changes after a purge, the alarm will reappear in the alarm list.

Systems may also remove alarms based on locally configured policies which is out of scope for this module.

```yang
uses common-alarm-parameters;
leaf time-created {
  type yang:date-and-time;
  mandatory true;
  description
    "The time-stamp when this alarm entry was created. This
    represents the first time the alarm appeared, it can
    also represent that the alarm re-appeared after a purge.
    Further state-changes of the same alarm does not change
    this leaf, these changes will update the ‘last-changed’
    leaf."
}
uses resource-alarm-parameters;
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;
}
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 writable-operator-state;
      mandatory true;
      description
        "Set this operator state."
    }
    leaf text {
      type string;
      description
        "Additional optional textual information."
    }
  }
}
notification operator-action {
  if-feature "operator-actions";
  description
    "This notification is used to report that an operator
     acted upon an alarm."
  uses operator-parameters;
}

action purge-alarms {
  description
    "This operation requests the server to delete entries from
     the alarm list according to the supplied criteria.

     Typically this operation is 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.";
    }
  }
}

action compress-alarms {
  if-feature "alarm-history";
  description
    "This operation requests the server to compress entries in
     the alarm list by removing all but the latest
     'status-change' entry for all matching alarms. Conditions
     in the input are logically ANDed. If no input condition
     is given, all alarms are compressed.";
  input {
    leaf resource {
      type resource-match;
      description
        "Compress the alarms matching this resource.";
    }
    leaf alarm-type-id {
      type leafref {
        path "/alarms/alarm-list/alarm/alarm-type-id";
        require-instance false;
      }
    }
  }
}
leaf alarm-type-qualifier {
  type leafref {
    path "/alarms/alarm-list/alarm/alarm-type-qualifier";
    require-instance false;
  }
  description
  "Compress the alarms with this alarm-type-qualifier.";
}
}
output {
  leaf compressed-alarms {
    type uint32;
    description
    "Number of compressed alarm entries.";
  }
}
}
}
}
}
}
}
}
}
}
}

if-feature "alarm-shelving";
config false;
description
"The shelved alarms. Alarms appear here if they match the criteria 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 shelved-alarms-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. Shelved alarms can only be updated from the underlying resource, no operator actions are supported."
uses common-alarm-parameters;
leaf shelf-name {
    type leafref {
        path "/alarms/control/alarm-shelving/shelf/name";
        require-instance false;
    }
    description
    "The name of the shelf."
}
uses resource-alarm-parameters;
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 shelved alarms, the system has set the list item in the list to 'shelved'."
    uses operator-parameters;
}
}
action purge-shelved-alarms {
    description
    "This operation requests the server to delete entries from the shelved alarms list according to the supplied criteria.

    In the shelved alarm list it makes sense to delete alarms that are not relevant anymore.

    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."
        }
    }
}
action compress-shelved-alarms {
    if-feature "alarm-history";
This operation requests the server to compress entries in the shelved alarm list by removing all but the latest 'status-change' entry for all matching shelved alarms. Conditions in the input are logically ANDed. If no input condition is given, all alarms are compressed.

```.yang
input {
  leaf resource {
    type leafref {
      path "/alarms/shelved-alarms/shelved-alarm/resource";
      require-instance false;
    }
    description
      "Compress the alarms with this resource.";
  }
  leaf alarm-type-id {
    type leafref {
      path "/alarms/shelved-alarms/shelved-alarm/
        + "alarm-type-id";
      require-instance false;
    }
    description
      "Compress alarms with this alarm-type-id.";
  }
  leaf alarm-type-qualifier {
    type leafref {
      path "/alarms/shelved-alarms/shelved-alarm/
        + "alarm-type-qualifier";
      require-instance false;
    }
    description
      "Compress the alarms with this alarm-type-qualifier.";
  }
}

output {
  leaf compressed-alarms {
    type uint32;
    description
      "Number of compressed alarm entries.";
  }
}
```

list alarm-profile {
  if-feature "alarm-profile";
  key "alarm-type-id alarm-type-qualifier-match resource";
  ordered-by user;
  description
```
"This list is used to assign further information or configuration for each alarm type. This module supports a mechanism where the client can override the system default alarm severity levels. The alarm-profile is also a useful augmentation point for specific additions to alarm types."

leaf alarm-type-id {
  type alarm-type-id;
  description
    "The alarm type identifier to match.";
}

leaf alarm-type-qualifier-match {
  type string;
  description
    "A W3C regular expression that is used to match the alarm type qualifier.";
}

leaf resource {
  type resource-match;
  description
    "Specifies which resources to match.";
}

leaf description {
  type string;
  mandatory true;
  description
    "A description of the alarm profile.";
}

container alarm-severity-assignment-profile {
  if-feature "severity-assignment";
  description
    "The client can override the system default severity level.";
  reference
    "ITU M.3100, ITU M.3160
     - Generic Network Information Model, Alarm Severity Assignment Profile";
  leaf-list severity-levels {
    type severity;
    ordered-by user;
    description
      "Specifies the configured severity level(s) for the matching alarm. If the alarm has several severity levels the leaf-list shall be given in rising severity order. The original M3100/M3160 ASAP function only allows for a one-to-one mapping between alarm type and severity but since the IETF alarm module supports stateful alarms the mapping must allow for several severity levels.";
}
Assume a high-utilisation alarm type with two thresholds with the system default severity levels of \( \text{threshold1} = \text{warning} \) and \( \text{threshold2} = \text{minor} \). Setting this leaf-list to \((\text{minor}, \text{major})\) will assign the severity levels \( \text{threshold1} = \text{minor} \) and \( \text{threshold2} = \text{major} \);

/* Notifications */

notification alarm-notification {
  description
  "This notification is used to report a state change for an alarm. The same notification is used for reporting 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 alarm-inventory-changed {
  description
  "This notification is used to report that the list of possible alarms has changed. This can happen when for example if a new software module is installed, or a new physical card is inserted.";
}

<CODE ENDS>

7. X.733 Extensions

Many alarm systems are based on the X.733, [X.733], and X.736 [X.736] alarm standards. This module augments the alarm inventory, the alarm lists and the alarm notification with X.733 and X.736 parameters.

The module also supports a feature whereby the alarm manager can configure the mapping from alarm types to X.733 event-type and probable-cause parameters. This might be needed when the default mapping provided by the system is in conflict with other management systems or not considered correct.
Note that the IETF Alarm Module term ‘resource’ is synonymous to the ITU term ‘managed object’.

8. The X.733 Mapping Module

This YANG module references [X.721], [X.733] and [X.736].

<CODE BEGINS>

```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;
  }

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  organization "IETF CCAMP Working Group";
  contact "WG Web:   <http://tools.ietf.org/wg/ccamp>
          WG List:  <mailto:ccamp@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 alarm parameters.

  The following structures are augmented with X.733 event type and probable cause:

  1) alarms/alarm-inventory: all possible alarm types
  2) alarms/alarm-list: every alarm in the system
  3) alarm-notification: notifications indicating alarm state changes
  4) alarms/shelved-alarms

  The module also optionally allows the alarm management system to configure the mapping from the IETF Alarm module alarm keys to the ITU tuple (event-type, probable-cause)."
```
The mapping does not include a corresponding X.733 specific problem value. The recommendation is to use the 'alarm-type-qualifier' leaf which serves the same purpose.

The module uses an integer and a corresponding string for probable cause instead of a globally defined enumeration, in order to be able to manage conflicting enumeration definitions. A single globally defined enumeration is challenging to maintain.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info).

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in the module text are to be interpreted as described in BCP 14 [RFC 2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This version of this YANG module is part of RFC XXXX (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for full legal notices."

reference

"ITU Recommendation X.733: Information Technology
 - Open Systems Interconnection
 - System Management: Alarm Reporting Function";

revision 2018-11-22 {

description

"Initial revision.";
reference "RFC XXXX: YANG Alarm Module";
}

/

* Features
*/

feature configure-x733-mapping {

description

"The system supports configurable X733 mapping from
the IETF alarm module alarm-type to X733 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.";
}
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.";
}
enum processing-error-alarm {
  value 4;
  description
  "An alarm of this type is principally associated with a
  software or processing fault.";
}
enum equipment-alarm {
  value 5;
  description
  "An alarm of this type is principally associated with an
  equipment fault.";
}
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.";
}
enum integrity-violation {

value 7;
   description
   "An indication that information may have been illegally
   modified, inserted or deleted.";
}

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.";
}

enum physical-violation {
   value 9;
   description
   "An indication that a physical resource has been violated
   in a way that suggests a security attack.";
}

enum security-service-or-mechanism-violation {
   value 10;
   description
   "An indication that a security attack has been detected by
   a security service or mechanism.";
}

enum time-domain-violation {
   value 11;
   description
   "An indication that an event has occurred at an unexpected
   or prohibited time.";
}
}

description
"The event types as defined by X.733 and X.736.";
reference
"ITU Recommendation X.733: Information Technology
 - Open Systems Interconnection
   - System Management: Alarm Reporting Function
ITU Recommendation X.736: Information Technology
 - Open Systems Interconnection
   - System Management: Security Alarm Reporting Function";
}

typedef trend {
   type enumeration {
      enum less-severe {
         description
         "There is at least one outstanding alarm of a
         severity higher (more severe) than that in the
typedef value-type {
  type union {
    type int64;
    type uint64;
    type decimal64 {
      fraction-digits 2;
    }
  }
} description
"A generic union type to match ITU choice of integer and real."

/*
 * 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."
}
leaf probable-cause-string {
  type string;
  description
  "The user friendly string matching
  the probable cause integer value. The string
  SHOULD match the X.733 enumeration. For example,
  value 27 is 'localNodeTransmissionError'."
}

container threshold-information {
  description
  "This parameter shall be present when the alarm
  is a result of crossing a threshold."
  leaf triggered-threshold {
    type string;
    description
    "The identifier of the threshold attribute that
    caused the notification."
  }
  leaf observed-value {
    type value-type;
    description
    "The value of the gauge or counter which crossed
    the threshold. This may be different from the
    threshold value if, for example, the gauge may
    only take on discrete values."
  }
  choice threshold-level {
    description
    "In the case of a gauge the threshold level specifies
    a pair of threshold values, the first being the value
    of the crossed threshold and the second, its corresponding
    hysteresis; in the case of a counter the threshold level
    specifies only the threshold value."
    case up {
      leaf up-high {
        type value-type;
        description
        "The going up threshold for rising the alarm."
      }
      leaf up-low {

type value-type;
description "The threshold level for clearing the alarm. This is used for hysteresis functions for gauges.";
}
}
case down {
leaf down-low {
  type value-type;
description "The going down threshold for rising the alarm.";
}
leaf down-high {
  type value-type;
description "The threshold level for clearing the alarm. This is used for hysteresis functions for gauges.";
}
}
leaf arm-time {
  type yang:date-and-time;
description "For a gauge threshold, the time at which the threshold was last re-armed, namely the time after the previous threshold crossing at which the hysteresis value of the threshold was exceeded thus again permitting generation of notifications when the threshold is crossed. For a counter threshold, the later of the time at which the threshold offset was last applied, or the time at which the counter was last initialized (for resettable counters).";
}
}
list monitored-attributes {
  uses attribute;
  key "id";
description "The Monitored attributes parameter, when present, defines one or more attributes of the resource and their corresponding values at the time of the alarm.";
}
leaf-list proposed-repair-actions {
  type string;
description "This parameter, when present, is used if the cause is known and the system being managed can suggest one or more solutions (such as switch in standby equipment,";
retry, replace media)."
}

leaf trend-indication {
  type trend;
  description
  "This parameter specifies the current severity trend of the resource. If present it indicates that there are one or more alarms (‘outstanding alarms’) which have not been cleared, and pertain to the same resource as that to which this alarm (‘current alarm’) pertains. The possible values are:

more-severe: The Perceived severity in the current alarm is higher (more severe) than that reported in any of the outstanding alarms.

no-change: The Perceived severity reported in the current alarm is the same as the highest (most severe) of any of the outstanding alarms.

less-severe: There is at least one outstanding alarm of a severity higher (more severe) than that in the current alarm."
}

leaf backedup-status {
  type boolean;
  description
  "This parameter, when present, specifies whether or not the object emitting the alarm has been backed-up, and services provided to the user have, therefore, not been disrupted. The use of this field in conjunction with the severity field provides information in an independent form to qualify the seriousness of the alarm and the ability of the system as a whole to continue to provide services. If the value of this parameter is true, it indicates that the object emitting the alarm has been backed-up; if false, the object has not been backed-up.";
}

leaf backup-object {
  type al:resource;
  description
  "This parameter shall be present when the Backed-up status parameter is present and has the value true. This parameter specifies the managed object instance that is providing back-up services for the managed object about which the notification pertains. This parameter is useful, for example, when the back-up object is from a pool of
any of which may be dynamically allocated to replace a faulty object.

} list additional-information {
    key "identifier";
    description
    "This parameter allows the inclusion of a set of additional information in the alarm. It is a series of data structures each of which contains three items of information: an identifier, a significance indicator, and the problem information.";
    leaf identifier {
        type string;
        description
        "Identifies the data-type of the information parameter.";
    }
    leaf significant {
        type boolean;
        description
        "Set to true if the receiving system must be able to parse the contents of the information subparameter for the event report to be fully understood.";
    }
    leaf information {
        type string;
        description
        "Additional information about the alarm.";
    }
}
leaf security-alarm-detector {
    type al:resource;
    description
    "This parameter identifies the detector of the security alarm.";
}
leaf service-user {
    type al:resource;
    description
    "This parameter identifies the service-user whose request for service led to the generation of the security alarm.";
}
leaf service-provider {
    type al:resource;
    description
    "This parameter identifies the intended service-provider of the service that led to the generation of the security alarm.";
}
grouping x733-alarm-definition-parameters {
    description "Common X.733 parameters for alarm definitions. This grouping is used to define those alarm attributes that can be mapped from the alarm-type mechanism in the ietf-alarm module.";
    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 management application.";
    }
    leaf probable-cause-string {
        type string;
        description "This string can be used to give a user friendly string to the probable cause value.";
    }
}

grouping attribute {
    description "A grouping to match the ITU generic reference to an attribute.";
    leaf id {
        type al:resource;"
description
"The resource representing the attribute.";
}
leaf value {
  type string;
  description
  "The value represented as a string since it could
  be of any type.";
}
reference
"ITU Recommendation X.721: Information Technology
  - Open Systems Interconnection
  - Structure of management information:
    Definition of management information
  Module Attribute-ASN1Module";
}

/*
 * Add X.733 parameters to the alarm definitions, 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;
}

/*
 * Add X.733 configurable mapping.
 */

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;
}

  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>

9. IANA Considerations

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registrations are requested to be made.

Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document registers two YANG modules in the YANG Module Names registry [RFC6020].
10. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/alarms/control/notify-status-change: This leaf controls whether an alarm should notify only raise and clear or all severity level changes. Unauthorized access to leaf could have a negative impact on operational procedures relying on fine-grained alarm state change reporting.

/alarms/control/alarm-shelving/shelf: This list controls the shelving (blocking) of alarms. Unauthorized access to this list could jeopardize the alarm management procedures since these alarms will not be notified and not be part of the alarm list.

Some of the operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:
/alarms/alarm-list/purge-alarms: This action deletes alarms from the alarm list. Unauthorized use of this action could jeopardize the alarm management procedures since the deleted alarms may be vital for the alarm management application.

11. Acknowledgements

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

The authors also wish to thank Nick Hancock, Joey Boyd, Tom Petch and Balazs Lengyel for their extensive reviews and contributions to this document.

12. References

12.1. Normative References


12.2. Informative References

[ALARMIRP]
Appendix A. Vendor-specific Alarm-Types Example

This example shows how to define alarm-types in a vendor-specific module. In this case the vendor "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-type-id;
    }

    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".
<alarm-inventory>
<alarm-type>
<alarm-type-id>xyz-al:link-alarm</alarm-type-id>
<alarm-type-qualifier/>
<resource>/dev:interfaces/dev:interface</resource>
<has-clear>true</has-clear>
<description>Link failure, operational state down but admin state up</description>
</alarm-type>
<alarm-type>
<alarm-type-id>xyz-al:environmental-alarm</alarm-type-id>
<alarm-type-qualifier>smoke-alarm</alarm-type-qualifier>
<has-clear>true</has-clear>
<description>Connected smoke detector to digital input</description>
</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.

<alarm-list>
<number-of-alarms>1</number-of-alarms>
<last-changed>2018-04-08T08:39:50.00Z</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/>
<time-created>2018-04-08T08:20:10.00Z</time-created>
<is-cleared>false</is-cleared>
<alt-resource>1.3.6.1.2.1.2.2.1.1.17</alt-resource>
<last-raised>2018-04-08T08:39:40.00Z</last-raised>
</alarm>
</alarm-list>
</alarms>
<last-changed>2018-04-08T08:39:50.00Z</last-changed>
<perceived-severity>major</perceived-severity>
<alarm-text>
  Link operationally down but administratively up
</alarm-text>
<status-change>
  <time>2018-04-08T08:39:40.00Z</time>
  <perceived-severity>major</perceived-severity>
  <alarm-text>
    Link operationally down but administratively up
  </alarm-text>
</status-change>
<status-change>
  <time>2018-04-08T08:30:00.00Z</time>
  <perceived-severity>cleared</perceived-severity>
  <alarm-text>
    Link operationally up and administratively up
  </alarm-text>
</status-change>
<status-change>
  <time>2018-04-08T08:20:10.00Z</time>
  <perceived-severity>major</perceived-severity>
  <alarm-text>
    Link operationally down but administratively up
  </alarm-text>
</status-change>
<operator-state-change>
  <time>2018-04-08T08:39:50.00Z</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>
                <name>FE10</name>
                <resource>
                    /dev/interfaces/dev:interface[name='FastEthernet1/0']
                </resource>
            </shelf>
            <shelf>
                <name>detectortest</name>
                <alarm-type-id>xyz-al:environmental-alarm</alarm-type-id>
                <alarm-type-qualifier-match>smoke-alarm</alarm-type-qualifier-match>
            </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>
Appendix F. Relationship to other alarm standards

This section briefly describes how this alarm module relates to other relevant standards.

F.1. Alarm definition

The table below summarizes relevant definitions of the term "alarm" in other alarm standards.

<table>
<thead>
<tr>
<th>Standard</th>
<th>Definition</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>X.733</td>
<td>error: A deviation of a system from normal operation. fault: The physical or algorithmic cause of a malfunction. Faults manifest themselves as errors. alarm: A notification, of the form defined by this function, of a specific event. An alarm may or may not represent an error.</td>
<td>The X.733 alarm definition is focused on the notification as such and not the state. It also uses the basic criteria of deviation from normal condition. There is no requirement for an operation action to be required.</td>
</tr>
<tr>
<td>G.7710</td>
<td>Alarms are indications that are automatically generated by an NE as a result of the declaration of a failure.</td>
<td>The G.7710 definition is close to the original X.733 definition.</td>
</tr>
<tr>
<td>Alarm MIB</td>
<td>Alarm: Persistent indication of a fault. Fault: Lasting error or warning condition. Error: A deviation of a system from normal operation.</td>
<td>RFC 3877 defines alarm referring back to &quot;a deviation from normal operation&quot;. This is problematic, since this might not require an operator action. The alarm MIB is state oriented rather than notification oriented, an alarm is a &quot;lasting condition&quot;, not a discrete notification reporting about a condition state change.</td>
</tr>
</tbody>
</table>
### Table 1: Definition of alarm in standards

The evolution of the definition of alarm moves from focused on events reporting a deviation from normal operation towards a definition to a undesired *state* which *requires an operator action*.
F.2. Data model

This section describes how this YANG alarm module relates to other standard data models. Note well that we cover other data-models for alarm interfaces. Not other standards such as SDO specific alarms for example.

F.2.1. X.733

X.733 has acted as a base for several alarm data models over the year. The YANG alarm module differs in the following ways:

X.733 models the alarm list as a list of notifications. The YANG alarm module defines the alarm list as the current alarm states for the resources, which is generated from the state change reporting notifications.

In X.733 an alarm can have the severity level clear. In the YANG alarm module "clear" is not a severity level, it is a separate state of the alarm. An alarm can have the following states for example (major, cleared), (minor, not cleared)

X.733 uses a flat globally defined enumerated "probable cause" to identify alarm types. This alarm module uses a hierarchical YANG identity, alarm-type. This enables delegation of alarm types within organizations. It also lets management reason about "abstract" alarm-types corresponding to base identities, see Section 3.2.

The YANG alarm module has not included the majority of the X.733 alarm attributes. Rather these are defined in an augmenting module if "strict" X.733 compliance is needed.

F.2.2. RFC 3877, the Alarm MIB

The MIB in RFC 3877 takes a different approach, rather than defining a concrete data model for alarms, it defines a model to map existing SNMP managed objects and notifications into alarm states and alarm notifications. This was necessary since MIBs were already defined with both managed objects and notifications indicating alarms, for example linkUp and linkDown notifications in combination with ifAdminState and ifOperState. So RFC 3877 can not really be compared to the alarm YANG module in that sense.

The Alarm MIB maps existing MIB definitions into alarms, alarmModelTable. The upside of that is that a SNMP Manager can at runtime read the possible alarm types. This corresponds to the alarmInventory in the alarm YANG module.
F.2.3. 3GPP Alarm IRP

The 3GPP Alarm IRP is an evolution of X.733. Main differences between the alarm YANG module and 3GPP are:

3GPP keeps the majority of the X.733 attributes, the alarm YANG module does not.

3GPP introduced overlapping and possibly conflicting keys for alarms, alarmId and (managed object, event type, probable cause, specific problem). (See Annex C in [X.733] Example 3). In the YANG alarm module the key for identifying an alarm instance is clearly defined by (resource, alarm-type, alarm-type-qualifier). See also Section 3.4 for more information.

The alarm YANG module clearly separates the resource/instrumentation life cycle from the operator life cycle. 3GPP allows operators to set the alarm severity to clear, this is not allowed by this module, rather an operator closes an alarm which does not affect the severity.

F.2.4. G.7710

G.7710 is different than the previous referenced alarm standards. It does define a data-model for alarm reporting. It defines common equipment management function requirements including alarm instrumentation. The scope is transport networks.

The requirements in G.7710 corresponds to features in the alarm YANG module in the following way:

Alarm Severity Assignment Profile (ASAP): the alarm profile "/alarms/alarm-profile/".

Alarm Reporting Control (ARC): alarm shelving "/alarms/control/alarm-shelving/" and the ability to control alarm notifications "/alarms/control/notify-status-changes". Alarm shelving corresponds to the use case of turning off alarm reporting for a specific resource, the NALM state in M.3100.

Appendix G. Alarm Usability Requirements

This section defines usability requirements for alarms. Alarm usability is important for an alarm interface. A data-model will help in defining the format but if the actual alarms are of low value we have not gained the goal of alarm management.
Common alarm problems and the cause of the problems are summarized in Table 2. This summary is adopted to networking based on the ISA [ISA182] and EEMUA [EEMUA] standards.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>How this module address the cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarms are generated but they are ignored by the operator.</td>
<td>&quot;Nuisance&quot; alarms (chattering alarms and fleeting alarms), faulty hardware, redundant alarms, cascading alarms, incorrect alarm settings, alarms have not been rationalized, the alarms represent log information rather than true alarms.</td>
<td>Strict definition of alarms requiring corrective response. Alarm requirements in Table 3.</td>
</tr>
<tr>
<td>When alarms occur, operators do not know how to respond.</td>
<td>Insufficient alarm response procedures and not well defined alarm types.</td>
<td>The alarm inventory lists all alarm types and corrective actions. Alarm requirements in Table 3.</td>
</tr>
<tr>
<td>The alarm display is full of alarms, even when there is nothing wrong.</td>
<td>Nuisance alarms, stale alarms, alarms from equipment not in service.</td>
<td>The alarm definition and alarm shelving.</td>
</tr>
<tr>
<td>During a failure, operators are flooded with so many alarms that they do not know which ones are the most important.</td>
<td>Incorrect prioritization of alarms. Not using advanced alarm techniques (e.g. state-based alarming).</td>
<td>State-based alarm model, alarm rate requirements in Table 4 and Table 5</td>
</tr>
</tbody>
</table>

Table 2: Alarm Problems and Causes

Based upon the above problems EEMUA gives the following definition of a good alarm:
<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant</td>
<td>Not spurious or of low operational value.</td>
</tr>
<tr>
<td>Unique</td>
<td>Not duplicating another alarm.</td>
</tr>
<tr>
<td>Timely</td>
<td>Not long before any response is needed or too late to do anything.</td>
</tr>
<tr>
<td>Prioritized</td>
<td>Indicating the importance that the operator deals with the problem.</td>
</tr>
<tr>
<td>Understandable</td>
<td>Having a message which is clear and easy to understand.</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>Identifying the problem that has occurred.</td>
</tr>
<tr>
<td>Advisory</td>
<td>Indicative of the action to be taken.</td>
</tr>
<tr>
<td>Focusing</td>
<td>Drawing attention to the most important issues.</td>
</tr>
</tbody>
</table>

Table 3: Definition of a Good Alarm

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

<table>
<thead>
<tr>
<th>Long Term Alarm Rate in Steady Operation</th>
<th>Acceptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than one per minute</td>
<td>Very likely to be unacceptable.</td>
</tr>
<tr>
<td>One per 2 minutes</td>
<td>Likely to be over-demanding.</td>
</tr>
<tr>
<td>One per 5 minutes</td>
<td>Manageable.</td>
</tr>
<tr>
<td>Less than one per 10 minutes</td>
<td>Very likely to be acceptable.</td>
</tr>
</tbody>
</table>

Table 4: Acceptable Alarm Rates, Steady State
<table>
<thead>
<tr>
<th>Number of alarms displayed in 10 minutes following a major network problem</th>
<th>Acceptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 100</td>
<td>Definitely excessive and very likely to lead to the operator to abandon the use of the alarm system.</td>
</tr>
<tr>
<td>20-100</td>
<td>Hard to cope with.</td>
</tr>
<tr>
<td>Under 10</td>
<td>Should be manageable - but may be difficult if several of the alarms require a complex operator response.</td>
</tr>
</tbody>
</table>

**Table 5: Acceptable Alarm Rates, Burst**

The numbers in Table 4 and Table 5 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 3

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 3?

3. Analyze and benchmark the performance of the system and compare it to the recommended metrics in Table 4 and Table 5. Start by identifying nuisance alarms, standing alarms at normal state and startup.

Authors’ Addresses

Stefan Vallin
Stefan Vallin AB

Email: stefan@wallan.se