OPEN NETWORKING
FOUNDATION

OpenFlow Notifications Framework
OpenFlow Management

Version 1.0
October 15, 2013

ONF TS-014

Contact: Dave Hood

ONF Document Type: OpenFlow Config
ONF Document Name: of-notifications-framework-1.0

Disclaimer

Contents

1 Introduction

This document specifies a framework for OpenFlow capable switches to notify events to OpenFlow configuration points, to OpenFlow controllers, and to other entities, including clients and managers north of the SDN controller. The framework aligns ONF notifications to telecommunications industry standards and practice, including a publish-and-subscribe notifications model and the mandatory and optional information elements appropriate to include in a notification message. Elements of protocol are not specified.

Section 2 describes the motivation, scope, and requirements of such a framework. The framework is specified in section 4, with the extensible information model defined in section 5. Notifications that lie outside the framework are described and listed in section 6. Section 7 identifies next steps, some of which could result in an update to the framework document.

The reader of this document is assumed to be familiar with the OpenFlow protocol [1] and the OpenFlow Management (OF-CONFIG) specification [2].

2 Motivation, Scope, and Requirements

A notifications mechanism is needed for asynchronously distributing information about events at OpenFlow capable switches, such as, for example, link failures, PM threshold crossings, configuration changes, or loss of connection to a controller[1].

This specification provides the framework that will be used to define specific events for OpenFlow capable and logical switches. These events will be defined in future specifications.

The notifications framework is based on a publish/subscribe mechanism and is open to be used for various tasks. The framework supports a clear separation of the publish/subscribe and notification mechanism on one side and the specification of event semantics and the information that is to be included in notifications on the other side.

Notification subscribers may include either or both of OpenFLow configuration points, OpenFlow controllers, and possibly other entities. The information model content and semantics are therefore fundamental. Mappings to specific protocols may be defined and extended over the course of time.

The framework recognizes existing notifications, both in OF-config and in OpenFlow-switch.

[1] Loss of controller connection would of course be signaled over one of the remaining controller connections, assuming that any exist.

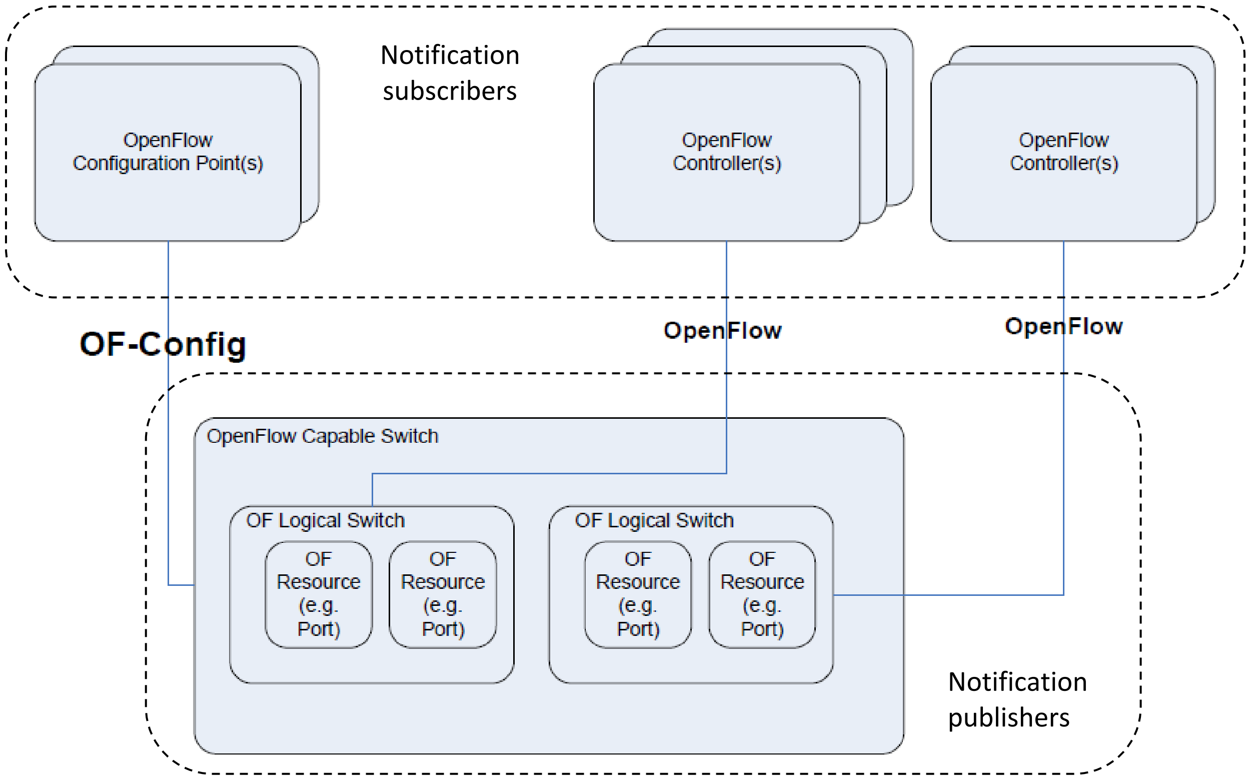
3 References

1. Open Networking Foundation. (2012). OpenFlow Switch Specification. Current revision available at https://www.opennetworking.org/sdn-resources/onf-specifications.
2. Open Networking Foundation. (2012). OpenFlow Management and Configuration Protocol (OF-CONFIG). Current revision available at https://www.opennetworking.org/sdn-resources/onf-specifications.
3. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., et al. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* , 69-74.
4. RFC 5277, Netconf event notifications
5. ITU-T M.3702 (2010), Common management services – Notification management – Protocol neutral requirement and analysis
6. 3GPP TS 32.301 (2012), Configuration Management (CM); Notification Integration Reference Point (IRP); Requirements (Release 11)
7. 3GPP TS 32.302 (2013), Configuration Management (CM); Notification Integration Reference Point (IRP); Information service (Release 11)
8. 3GPP TS 32.111 (2012), Fault Management; Part 2: Alarm Integration Reference Point (IRP); Information service (Release 11)
9. ITU-T X.730 (1992, as amended), Object management function
10. ITU-T X.731 (1992, as amended), State management function
11. ITU-T X.733 (1992, as amended), Alarm reporting function
12. ITU-T X.736 (1992, as amended), Security alarm reporting function
13. RFC 6470 (2012), Network configuration protocol (NETCONF) base notifications

4 Notifications Framework

For a notifications framework, there are the general alternatives of either specifying a new proprietary one or choosing an already existing notifications framework. To minimize effort and maximize compatibility with the industry, the ITU-T and TM Forum event frameworks are specified. Additional notification-related features or notification types may be added, preferably by requesting the appropriate standards bodies to address extensions that may be perceived as necessary.

An OF-capable switch acts as a notification publisher, to which clients may subscribe. Possible subscribers include OpenFlow config points and OpenFlow controllers, but the framework does not preclude other non-ONF subscribers as well.



If NETCONF is used as the transport protocol, subscription details are as defined in RFC 5277 [4]. They include the specification of a filter that requests the receipt of only some notifications, rather than all. Start and stop times may be specified. If specified, the start time must not be later than the instant of subscription, thereby invoking a best-efforts replay of notifications from the network element. A subscription may be cancelled, but neither suspended nor modified during its lifetime.

Another subscription model is defined in ITU-T M.3702 [5]. This model does not include replay of notification history, which would be a function of log management in ITU-T. As well as subscribe and unsubscribe functionality, the ITU-T model includes the ability to suspend and resume an active subscription. A filter can be modified during the lifetime of a subscription.

In the ITU-T model, one manager, for example, an OF-config point, can control subscriptions on behalf of some other entity, for example an OF controller. This is important, because some subscribers may have read-only access to the network. More specifically to ONF, the OpenFlow switch protocol [1] does not include commands that allow an OpenFlow controller to create or modify subscriptions. OpenFlow-switch defines a de facto subscription, a set of notifications that are enabled by default; if this set suffices, no further action need be taken on behalf of the controller. However, the framework allows for the possible modification of an OpenFlow controller's subscription if desired.

The ITU-T model is explicit that subscription criteria, including filters, be retrievable from the network element, and that managed entities can have their notifications administratively suppressed. NETCONF is not explicit about these points.

In TS 32.301 and TS 32.302, 3GPP also specifies a subscription and notifications model, which is aligned with ITU-T M.3702.

Continuing work in TM Forum also contemplates subscription, but in the interest of simplicity, TM Forum expects to require only create and delete subscription functions, with filters unchanged during the lifetime of the subscription.

Other entities, such as OF-config points and OF controllers may also publish notifications to higher-level clients. Such a publisher may originate notifications on its own behalf, or may filter or translate notifications received from lower-level publishers according to access control policy, abstract network views presented to clients, and possibly for other reasons.

5 Notifications information model

Mappings to various specific data models and transport protocols may be defined over the course of time. Mandatory fields must be present in all notification messages, and the message structure must provide for the possible presence of any or all optional fields.

5.1 Common information

ITU-T M.3702 defines the notifications model, including subscription capability and the skeleton of the notification report itself. Notifications are classified into several types [M.3702: notificationType], as further described in the ITU-T X.700 and M.3000 series of recommendations. Rather than a single flat notification space, notification types define a limited number of somewhat different formats, each of which contains information specific to that type. Notification types may be extended as needed.

As illustrated in figure 1, all notification messages start with a common set of information. These fields are mandatory.

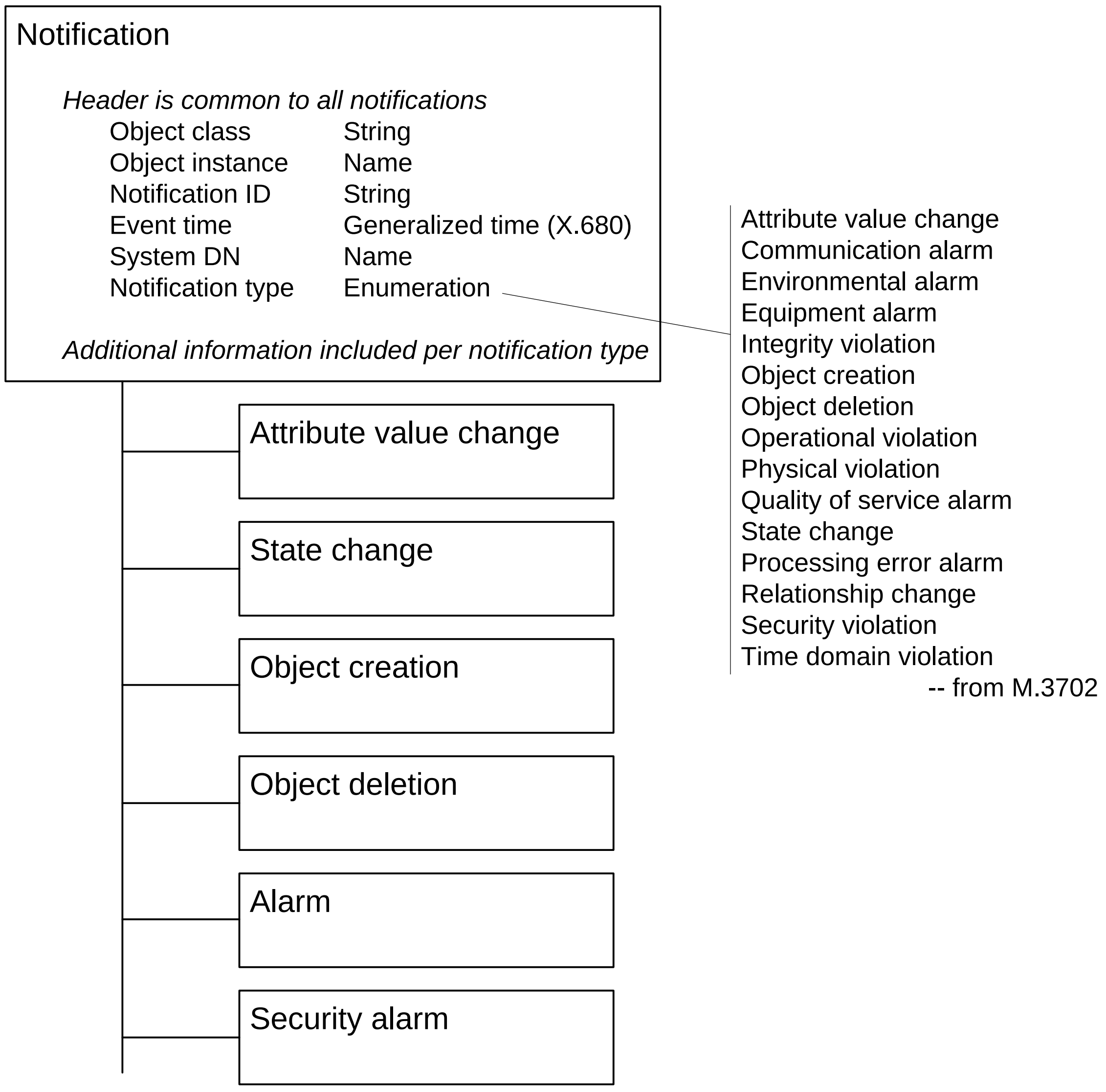


Figure 1. Notification structure

In MTOSI, TM Forum defines a number of additional notification types, listed below. This notifications framework document revision does not expand the TM Forum extensions, but they are candidates for future definition as appropriate.

- FileTransferStatusNotification
- HeartbeatNotification
- LogAttributeValueChangeNotification
- LogCapacityThresholdAlarmNotification
- LogProcessingErrorAlarmNotification
- LogStateChangeNotification
- ObjectDiscoveryNotification
- EquipmentProtectionSwitchNotification
- PmpStateChangeNotification
- ProtectionSwitchNotification

- TCANotification (probably redundant with alarm)
- RouteChangeNotification
- SoftwareBackupStatusNotification

Each notification type includes additional information appropriate to its nature, and most notification types are flexible in allowing fields to be omitted where they are not needed, and extended where additional information is useful.

5.2 Alarm notification

The network element is understood to contain the functions necessary to recognize anomalies, integrate them into defects, and further integrate defects into alarms and events, along with triggering additional action such as protection switching or the generation of AIS.

Refer to ITU-T recommendation X.733.

Notification types included in this category form an (extensible) subset of the types shown in figure 1:

- Communication alarm
- Environmental alarm
- Equipment alarm
- Quality of service alarm
- Processing error alarm

In addition to the mandatory header, the content of an alarm notification is illustrated in figure 2. Only the probable cause and perceived severity fields are mandatory. Additional information fields in lighter text are optional, to be used where appropriate.

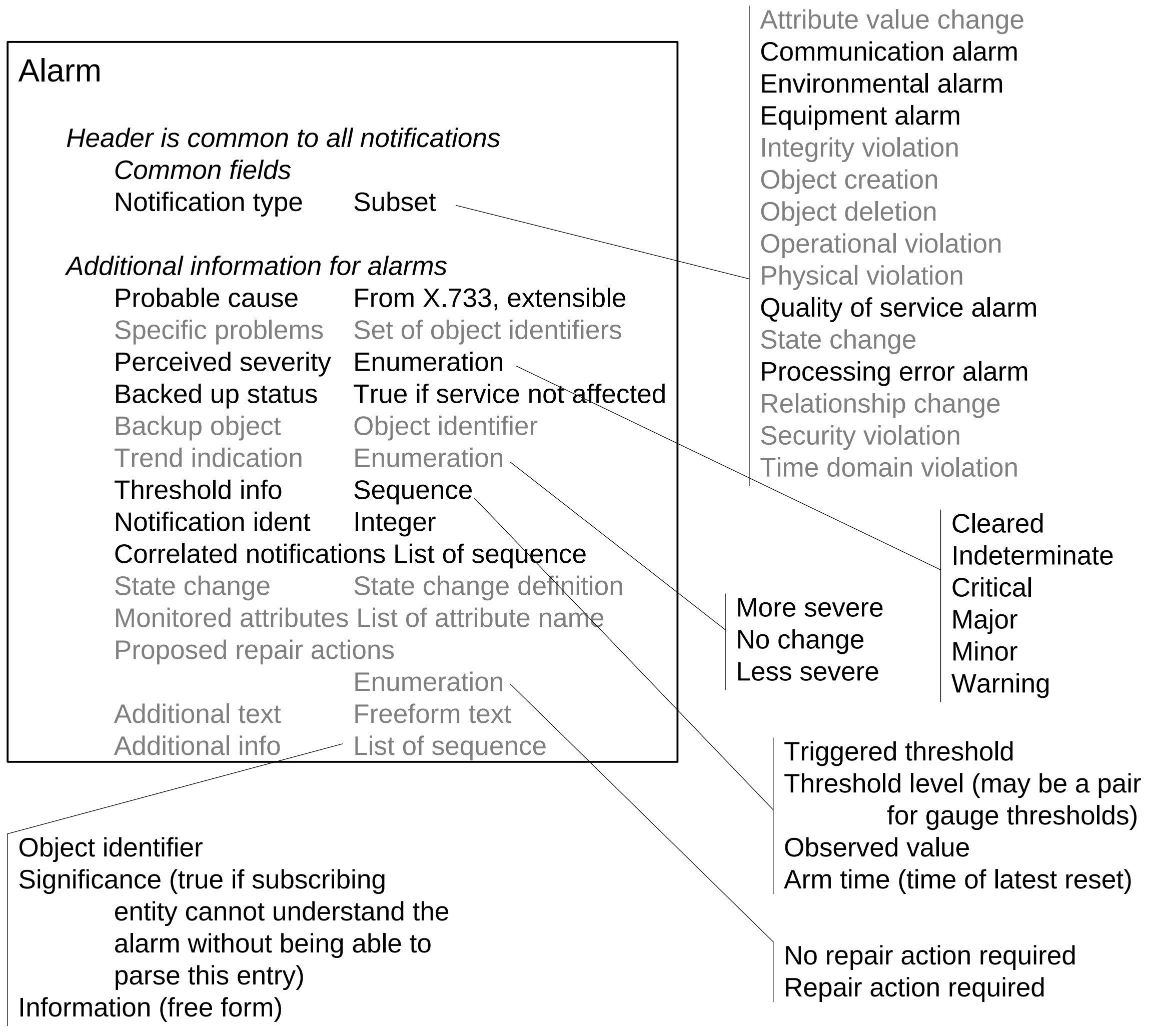

Alarm
Header is common to all notifications
Common fields
Notification type Subset
Additional information for alarms
Probable cause From X.733, extensible
Specific problems Set of object identifiers
Perceived severity Enumeration
Backed up status True if service not affected
Backup object Object identifier
Trend indication Enumeration
Threshold info Sequence
Notification ident Integer
Correlated notifications List of sequence
State change State change definition
Monitored attributes List of attribute name
Proposed repair actions Enumeration
Additional text Freeform text
Additional info List of sequence
Attribute value change
Communication alarm
Environmental alarm
Equipment alarm
Integrity violation
Object creation
Object deletion
Operational violation
Physical violation
Quality of service alarm
State change
Processing error alarm
Relationship change
Security violation
Time domain violation
Cleared
Indeterminate
Critical
Major
Minor
Warning
More severe
No change
Less severe
Triggered threshold
Threshold level (may be a pair for gauge thresholds)
Observed value
Arm time (time of latest reset)
No repair action required
Repair action required
Object identifier
Significance (true if subscribing entity cannot understand the alarm without being able to parse this entry)
Information (free form)

Figure 2. Alarm notification structure

- Probable cause, mandatory {set of specific alarm types}. The predefined list from X.733 is too long to quote here; X.733 is recommended as the first place to look, before extending the enumeration.
- Specific problem (optional, set of integers [enumeration] or set of MO identifiers)
- Perceived severity, mandatory {cleared, indeterminate, critical, major, minor, warning}. X.733 does not include a service-affecting boolean. Effect on service is included in critical/major/minor definitions.
- Backed up status (true indicates backed up)
- Backup object (present if B/U status = true, indicates MO providing backup service)
- Trend indication {more severe, no change, less severe}
- Threshold information (if alarm is due to threshold crossing)
    - triggered threshold identifier
    - threshold level (may be a pair of values for gauge threshold, value and hysteresis)
    - observed value
    - arm time (when the counter was last reset or the gauge re-armed)

- Notification identifier, arbitrary choice by publisher, should be unique over a reasonable time span
- Correlated notifications (list of notification identifiers)
- State change (if present, 2 notifications to be generated: 1 alarm, 1 state change)
- Monitored attributes (e.g. PM, list of {attribute, value at time of alarm})
- Proposed repair actions (optional, set of integers [enumeration] or set of MO identifiers; also {no repair action required, repair action required})
- Additional text, free-form, displayed on a screen or captured in a log
- Additional information (optional series of {identifier, significance [boolean: receiver is or is not required to be able to parse], problem information})

5.3 Security alarm

As with alarm notifications, the network element is understood to contain the functions necessary to recognize security anomalies and generate appropriate security alarms, possibly along with action to protect or recover its integrity.

Security alarms are distinguished from network alarms because their consumers, purposes and uses differ substantially. Refer to ITU-T recommendation X.736.

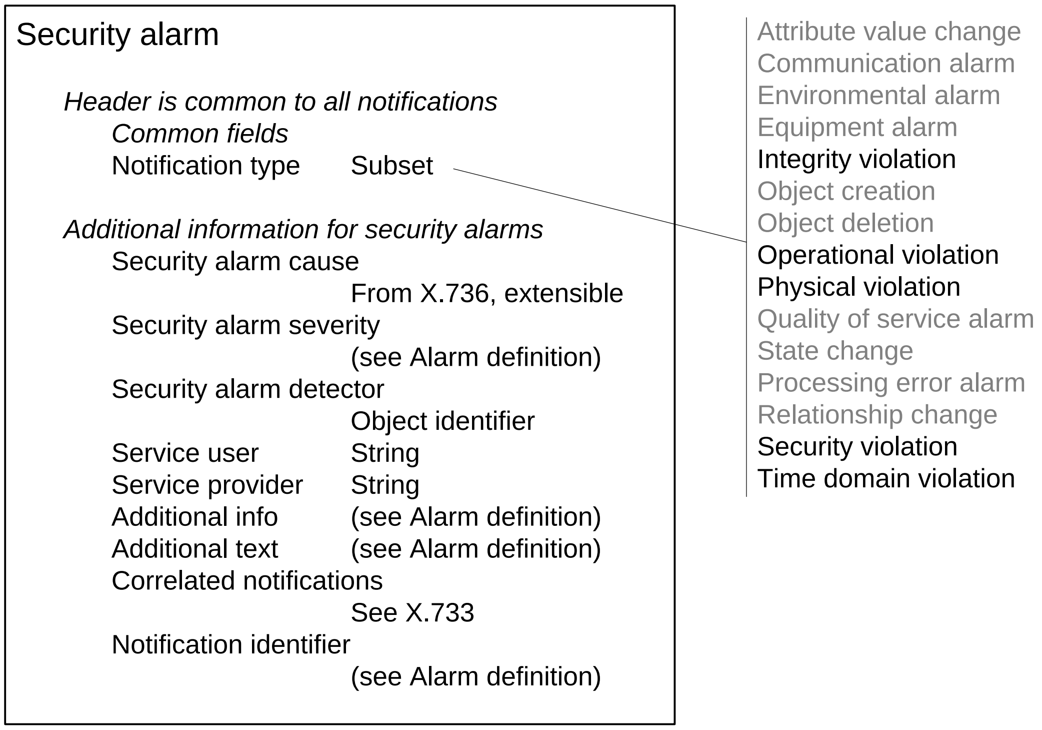


Figure 3. Security alarm notification structure

In the following description, fields not marked mandatory are optional.

- Notification type (mandatory) is a subset of the types illustrated in figure 1: {integrity violation, operational violation, physical violation, security service or mechanism violation, time domain violation}.

- The security alarm cause field allows further refinement. The following enumerations may be extended if appropriate.
  - Integrity violation {duplicate information, information missing, information modification detected, information out of sequence, unexpected information}
  - Operational violation {denial of service, out of service, procedural error, unspecified reason}
  - Physical violation {cable tamper, intrusion detection, unspecified reason}
  - Security service violation {authentication failure, breach of confidentiality, non-repudiation failure, unauthorized access attempt, unspecified reason}
  - Time domain violation {delayed information, key expired, out of hours activity}
- Security alarm severity, mandatory {indeterminate, critical, major, minor, warning}
- Security alarm detector, mandatory (MO ID)
- Service user, mandatory (originator of request that triggered the security alarm)
- Service provider, mandatory (intended provider of service)
- Notification identifier
- Correlated notifications (list of notification identifiers)
- Additional text
- Additional information (optional series of {identifier, significance [receiver required or not required to be able to parse], problem information})

5.4 Attribute value change

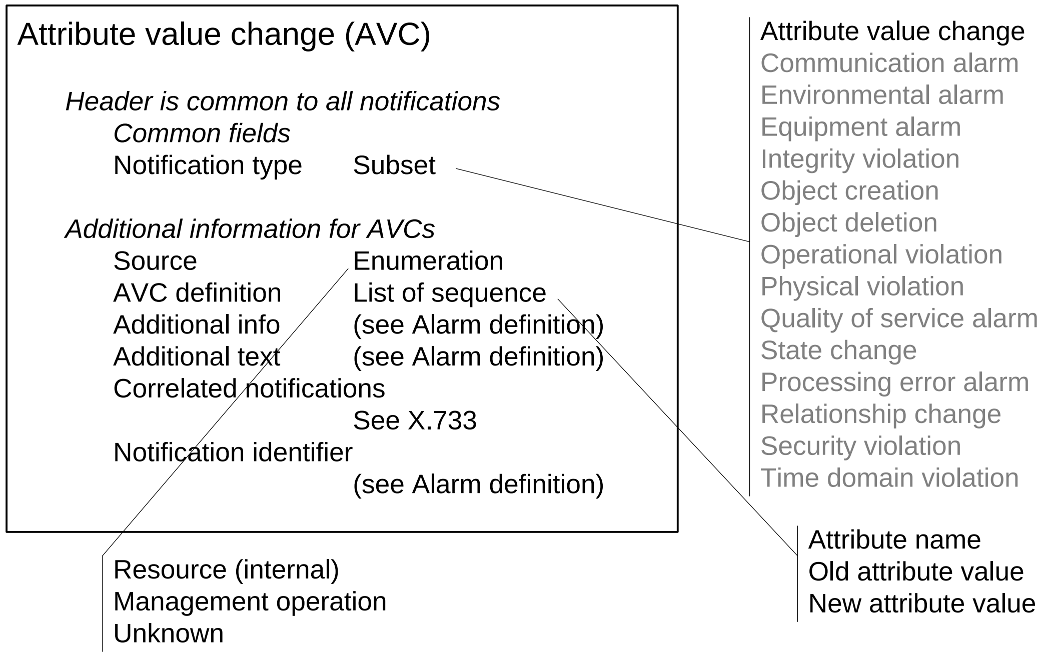


Figure 4. AVC notification structure

- Source {resource (ie internal operation), management operation, unknown}
- Attribute list
- AVC definition
  - Attribute identifier
  - Old attribute value

  - New attribute value
- Other information per X.733: {additional information, additional text, correlated notifications, notification identifier}

5.5 State change

Refer to ITU-T recommendation X.731. This notification is very similar to an AVC.

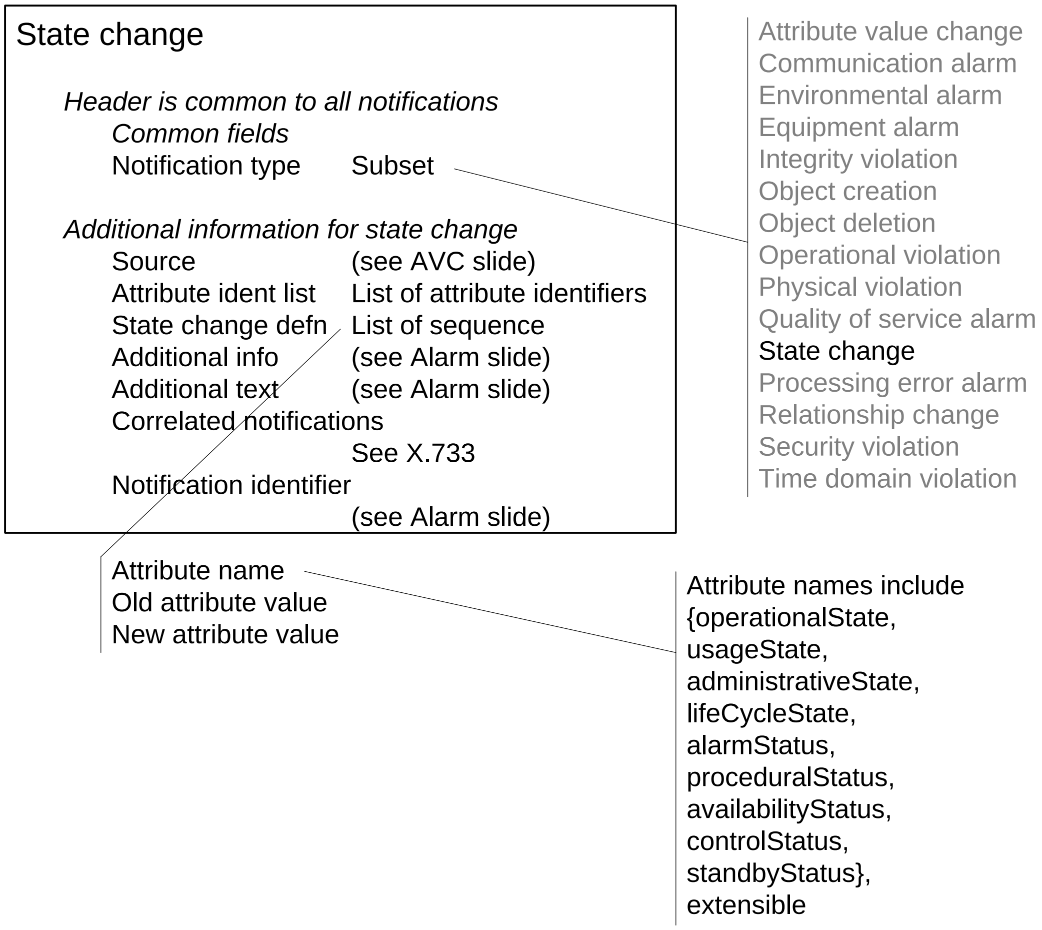

State change
Header is common to all notifications
Common fields
Notification type Subset
Additional information for state change
Source (see AVC slide)
Attribute ident list List of attribute identifiers
State change defn List of sequence
Additional info (see Alarm slide)
Additional text (see Alarm slide)
Correlated notifications See X.733
Notification identifier (see Alarm slide)
Attribute value change
Communication alarm
Environmental alarm
Equipment alarm
Integrity violation
Object creation
Object deletion
Operational violation
Physical violation
Quality of service alarm
State change
Processing error alarm
Relationship change
Security violation
Time domain violation
Attribute name
Old attribute value
New attribute value
Attribute names include {operationalState, usageState, administrativeState, lifeCycleState, alarmStatus, proceduralStatus, availabilityStatus, controlStatus, standbyStatus}, extensible

Figure 5. State change notification structure

- Source {resource (ie internal operation), management operation, unknown}
- Attribute list
- State change definition
  - Attribute identifier
  - Old attribute value
  - New attribute value
- Other information per X.733: {additional information, additional text, correlated notifications, notification identifier}

Defined attributes include state types {operational, usage, administration, lifecycle} and status types {alarm, procedural, availability, control, standby, unknown}. Each type can be extended or subsetted.

State types

- Operability {enabled, disabled} – refers to capability, not admin permissions
- Usage {active, idle, busy}. Active indicates some usage but not necessarily 100% of resource. Busy indicates no capacity for additional users
- Administration {locked, unlocked, shutting down} – whether allowed by management to perform its function
- Lifecycle {planned, installed, pending removal}

Status types

- Alarm {under repair, critical, major, minor, alarm outstanding}
- Procedural {init required, not initialized, initializing, reporting, terminating}
- Availability {in test, failed, power off, off line, off duty, dependency, degraded, not installed, log full}
- Control {subject to test, part of services locked, reserved for test, suspended}
- Standby {hot standby, cold standby, providing service}
- Unknown

5.6 Object creation and deletion

Refer to ITU-T recommendation X.730.

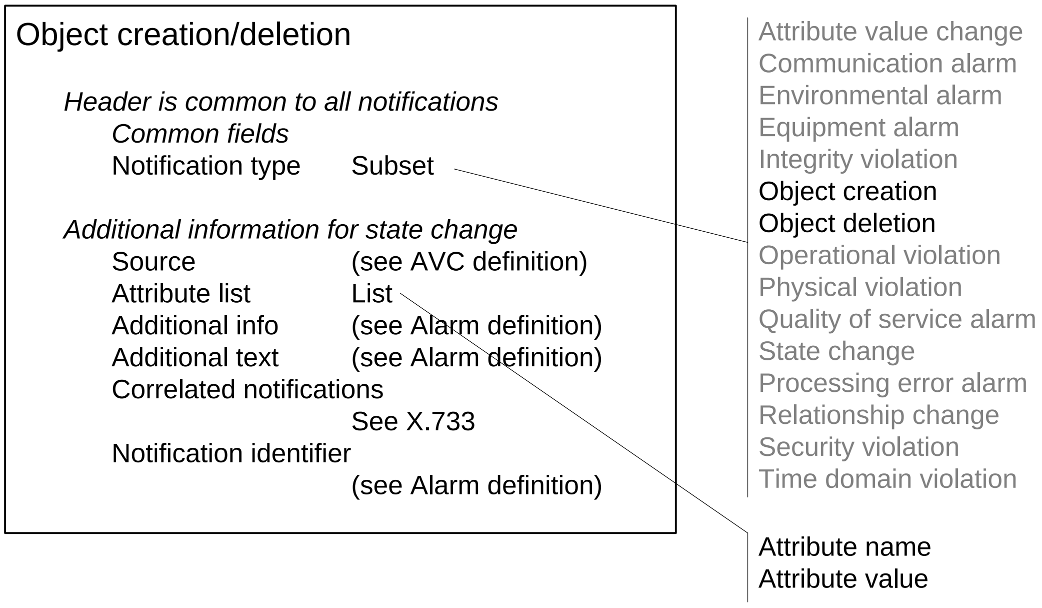


Figure 6. Object creation/deletion notification structure

- Source, i.e., cause of event {resource (ie internal operation), management operation, unknown}
- Attribute list
- Other information per X.733: {additional information, additional text, correlated notifications, notification identifier}

6 Pre-existing Notifications

A number of notifications existed before the notifications framework was developed. They do not necessarily fit into the framework, but are recognized by the framework. Such notifications may be extended ad hoc, although new notifications are strongly encouraged to align with the framework. Further, such existing notifications (port failure/recovery and port status, for example) may be redundant with future alarms or state change notifications defined in alignment with the framework.

The following lists are intended to be informative only. The lists may not be complete, and the notification details are specified elsewhere [1], [2].

6.1 OpenFlow switch notifications

- Packet-in
- Flow removed
- Port status
- Error
- Hello
- Echo request
- Echo reply
- Experimenter

6.2 OpenFlow config notifications

- OpenFlow logical switch instantiation
- OpenFlow capable switch capability change
- Successful OpenFlow session establishment
- Failed OpenFlow session establishment
- Port failure or recovery

7 Next steps

1. Fill in detail of publish and subscribe mechanism for OF-config and OF-switch, including how to specify, retrieve and modify a notifications filter.
2. As an integral part of information model development, define the notifications emitted by each object class, with attributes, ranges, and specializations as appropriate. Document the notificaions as an integral part of the information model. Consider collecting notifications together from the information model into a separate notifications-only view.
   a. Publish example notifications of the various types against managed object classes found in ONF specifications (protocol independent).
3. Topics for further study:
   a. Consider what notifications are appropriate beyond the scope of the information model, for example from the SDN infrastructure itself. Some of these notifications are already part of the ONF protocols, particularly as described in clause 6, but new ones may be needed over the course of time.
   b. Consider whether notification replay is required, optional or unnecessary. If required or optional, relate notification replay to a logging feature.