**3GPP TSG-SA5 Meeting #147 *S5-233557***

**Electronic meeting, 17 -25 April 2023**

**Source: Ericsson Hungary**

**Title: DP on FM restructuring TS 28-545**

**Document for: Information, Discussion, Endorsement**

**Agenda Item: 6.7.7.1**

# 1 Decision/action requested

***The group is asked to discuss and endorse the proposal.***

# 2 References

1. 3GPP TS 28.545 Fault Supervision (FS)
2. 3GPP TS 32.160 Management service template

# 3 Rationale

In clause 4 is a proposal how a new restuctured 28.545 could look potetially. It is a work in progress. The goal of this discussion paper is to agree on the principles of restructuring describe in this clause. Details of the new 28.545 will need to be agreed in a later submission.

## Main Proposals

Goal1: Restructure document to bring together stage 1,2 and maybe stage 3 items.

Goal-2: Improve FM documentation (28.545)

Goal-3: Separate generic mechanism for FM and the list of managed entities/fault reporting entities that use the generic use-cases. The latter may be documented in some other document. Don't list that FM supports NSI, NSSI, NF and who knows what else. This should be documented separately.

Replace triple documentation for NSI, NSSI, NF. Only one general description needed for use-cases, requirements, etc.

Remove current use-cases. Define new use cases for each use-case containing direct references to stage-2 notifications, NRM IOCs and operations.

Remove IRP related parts (operations like subscribe).

Merge "fault supervision data report " and "fault supervision data control" into a single service. The separation just creates confusion. It is very hard to justify why they are two separate service. Why would the notifications notifyChangedAlarm and notifyComments belong to different services. There is no advantage in the separation.

Reformat requirements according to TS 32.160

Add a definitions and concepts section.

Move detailed stage 2 and 3 definitions into this document.

### Things to do in 28.532

Remove all of 11.2 as it is moved (partly) to the new 28.545

During/before the move:

* Remove all of 11.2.2. Managed Information. Reference instead 28.622
* Remove operations for FM subscribe, unsubscribe, getAlarmList, getAlarmCount, setComment, acknowledgeAlarms, unacknowledgeAlarms, clearAlarms
* Move alarm state diagram to 28.545 (?)
* Cleaup correlation related problems (later step, not part of restructuring).

### Things to do in 28.622/623

Move \_3gpp-common-fm.yang to new 28.545

Move some YAML to new 28.545

Move AlarmList IOC and AlrmRecord datatype to new 28.545

During/before the move:

* Any information that is today transferred in notifications should be modeled in NRM and should be possible via the CRUD operations.
* Add to AlarmList IOC: alarmCount separately for each severity
* Add missing fields to AlarmRecord dataType: commentUserId, commentSystemId, commentText, commentTime, correlatedNotifications, others???. All new writable fields shall be optional to support.
* Things included in 28.532 alarmlist that maybe should be in the alarmRecordDataType (or maybe not): notificationType, systemDN,
* Add AlarmList.unreliableAlarmScope.

## General Editorial Questions Issues - to be removed later

TODO.specificationError correlation should be describe here in some detail.

* + Any type of notification can be correlated e.g. notifyEvent or notifyMOIChanges
* There are problems with correlations
  + Correlated notifications are missing from the AlarmRecord datatype
  + The correlatedNotifications paramer is not properly defined on stage 2 in 28.532. As notificationIds are unique only for a specific MOI correlatedNotifications should be a list of DN-notificationId pairs. This is not stated anywhere.
  + As a notifyChangedAlarm can change the notificationId in the Alarmlist, in order to make corelations the OSS needs to read all notificatiuons from the notification logs which are not readable in the NRM. BUG.
  + It is not the always notifications but sometimes the Alarms that should be correlated. However even CM notifications could be correlated.
  + If an alarm is reported with notifyNewAlarm and then later updted with notifyChangedAlarm which notification is correlated? New, changed, both?

Don't handle this during restructuring, but it should be handled in a next step

TODO.specificationError requirement REQ-FS-15 , REQ-FS-15 are unclear.

It could mean multiple different things. Does it mean that faults of the underlying virtual machines or kubernetes PODs should be reported via this service?

Is there an implementation for it? I havent seen any.

If it is informational should it be here?

Is this covered by the other FM requirements or will this result in the definition of a new NRM element, notification or operation?

* It should be clarified and implemented or removed.

Note to be considered later: there is no notification for rootCauseIndicator.

Note to be considered later: If ackSystemId, ackUserId is changed without changing the ackState there is no notification.

Note to be considered later: If clearUserId or clearSystemId is changed without changing the percievedSeverity there is no notification.

Note to be considered later: Is the order of the notification input parameters important? If yes, checks it is OK.

# 4 Detailed proposal of a potential solution

3GPP TS 28.XXX V18.x.x (2023-04)

Technical Specification

3rd Generation Partnership Project;

Technical Specification Group Services and System Aspects;

Management and orchestration;

Fault Supervision (FS);

(Release 18)



The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP..  
The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented.  
This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification.  
Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices.

.

Keywords

Management,orchestration,faul

.

***3GPP***

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis

Valbonne - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

***Copyright Notification***

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© 2021, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).

All rights reserved.

UMTS™ is a Trade Mark of ETSI registered for the benefit of its members

3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  
LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners

GSM® and the GSM logo are registered and owned by the GSM Association

Contents

1 Decision/action requested 1

2 References 1

3 Rationale 1

Main Proposals 1

Things to do in 28.532 1

Things to do in 28.622/623 2

General Editorial Questions Issues - to be removed later 2

4 Detailed proposal of a potential solution 3

Foreword 9

Introduction 9

1 Scope 10

2 References 10

3 Definitions and abbreviations 11

3.1 Definitions 11

3.2 Abbreviations 11

4 Concepts and overview 11

4.1 Overview 11

4.2 Concepts 11

4.2.1 Identifying an Alarm 11

4.2.2 Alarm States 12

4.2.2.1 State diagram 12

5 Requirements for fault supervision service 13

6 Fault Supervision service Components overview 15

7 Use Cases 16

7.1 Overview 16

7.2 Behaviour of the Fault Supervision service 16

7.3 Subscription to Alarm Notifications 16

7.4 Unsubscription of Alarm notifications 17

7.5 Report Alarm 17

7.5.1 Notify New Alarm 17

7.5.2 Notify Changed Alarm 18

7.6 Get Alarm List 18

7.7 Notify alarm list potentially faulty 18

7.8 Notify Alarm List Rebuilt 19

7.9 Acknowledge Alarms 19

7.10 Clear Alarms 19

7.11 Alarm loss detection 20

7.12 Virtualized resource alarm correlation 20

8.3 Class definitions 22

9 Operations and notifications 32

9.1 Overview 32

9.2 Common Notification Header 32

9.3 notifyNewAlarm 32

9.3.1 Definition 32

9.3.2 Input parameters 32

9.3.3 Triggering event 33

9.3.3.1 From-state 33

9.3.3.2 To-state 33

9.4 notifyClearedAlarm 33

9.4.1 Definition 33

9.4.2 Input parameters 34

9.4.3 Triggering event 34

9.4.3.1 From-state 34

9.4.3.2 To-state 34

9.5 notifyAlarmListRebuilt 34

9.5.1 Definition 34

9.5.2 Input parameters 35

9.5.3 Triggering event 35

9.5.3.1 From-state 35

9.5.3.2 To-state 35

9.6 notifyChangedAlarm 35

9.6.1 Definition 35

9.6.2 Input parameters 36

9.6.3 Triggering event 36

9.6.3.1 From-state 36

9.6.3.2 To-state 36

9.7 notifyChangedAlarmGeneral 36

9.7.1 Definition 36

9.7.2 Input parameters 37

9.7.3 Trigger event 37

9.7.3.1 From-state 37

9.7.3.2 To-state 37

9.8 notifyCorrelatedNotificationChanged 38

9.8.1 Definition 38

9.8.2 Input parameters 38

9.8.3 Triggering event 38

9.8.3.1 From-state 38

9.8.3.2 To-state 38

9.9 notifyAckStateChanged 38

9.9.1 Definition 38

9.9.2 Input parameters 38

9.9.3 Triggering event 39

9.9.3.1 From-state 39

9.9.3.2 To-state 39

9.10 notifyComments 39

9.10.1 Definition 39

9.10.2 Input parameters 39

9.10.3 Trigger event 39

9.10.3.1 From-state 39

9.10.3.2 To-state 40

9.11 notifyPotentialFaultyAlarmList 40

9.11.1 Definition 40

9.11.2 Input parameters 40

9.11.3 Trigger event 40

9.11.3.1 From-state 40

9.11.3.2 To-state 41

10 Stage 3 – Solution Sets 42

10.1 YANG Definitions 42

10.1.1 NRM Definitions 42

10.1.1 module \_3gpp-common-fm.yang 42

10.1.2 Notifications 47

10.2 JSON Definitions 47

10.1.1 NRM Definitions 47

10.2.1 OpenAPI document "TS28545\_FmNrm.yaml" 47

10.21.2 Notifications 47

Annex X (informative): Change history 48

V11 to v12 48

V10 to v11 48

V9 to v10 48

V8 to v9 49

v7 to v8 49

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

# Introduction

The present document is part of a TS-family covering the 3rd Generation Partnership Project Technical Specification Group Services and System Aspects Management and orchestration of networks, as identified below:

TS 28.545: Management and orchestration; Fault Supervision (FS)

# 1 Scope

This document describes the SBMA based Fault Supervision management service. It includes

- Fault service related concepts and definitions.

- Stage 1 – requirements

- Use-cases, explaining how NRM, operations and notifications shall work together to provide the functionality required.

- A list of Stage-2 components (operations, notifications, IOCs and datatypes) provided or used by fault supervision with references towards the relevant specifications.

- Stage 2 – NRM IOCs and notifications implemented by the Fault Supervision management service.

- Stage 3 – YANG and OpenApi solution sets

- Some additional considerations (alarm loss detection, virtualized resource alarm correlation)

This specification of the Fault Supervision MnS is based on the SBMA principles using CRUD operations, modeled OAM data in the NRM together with fault management specific notifications. An IRP based solution for fault management is out of scope for this document.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 28.532: "Management and orchestration; Management services".

[3] 3GPP TS 28.541: "Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3".

[4] 3GPP TS 28.516: "Fault Management (FM) for mobile networks that include virtualized network functions; Procedure".

[5] 3GPP TS 28.622: "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)".

[6] 3GPP TS 28.623: "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Solution Set (SS) definitions".

[7] 3GPP TS 32.111-2: " Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)".

[8] ITU-T Recommendation X.733 (02/92): "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

**Event:** Anything that occurs at a certain point in time, for example a configuration change, a threshold crossing, a transition to an error state or a transition to a failure state. Events do not have states.

**Error**: A state of the system different from the correct system state as defined by the service specification. An error may or may not lead to a service failure. An error has a begin and end time.

**Failure:** A state of inability to deliver the correct service as defined by the service specification. A service failure may be the result of an error or a poor service function design.

**Fault**: The (hypothesized or adjudged) cause for an error or a failure.

**MonitoredEntity**: Any class that can have an alarmed state.

**Alarm**: An error or failure thar requires attention or reaction by an operator or some machine. Alarms have state.

**Root cause:** The primary fault (cause), if any, leading to one or multiple errors or failures.

**Matching-Criteria-Attributes:** which identifies a set of ITU-T Recommendation X.733 [8] defined attributes.  
Notifications carrying identical values for these attributes are considered to be carrying alarm information related to (a) the same network resource and (b) the same alarmed condition. The matching-criteria-attributes are: *objectInstance, eventType, probableCause and specificProblem*, if present.

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

ADAC Automatically Detected and Automatically Cleared

ADMC Automatically Detected and Manually Cleared

CRUD Create, Read, Update, Delete basic data maipulation operations

FS Fault Supervision

ME Managed Element

MnS Management Service

NRM Network Resource Model

# 4 Concepts and overview

## 4.1 Overview

Fault Supervision is considered a generic service. It shall be able to support fault indications about any resource that can be addressed by a distinguished name e.g., ManagedElements, GNBDUs or NetworkSlices.

## 4.2 Concepts

### 4.2.1 Identifying an Alarm

In the NRM AlarmList object the individual alarm instances represented by a specific alarmRecord element in the *alarmRecord* is identified by the *alarmRecord.alarmId*.

Both in NRM and in the notifications alarms can be identified by the Matching-Criteria-Attributes; see clause “3.1 Definitions”.

### 4.2.2 Alarm States

An alarm is a stateful concept. It has the following states:

* **unack&unclear** – the alarm has been raised (created) but is neither acknowledge or cleared. If the MnS provider implements automatic acknowledgement this state might not be used.
* **ack&unclear** – the alarm has been raised (created) and acknowledge but has not been cleared.
* **unack&clear** – the alarm has been raised (created) and cleared, but is not acknowledged. If the MnS provider implements automatic acknowledgement this state might not be used.
* **ack&clear** – this is the transient, terminal state of the alarm. At this point the alarm is removed from the *AlarmList*, but usually still visible in (vendor specific) alarm log.

Alarm states are visible in the AlarmRecord in AlarmList indicated by the attribute fields *alarmRecord.ackState* and *alarmRecord.percievedSeverity*.

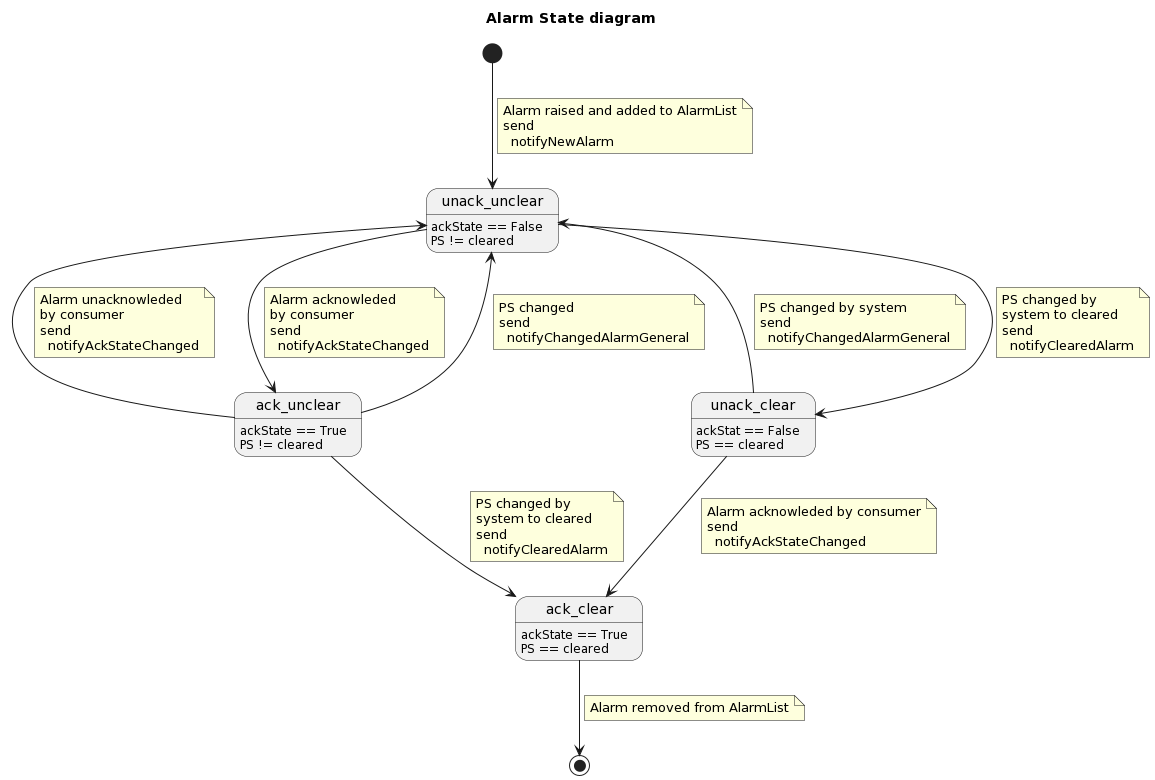
#### 4.2.2.1 State diagram

For brevity some possible activities are excluded. Data in the AlarmRecord may be updated e.g., backedUpStatus, backUpObject, trendIndication, thresholdInfo, stateChangeDefinition, monitoredAttributes, proposedRepairActions, additionalText, additionalInformation, serviceUser, serviceProvider or securityAlarmDetector, comments, correlated notifications.

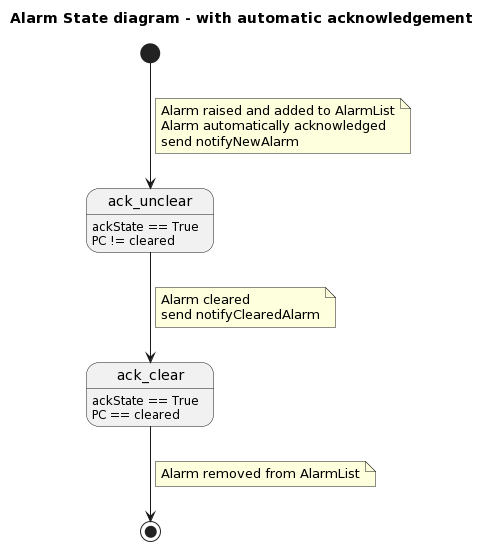
The only changes that result in alarm state changes are changing the *alarmRecord.ackState* and setting *alarmRecord.percievedSeverity* .

The solid circle icon represents the Start State. The double circle icon represents the End State. In this state, the alarm is cleared and acknowledged. The AlarmRecord shall not be accessible via the Service interface and is removed from the *AlarmList*.

"PS" used in the state diagram stands for "perceived severity".



If the system uses automatic acknowledgement (not allowing acknowledgement by the consumer) the following simpler state diagram is applicable:



# 5 Requirements for fault supervision service

The following requirements are valid for any MonitoredEntity.

|  |  |  |
| --- | --- | --- |
| **Requirement label** | **Description** | **Related use case(s)** |
| **REQ-FS-1** | The fault supervision service shall have the capability to provide **alarm notifications** to authorized consumers. | * Report Alarm * Notify New Alarm * Notify Changed Alarm * Notify Alarm List Rebuilt   Motivation: the consumer should receive information about alarms immediately when an alarm is raised or changed. |
| **REQ-FS-2** | The fault supervision service shall have the capability to allow authorized consumers to **subscribe** to alarm notifications. | * Subscription to Alarm Notifications   Motivation: Needed for REQ-FS-1. Providers will not send notification without an explicit subscription. |
| **REQ-FS-3** | The fault supervision service shall have the capability to allow authorized consumers to **unsubscribe** from alarm notifications. | * Unsubscription of Alarm notifications   Motivation: The consumer needs to be able to indicate that it is no longer interested in receiving immeditate alarm information |
| **REQ-FS-4** | The fault supervision service shall have the capability to allow authorized consumers to provide a **filter** for alarm **notifications**. | * Subscription to Alarm Notifications   Motivation: The consumer shall be able to indicate that it is interested only in a subset of alarms. |
| **REQ-FS-5** | The fault supervision service shall have the capability to allow authorized consumers to **retrieve the alarm list**. | * Get Alarm List   Motivation: The consumer shall be able to read all current alarms. It needs this if the sequence of received alarm notifications does not provide a reliable and complete view of the alarm situation. This may happen after the start-up of the consumer fault mananagement service, if the connection or some alarm notifications are lost, or if the alarm producer was not able to provide on-time indication of all alarm changes. |
| **REQ-FS-6** | The fault supervision service shall have the capability to allow authorized consumers to **retrieve a filtered** subset of the **alarm list**. | * Get Alarm List   Motivation: If the consumer is interested only in a subset of alarms, it shall be retrieve only that subset. |
| **REQ-FS-7** | The fault supervision service shall have the capability to provide **changed alarm notifications** to its authorized consumer. | * Notify Changed Alarm   Motivation: the consumer should receive information about changed alarms immediately. |
| **REQ-FS-8** | The fault supervision service shall have the capability to provide **cleared alarm notifications** to its authorized consumer. | * Notify Changed Alarm   Motivation: the consumer should receive information about cleared alarms immediately. |
| **REQ-FS-9** | The fault supervision service shall have the capability to provide **new** generated **alarm notifications** to its authorized consumer. | * Notify New Alarm   Motivation: the consumer should receive information about alarms immediately when an alarm is raised. |
| **REQ-FS-10** | The fault supervision service shall have the capability to provideindicate that the **alarm list** is **potentially faulty** and also that the **alarm list rebuilt notification**s to authorized consumers whenever the alarm list is rebuilt. | * Notify alarm list potentially faulty * Notify alarm list rebuilt   Motivation: the consumer should receive information when the alarm list is corrupt or out-of-date. The consumer should also be notified when when the correct alarm information is available again. |
| **REQ-FS-11** | The fault supervision shall have the capability to satisfy the request to **acknowledge** one or multiple **alarms**. If this capability is not supported, then the producer shall be able to automatically acknowledge alarms. | * Acknowledge Alarms   Motivation: the consumer should be able to register in the producer that it has received the alarm and has done some vendor specific level of processing of the alarm information. |
| **REQ-FS-12** | The fault supervision shall have the capability to satisfy the request to **clear** one or multiple **alarms**. This capability is only required if one or more of the alarms supported by the producer is of type ADMC. | * Clear Alarms |
| **REQ-FS-13** | The fault supervision shall have the capability to provide **acknowledgement** state change **notifications** to its authorized consumer. | * Notify Changed Alarms   Motivation: If the producer supports ADMC alarms, the consumer shall be able to clear those. |
| **REQ-FS-14** | The fault supervision data report service for NF shall have the capability to provide alarm notifications of virtualized resources correlated with an NF instance to its authorized consumer. | * ???   Motivation: ??? |
| **REQ-FS-15** | The fault supervision data report service for NF shall have the capability to provide the alarm list of virtualized resources correlated with an NF instance. | * ???   Motivation: ??? |

# 6 Fault Supervision service Components overview

The service consists of a mandatory “Base” feature and three that

List of components used by Fault Supervision:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **S** | **Component A (Operations, notifications)** | **Component B (NRM)** | **Implemented or used\*** |
| FS NmS Base | M | createMOI |  | Used |
| M | getMOIAttributes |  | Used |
| M | modifyMOIAttributes |  | Used |
| M | deleteMOI |  | Used |
| M | notifyNewAlarm |  | Implemented |
| M | notifyClearedAlarm |  | Implemented |
| O | notifyChangedAlarm |  | Implemented |
| M | notifyAlarmListRebuilt |  | Implemented |
| O | notifyCorrelatedNotificationChanged |  | Implemented |
| O | notifyChangedAlarmGeneral |  | Implemented |
| O | notifyPotentialFaultyAlarmList |  | Implemented |
| O | notifyMOICreation |  | Used |
| O | notifyMOIAttributeValueChanges |  | Used |
| O | notifyMOIDeletion |  | Used |
| O | notifyMOIChanges |  | Used |
| M |  | AlarmList, IOC | Implemented |
| M |  | AlarmRecord, dataType | Implemented |
|  |  | CorrelatedNotification, datatype |  |
| M |  | NtfSubscriptionControl, IOC | Used |
| O |  | HeartbeatControl, IOC | Used |
| Acknowledgment of alarms | O | notifyAckStateChanged | AlarmRecord.ackTime AlarmRecord.ackUserId  AlarmRecord.ackSystemId  AlarmRecord.ackState | Implemented |
| Clearing alarms by the consumer | O |  | AlarmRecord.clearUserId  AlarmRecord.clearSystemId | Implemented |
| Setting comments for alarms | O | notifyComments | AlarmComment dataType  AlarmRecord.comments | Implemented |

* Indicates whether the component is implemente by the Fault Supervision service or implemented by another serviced and just used by Fault Supervision.
* Some parts of the AlarmRecord dataType are only needed for the support the optional features.



# 7 Use Cases

## 7.1 Overview

Fault supervision uses SBMA principles to set and read data in the NRM. CRUD operations (createMOI, getMOIAttributes, modifyMOIAttributes , deleteMOI) implemented by the provisioning MnS are used to manipulate fault supervision related data. Fault supervision itself does not implement any operations. Additionally, notifications are also used.

NRM data is written to control the behavior of the fault supervision.

Data provided to the fault supervision consumer is made available in two ways. Data that should be provided as soon as it is available in the MnS provider is sent to subscribed and authorized MnS consumers in notifications (e.g., information about a new alarm). Other data is available to the authorized MnS consumers using the provisioning getMOIAttributes operation (e.g., information about a subscription).

To receive notifications consumers need to subscribe to them. When notifications are no longer needed the consumer shall unsubscribe.

Note: *The name of operations, notifications IOCs, attributes defined in stage 2 and stage 3 in clauses 8,9 and 10 are indicated in italics*.

## 7.2 Behaviour of the Fault Supervision service

When the MnS producer is installed or started it automatically creates one *AlarmList* MOI. The management scope of an *AlarmList* MOI is defined by all descendant objects of the base managed object, which is the object name-containing the *AlarmList*, and the base object itself.

Individual alarms are represented by an attribute element (an individual value) of the *AlarmList.alarmRecords* attribute; this attribute element will be referred to as the alarmRecord in this document. An alarmRecord is created and modified by the MnS producer as a result of a new alarm being raised. It cannot be created by the MnS consumer.

Alarm records are maintained only for active alarms. Inactive alarms are automatically deleted by the MnS producer from the *AlarmList*. Active alarms are alarms whose

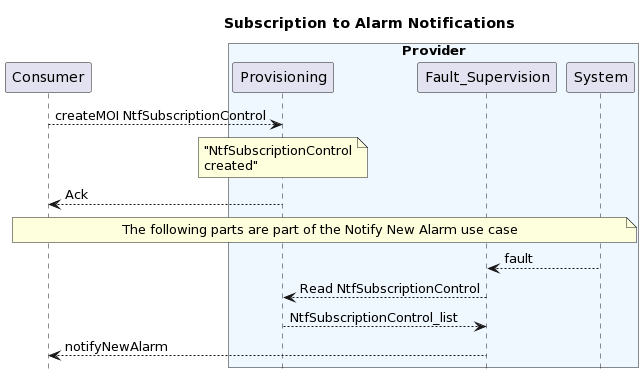
a) *perceivedSeverity* is not "CLEARED", or whose

b) *perceivedSeverity* is "CLEARED" and its *ackState* is not "ACKNOWLEDED".

The *AlarmList.administrativeState* attribute can be used to enable/disable Fault supervision. When the alarm list is locked or disabled, the existing alarm records are not updated or deleted, new alarm records are not added to the alarm list and no alarm notifications are sent. When the *AlarmList* is enabled the MnS provider should update or rebuild the *AlarmList* and send out the corresponding notifications (see clause 7.5, 7.7).

## 7.3 Subscription to Alarm Notifications

To receive notifications consumer needs to subscribe to them. A subscription can be created by the consumer by using the *createMOI* operation to create a new *NtfSubscriptionControl* MOI. The MOI attributes include data about the recipient of the notifications and attributes to filter which notification types are sent and the scope of resources about which the faults are reported. For details about *NtfSubscriptionControl* MOI see TS 28.622 [5]. Usually, the consumer should subscribe to all four of the *notifyNewAlarm, notifyChangedAlarm, notifyAlarmListRebuilt, notifyCorrelatedNotificationChanged notificationTypes*.



The consumer can modify its subscription at any time using the *modifyMOIAttributes* operation targeted at the *NtfSubscriptionControl* MOI previously.

If requested by the consumer the MnS provider will provide Heartbeat notifications to the consumer. These can be used to monitor that the Fault supervision service is up and running at the MnS provider. Heartbeats can be requested by creating a *HeartbeatControl* MOI contained by the previously created NtfSubscriptionControl MOI. For details about HeartbeatControl MOI see TS 28.622 [5].

## 7.4 Unsubscription of Alarm notifications

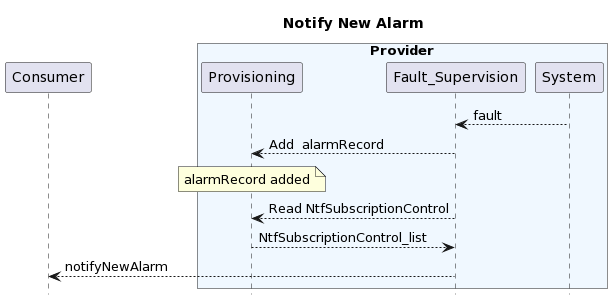
When notifications are no longer needed the consumer shall unsubscribe. This is accomplished by deleting the *NtfSubscriptionControl* MOI created during subscription using the *deleteMOI* operation.

## 7.5 Report Alarm

### 7.5.1 Notify New Alarm

When a new alarm is raised and if the *AlarmList.administrativeState*=UNLOCKED, a the MnS provider will

* Add a new attribute element to the *AlarmList.alarmRecords* attribute, this attribute element will be referred to as the alarmRecord in this document. alarmRecord represents the new alarm. A system may support multiple *AlarmLists*. The relevant *AlarmList* instance can be found by finding the ancestor MOI of type ManagedElement and/or Subnetwork containing the alarming resource. The *AlarmList* will be contained under this ancestor MOI.
* Send a *notifyNewAlarm* notification to each MnS consumer subscribed (see clause 7.2). The notification will carry the same data as stored in the new attribute element to each subscribed consumer. Sending of the notification is dependent on the attributes set in the *NtfSubscriptionControl* MOI e.g., *notificationFilter*.



### 7.5.2 Notify Changed Alarm

This clause describes use-cases "Notify changed alarm", "Notify cleared alarm", " Notify acknowledgement state change of alarm".

When an attribute field of the *alarmRecord* has changed and if the *AlarmList.administrativeState*=UNLOCKED, a the MnS provider will

* Update the alarmRecord for the alarm accordingly. This might involve deleting the alarmRecord.
* Send a notification to each MnS consumer subscribed (see clause 7.2). The notification will carry the data as updated in the alarmRecord to each subscribed consumer. Sending of the notification is dependent on the attributes set in the *NtfSubscriptionControl* MOI e.g., *notificationFilter*. Depending on which field changed different notifications will be sent. If the changed field is
  + *perceivedSeverity* theneither *notifyChangedAlarm*, *notifyClearedAlarm* or *notifyChangedAlarmGeneral* is sent. *notifyChangedAlarm* may only be sent if the new value of *perceivedSeverity* is not CLEARED. *notifyClearedAlarm* may be sent only if the new value of *perceivedSeverity* is CLEARED.
  + *correlatedNotifications* then *notifyCorrelatedNotificationChanged* will be sent.
  + *ackState* then *notifyAckStateChanged* will be sent.
  + *commentUserId, commentSystemId, commentText, commentTime* then *notifyComments* will be sent.
  + *backedUpStatus, backUpObject, trendIndication, thresholdInfo, stateChangeDefinition, monitoredAttributes, proposedRepairActions, additionalText, additionalInformation, serviceUser, serviceProvider or securityAlarmDetector* then *notifyChangedAlarmGeneral* will be sent.

## 7.6 Get Alarm List

The MnS consumer might read the list of alarms by retrieving data from the *AlarmList* MOI using the *getMOIAttributes* operation. *getMOIAttributes* allows scoping and filtering to retrieve only a subset of *alarmRecords*.

## 7.7 Notify alarm list potentially faulty

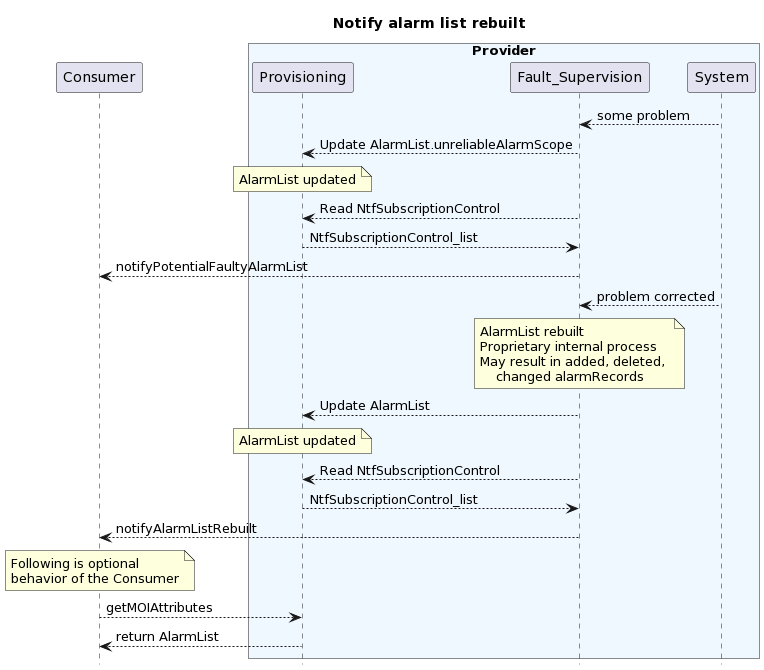
When the MnS producer detects that the AlarmList or a part of it becomes potentially unreliable the producer will

* Update AlarmList.unreliableAlarmScope to indicate the part of the AlarmList’s scope that may not be reliable.
* Send the *notifyPotentialFaultyAlarmList* notification to the consumer. The notification is usually followed by the notification *notifyPotentialFaultyAlarmList*.

## 7.8 Notify Alarm List Rebuilt

If the MnS producer rebuilds the AlarmList (completely or partially) it will

* Update the AlarmList.alarmRecords attribute accordingly. This might involve adding, deleting or updating one or more alarmRecords. Individual notifications about new or changed alarmRecords will not be sent in this case.
* Send a notifyAlarmListRebuilt notification to each MnS consumer subscribed (see clause 7.2). Sending of the notification is dependent on the attributes set in the *NtfSubscriptionControl* MOI e.g., notificationFilter. The consumer is expected to read the AlarmList as described in clause 7.6.



## 7.9 Acknowledge Alarms

The MnS consumer may support acknowledging alarms by the MnS consumer.

If the functionality is supported, the MnS consumer may acknowledge an alarm by using the *modifyMOIAttributes* operation to set the attribute fields *alarmRecords*.*ackUserId.,* *alarmRecords.ackSystemId,* *alarmRecords.ackState.*

Acknowledging an alarm may result in its removal from the AlarmList.

## 7.10 Clear Alarms

If the MnS provider supports any ADMC alarms, it is possible for the MnS consumer to clear these alarms. ADAC alarms cannot be cleared by the consumer.

If the functionality is supported, the MnS consumer may clear an alarm by using the *modifyMOIAttributes* operation to set the attribute fields *alarmRecords*.clear*UserId.,* *alarmRecords.clearSystemId.*

Clearing an alarm may result in its removal from the AlarmList.

## 7.11 Alarm loss detection

The present document does not specify methods for a management service consumer to detect alarm loss. The use of alarmId to detect alarm loss is an arrangement made between management service producer and management service consumer. The use of such arrangement is outside the scope of the present document. For example, management service producer may use integer sequence (e.g., 1, 2, 3, 4, 5, …) as alarmId instances for its alarms. Based on this knowledge, the management service consumer can detect alarm loss.

The present document does not specify how a management service producer can determine if management service consumer has received alarms correctly.

The present document does not specify methods for a management consumer and a management service producer to recover alarm loss. The only mechanism recommended to deal with alarm loss is the use of getAlarmList operation. The present document does not specify conditions under which the management service consumer should invoke this operation.

*notifyHeartbeat* notifications can be used to detect the loss of connection between the producer and the consumer, however, they do not guarantee that all alarms are sent and received correctly.

## 7.12 Virtualized resource alarm correlation

The authorized MnS consumer request VNF application alarms and VNF instance alarms related to virtualized resource from MnS producer by consuming the fault supervision MnS. MnS producer obtain the VNF application alarms (including affected VNF/VNFC instance identifier and detailed VNF alarm information) from the VNF, and the VNF instance alarms related to virtualized resource (including affected VNF/VNFC instance identifier and detailed NFVI/VM alarm information) received from the VNFM. Based on the above obtained alarms, MnS producer can correlate the VNF application alarms and the VNF and/or VNFC alarms related to virtualized resource according to affected VNF instance identifier and/or VNFC instance identifiers. The detailed procedure for virtualized resource alarm correlation procedure see corresponding procedure of NE alarm correlation made by EM in the context of NFV described in TS 28.516[4] and the MnS producer act as the role of EM.

The authorized MnS consumer requests alarms including alarm reports related to virtualized resource by consuming the fault supervision MnS. The MnS producer should collect alarm report related to virtualized resource according to affected VNF instance identifier.

8 Model

8.1 Imported information entities and local labels

|  |  |
| --- | --- |
| **Label reference** | **Local label** |
| 3GPP TS 28.532 [27], notification, notifyMOICreation | notifyMOICreation |
| 3GPP TS 28.532 [27], notification, notifyMOIDeletion | notifyMOIDeletion |
| 3GPP TS 28.532 [27], notification, notifyMOIAttributeValueChanges | notifyMOIAttributeValueChanges |
| 3GPP TS 28.532 [27], notification, notifyMOIChanges | notifyMOIChanges |
| 3GPP TS 28.622 [?], IOC, Top | Top |
| 3GPP TS 28.622 [?], IOC, ManagedElement | ManagedElement |
| 3GPP TS 28.622 [?], IOC, SubNetwork | SubNetwork |
| 3GPP TS 28.622 [?], IOC, NtfSubscriptionControl | NtfSubscriptionControl |
| 3GPP TS 28.622 [?], IOC, HeartbeatControl | HeartbeatControl |
| 3GPP TS 28.622 [?], IOC, ThresholdInfo | ThresholdInfo |

8.2 Class diagrams

8.2.1 Relationships

This clause depicts the set of classes (e.g. IOCs) implemented by Fault Supervision. This clause provides the overview of the relationships of relevant classes in UML. Subsequent clauses provide more detailed specification of various aspects of these classes.

****

**Figure 8.2.1-6: FM control NRM fragment**

8.2.2 Inheritance

This clause depicts the inheritance relationships.

****

**Figure 8.2.2-5: FM control NRM fragment**

## 8.3 Class definitions

8.3.1 AlarmRecord <<dataType>>

8.3.1.1 Definition

An AlarmRecord contains alarm information of an alarmed object instance. A new record is created in the alarm list when an alarmed object instance generates an alarm and no alarm record exists with the same values for objectInstance, alarmType, probableCause and specificProblem. When a new record is created the MnS producer creates an alarmId, that unambiguously identifies an alarm record in the AlarmList.

Alarm records are maintained only for active alarms. Inactive alarms are automatically deleted by the MnS producer from the AlarmList. Active alarms are alarms whose

a) perceivedSeverity is not "CLEARED", or whose

b) perceivedSeverity is "CLEARED" and its ackState is not "ACKNOWLEDED".

8.3.1.2 Attributes

The attributes are defined in clause 11.2.2.1.5.1 of TS 28.532 [27]. Many of them are based on definitions in ITU-T X.733 [31].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute name** | **S** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| alarmId | M | T | F | T | F |
| objectInstance | M | T | F | T | F |
| notificationId | M | T | F | T | F |
| alarmRaisedTime | M | T | F | F | F (note 5) |
| alarmChangedTime | O | T | F | F | F (note 6) |
| alarmClearedTime | M | T | F | F | F (note 7) |
| alarmType | M | T | F | T | F |
| probableCause | M | T | F | T | F |
| specificProblem | O | T | F | T | F |
| perceivedSeverity | M | T | T (note 4) | F | F(note 6) |
| backedUpStatus | O | T | F | F | F |
| backUpObject | O | T | F | F | F |
| trendIndication | O | T | F | F | F |
| thresholdInfo | O | T | F | F | F |
| stateChangeDefinition | O | T | F | F | F |
| monitoredAttributes | O | T | F | F | F |
| proposedRepairActions | O | T | F | F | F |
| additionalText | O | T | F | F | F |
| additionalInformation | O (see note 3) | T | F | F | F |
| rootCauseIndicator | O | T | F | F | F |
| ackTime | M | T | F | F | F |
| ackUserId | M | T | T (see note 8) | F | F |
| ackSystemId | O | T | T (see note 8) | F | F |
| ackState | M | T | T (see note 8) | F | F |
| clearUserId | O (see note 1) | T | T | F | F |
| clearSystemId | O (see note 1) | T | T | F | F |
| serviceUser | O (see note 2) | T | F | F | F |
| serviceProvider | O (see note 2) | T | F | F | F |
| securityAlarmDetector | O (see note 2) | T | F | F | F |
| comments | O | T | T | F | F |
| correlatedNotifications | O | T | F | F | F |
| NOTE 1: These attributes and qualifiers are applicable only if producer supports consumer to set perceivedSeverity to CLEARED.  NOTE 2: These attributes are supported if the producer emits notifyNewAlarm that carries security alarm information.  NOTE 3: This attribute is supported to carry vendor specific information.  NOTE 4: This isWritable property is True only if producer supports consumer to set perceivedSeverity to CLEARED  NOTE 5: Emit notifyNewAlarm.  NOTE 6: Emit notifyChangedAlarm  NOTE 7: Emit notifyClearedAlarm  NOTE 8: This isWritable property is True only if producer supports the consumer to acknowledge alarms. | | | | | |

8.3.1.3 Attribute constraints

None.

8.3.1.4 Notifications

See clause 8.5.

8.3.2 AlarmList

8.3.2.1 Definition

The AlarmList represents the capability to store and manage alarm records. It can be name-contained by SubNetwork and ManagedElement. The management scope of an AlarmList is defined by all descendant objects of the base managed object, which is the object name-containing the AlarmList, and the base object itself.

AlarmList instances are created by the system or are pre-installed. They cannot be created nor deleted by MnS consumers.

An instance of SubNetwork or ManagedElement has at most one name-contained instance of AlarmList.

When the alarm list is locked or disabled, the existing alarm records are not updated or deleted, and new alarm records are not added to the alarm list.

8.3.2.2 Attributes

The AlarmList IOC includes attributes inherited from Top IOC (defined in clause 8.3.29) and the following attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **S** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| administrativeState | O | T | T | F | T |
| operationalState | M | T | F | F | T |
| numOfAlarmRecords | M | T | F | F | F |
| lastModification | M | T | F | F | F |
| alarmRecords | M | T | T | F | F |
| warningCount | O | T | F | F | F |
| minorCount | O | T | F | F | F |
| majorCount | O | T | F | F | F |
| criticalCount | O | T | F | F | F |
| indeterminateCount | O | T | F | F | F |
| clearedCount | O | T | F | F | F |
| unreliableAlarmScope | O | T | F | F | F |

8.3.2.3 Attribute constraints

None

8.3.2.4 Notifications

The common notifications defined in clause 8.5 are valid for this IOC, without exceptions or additions.

8.3.4 AlarmComment <<dataType>>

8.3.4.1 Definition

This data type represents a comment on alarm.

8.3.4.2 Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **S** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| commentTime | M | T | F | F | F |
| commentUserId | M | T | T | F | F |
| commentSystemId | O | T | F | F | F |
| commentText | M | T | T | F | F |

8.3.4.3 Attribute constraints

None

8.3.4.4 Notifications

See clause 8.5.

8.3.5 CorrelatedNotification <<dataType>>

8.3.5.1 Definition

The sourceObjectInstance attribute of CorrelatedNotification identifies one MonitoredEntity. For the MonitoredEntity identified, a set of notification identifiers is also identified. One or more CorrelatedNotification instances can be included in an AlarmRecord. In this case, the information of the AlarmRecord is said to be correlated to information carried in the notifications identified by the CorrelatedNotification instances. See further definition of correlated notification in ITU-T Recommendation X.733 [8], clause 8.1.2.9.

The notification identified by the CorrelatedNotification, as defined in ITU-T and used here, can carry all types of information and is not restricted to carrying alarm information only. For example, a notification, identified by the CorrelatedNotification, can indicate a managed instance attribute value change. In this case, the information of the AlarmRecord is said to be correlated to the managed instance attribute value change event.

If a CorrelatedNotification references an alarm (e.g., by referencing the notificationId of a notifyNewAlarm notification), the alarmRecord for that alarm may or may not exist in the AlarmList. For example, the alarm may have been acknowledged and cleared and therefore, removed from the AlarmList.

8.3.5.2 Attributes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **S** | **isReadable** | **isWritable** | **isInvariant** | **isNotifyable** |
| sourceObjectInstance | M | T | F | F | F |
| notificationIdSet | M | T | F | F | F |

8.3.5.3 Attribute constraints

None

8.3.5.4 Notifications

See clause 8.5.

8.4 Attribute definitions

8.4.1 Attribute properties

The following table defines the properties of attributes specified in the present document.

| **Attribute Name** | **Documentation and Allowed Values** | **Properties** |
| --- | --- | --- |
| objectClass | Class of a managed object instance. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| objectInstance | Managed object instance identified by its DN. | type: String  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| systemDN | Distinguished Name (DN) of a IRPAgent or a MnSAgent. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| administrativeState | Administrative state of a managed object instance. The administrative state describes the permission to use or prohibition against using the object instance. The adminstrative state is set by the MnS consumer.  allowedValues: LOCKED, UNLOCKED. | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: LOCKED  isNullable: False |
| operationalState | Operational state of manged object instance. The operational state describes if an object instance is operable ("ENABLED") or inoperable ("DISABLED"). This state is set by the object instance or the MnS producer and is hence READ-ONLY.  allowedValues: ENABLED, DISABLED. | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: DISABLED  isNullable: False |
| alarmRecords | List of alarm records | type: AlarmRecord  multiplicity: \*  isOrdered: False  isUnique: True  default value: None  isNullable: False |
| numOfAlarmRecords | Number of alarm records in the AlarmList.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| lastModification | Time an alarm record was modified the last time | type: DateTime  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| warningCount | Number of alarm records in the AlarmList with perceivedSeverity of WARNING.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| minorCount | Number of alarm records in the AlarmList with perceivedSeverity of MINOR.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| majorCount | Number of alarm records in the AlarmList with perceivedSeverity of MAJOR.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| criticalCount | Number of alarm records in the AlarmList with perceivedSeverity of CRITICAL.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| indeterminateCount | Number of alarm records in the AlarmList with perceivedSeverity of INDETERMINATE.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| clearedCount | Number of alarm records in the AlarmList with perceivedSeverity of CLEARED.  allowedValues: Non-negative numbers. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False |
| unreliableAlarmScope | Identifies, the part of the alarm scope that may not be reliable.  If this parameter is equal to the instance carried in systemDN, then all AlarmRecord instances in the AlarmList may not be reliable.  If this parameter is equal to some instance represented by MonitoredEntity, then only AlarmRecord related to this instance and its descendants may not be reliable. | type: DN  multiplicity: 0..\*  isOrdered: False  isUnique: True  defaultValue: None  isNullable: False |
| alarmId | Identifies an AlarmRecord in the AlarmList. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| notificationId | The Id of the last notification updating the AlarmRecord. | type: integer  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| alarmRaisedTime | Date and time the alarm was raised. | type: DateTime  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| alarmChangedTime | It indicates the last date and time when the AlarmInformation is changed by the alarmed resource. Changes to AlarmInformation caused by invocations of the management service consumer would not change this date and time. | type: DateTime  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| alarmClearedTime | Date and time the alarm was cleared. | type: DateTime  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| alarmType | It indicates the type of alarm.  Communications Alarm:  An alarm of this type is associated with the procedure and/or process required conveying information from one point to another (ITU-T Recommendation X.733 [8]).  Processing Error Alarm:  An alarm of this type is associated with a software or processing fault (ITU T Recommendation X.733 [8]).  Environmental Alarm:  An alarm of this type is associated with a condition related to an enclosure in which the equipment resides (ITU-T Recommendation X.733 [8]).  Quality of Service Alarm:  An alarm of this type is associated with degradation in the quality of a service (ITU T Recommendation X.733 [8]).  Equipment Alarm:  An alarm of this type is associated with an equipment fault (ITU-T Recommendation X.733 [8]).  Integrity Violation:  An indication that information may have been illegally modified, inserted or deleted.  Operational Violation:  An indication that the provision of the requested service was not possible due to the unavailability, malfunction or incorrect invocation of the service.  Physical Violation:  An indication that a physical resource has been violated in a way that suggests a security attack.  Security Service or Mechanism Violation:  An indication that a security attack has been detected by a security service or mechanism.  Time Domain Violation: An indication that an event has occurred at an unexpected or prohibited time.  Allow values:  COMMUNICATIONS\_ALARM,  QUALITY\_OF\_SERVICE\_ALARM,  PROCESSING\_ERROR\_ALARM,  EQUIPMENT\_ALARM,  ENVIRONMENTAL\_ALARM,  INTEGRITY\_VIOLATION,  OPERATIONAL\_VIOLATION,  PHYSICAL\_VIOLATION, SECURITY\_SERVICE\_OR\_MECHANISM\_VIOLATION, TIME\_DOMAIN\_VIOLATION | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| probableCause | It qualifies alarm and provides further information than alarmType. Probable causes are ouside the scope of the present document. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| specificProblem | It provides further refinement to the probableCause. This attribute value shall be single-valued and of simple type such as integer or string. See definition in ITU-T Recommendation X.733 [8] clause 8.1.2.2. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| perceivedSeverity | It indicates the relative level of urgency for operator attention.  allowedValues:  CRITICAL,  MAJOR,  MINOR,  WARNING,  INDETERMINATE,  CLEARED | type: ENUM  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| backedUpStatus | It indicates if an object (the MonitoredEntity) has a back up. See definition in ITU-T Recommendation X.733 [8] clause 8.1.2.4. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| backUpObject | TODO | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| trendIndication | It indicates if some observed condition is getting better, worse, or not changing. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| thresholdInfo | It indicates the crossed threshold information such as:  - The identifier of the monitored attribute whose value has crossed a threshold,  - The threshold settings,  - The observed value that have crossed a threshold, etc.  See definition in ITU-T Recommendation X.733 [8] clause 8.1.2.7. See also for information in TS 32.401 [19] clause 5.6. | type: ThresholdInfo  multiplicity: \*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| stateChangeDefinition | It indicates attribute value changes associated with the alarm for state attributes of the monitored entity (state transitions). The change is reported with the name of the state attribute, the new value and an optional old value. See definition in ITU-T Recommendation X.733 [8] clause 8.1.2.11. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| monitoredAttributes | It indicates attributes of the monitored entity and their values at the time the alarm occurred that are of interest for the alarm report. How these attributes are chosen is outside of the scope of the present document. See definition in ITU-T Recommendation X.733 [8] clause 8.1.2.11. | type: string  multiplicity: \*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| proposedRepairActions | Used if the cause is known and the system being managed can suggest one or more solutions to fix the problem causing the alarm as defined in ITU-T Rec. X. 733 [8] | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| additionalText | Allows a free form text description to be reported as defined in ITU-T Rec. X. 733 [8]. | type: string  multiplicity: \*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| additionalInformation | This attribute when present allows the inclusion of a set of vendor specific alarm information in the alarm.  A specific condition for this optional population is when an alarm presented by the Management System (e.g. via the user interface) has different values of perceived severity, and / or alarm type, compared with the values presented to the Itf-N.  Any other uses of additional information on the alarm and its semantics is outside the scope of the present document | type: string  multiplicity: \*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| rootCauseIndicator | It indicates that this AlarmInformation is the root cause of the events captured by the notifications whose identifiers are in the related CorrelatedNotification instances. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| ackTime | It identifies the time when the alarm has been acknowledged or unacknowledged the last time, i.e. it registers the time when ackState changes. | type: DateTime  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| ackUserId | It identifies the last user who has changed the acknowledgement state. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| ackSystemId | It identifies the system that last changed the ackState of an alarm, i.e. acknowledged or unacknowledged the alarm. | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| ackState | It identifies the acknowledgement state of an alarm. | type: boolean  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| clearUserId | It carries the identity of the user who invokes the clearAlarms operation. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| clearSystemId | It carries the identity of the system in consuming the fault management service. That management service consumer supports the user who invokes the clearAlarms(). | type: DN  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| serviceUser | It identifies the service-user whose request for service provided by the serviceProvider led to the generation of the security alarm. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| serviceProvider | It identifies the service-provider whose service is requested by the serviceUser and the service request provokes the generation of the security alarm. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| securityAlarmDetector | It carries the identity of the detector of the security alarm. | type: string  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| comments | List of comments and data about the comments. | type: AlarmContent  multiplicity: \*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| correlatedNotifications | List of correlated notifications. | type: CorrelatedNotification  multiplicity: \*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| commentTime | Date and Time the comment was last updated. | type: DateTime  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| commentUserId | It carries the identification of the user who made the comment. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| commentSystemId | It carries the identification of the system (Management System) from which the comment is made. That system supports the user that made the comment. | type: DN  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| commentText | It carries the textual comment. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| sourceObjectInstance | It identifies one MonitoredEntity. | type: string  multiplicity: 1  isOrdered: N/A  isUnique: N/A defaultValue: None  isNullable: False |
| notificationIdSet | Notification identifier | type: integer  multiplicity: 1..\*  isOrdered: False  isUnique: True defaultValue: None  isNullable: False |
| NOTE 1: xxxxx. | | |

8.4.2 Constraints

None

8.5 Common notifications

8.5.1 Alarm notifications

This clause presents a list of notifications, defined in clause 9, that a MnS consumer can receive. The notification header attribute objectClass/objectInstance, defined in [3], captures the DN of an instance of an IOC defined in the present document.

| **Name** | **S** | **Notes** |
| --- | --- | --- |
| notifyNewAlarm | M |  |
| notifyClearedAlarm | M |  |
| notifyChangedAlarm | O |  |
| notifyChangedAlarmGeneral | O |  |
| notifyCorrelatedNotificationChanged | O |  |
| notifyAckStateChanged | O |  |
| notifyComments | O |  |
| notifyPotentialFaultyAlarmList | O |  |
| notifyAlarmListRebuilt | M |  |

8.5.2 Configuration notifications

This clause presents a list of notifications, defined in [27], that a MnS consumer can receive. The notification header attribute objectClass/objectInstance, defined in [3], captures the DN of an instance of an IOC defined in the present document.

| **Name** | **S** | **Notes** |
| --- | --- | --- |
| notifyMOIObjectCreation | O |  |
| notifyMOIObjectDeletion | O |  |
| notifyMOIAttributeValueChanges | O |  |
| notifyMOIChanges | O |  |

# 9 Operations and notifications

## 9.1 Overview

Clause 8.5.1 defines a list of notifications defined and implemented by Fault Supervision. The notification header attribute objectClass/objectInstance captures the DN of an instance of an IOC defined in the present document.

## 9.2 Common Notification Header

The following parameters are part of all Fault Supervision notifications. Additional fields are defined for each notification in other chapters. Some notifications may redefine the semantics of individual parameters, but unless specified in the relevan clauses the semantics in the following table is valid.

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| objectClass | M | String  ClassName of the object identified by objectInstance. |  |
| objectInstance | M | alarmRecord.objectInstance  DN of the MonitoredEntity that is the source of the alarm |  |
| notificationId | M | This is an identifier for the notification, which may be used to correlate notifications. | The identifier of the notification shall be chosen to be unique across all notifications of a particular managed object instance throughout the time that correlation is significant, it uniquely identifies the notification from other notifications generated by the subject MOI. |
| notificationType | M | The type of the notification | A fixed value different for each notification type. |
| eventTime | M | DateTime |  |
| systemDN | M | It shall carry the DN of management service providers. |  |

## 9.3 notifyNewAlarm

### 9.3.1 Definition

This notification is generated by the MnS producer when a new alarm is raised AlarmRecord and is added to the AlarmList. The notification parameters depend on the alarmType and are different for non-security and security alarms.

### 9.3.2 Input parameters

The notifyNewAlarm notification is defined by Table 9.3.2-1.

If the alarmType is "Communications Alarm", "Processing Error Alarm", "Environmental Alarm". "Quality Of Service Alarm" or "Equipment Alarm" the alarm is considered to be non-security related. If the alarmType is "Integrity Violation", "Operational Violation", "Physical Violation", "Security Service or Mechanism Violation" or "Time Domain Violation" the alarm is considered to be security related.

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Description** |
| --- | --- | --- | --- |
| notificationType | M | "notifyNewAlarm" |  |
| eventTime | M | alarmRecord.alarmRaisedTime |  |
| alarmId | M | alarmRecord.alarmId |  |
| alarmType | M | alarmRecord.alarmType |  |
| probableCause | M | alarmRecord.probableCause |  |
| perceivedSeverity | M | alarmRecord.perceivedSeverity |  |
| specificProblem | CO | alarmRecord.specificProblem | Used only in non-security notifications. |
| backedUpStatus | O | alarmRecord.backedUpStatus | Used only in non-security notifications. |
| backUpObject | O | alarmRecord.backUpObject | Used only in non-security notifications. |
| trendIndication | O | alarmRecord.trendIndication | Used only in non-security notifications. |
| thresholdInfo | O | alarmRecord.thresholdInfo | Used only in non-security notifications. |
| correlatedNotifications | O | alarmRecord.correlatedNotifications |  |
| stateChangeDefinition | O | alarmRecord.stateChangeDefinition | Used only in non-security notifications. |
| monitoredAttributes | O | alarmRecord.monitoredAttributes | Used only in non-security notifications. |
| proposedRepairActions | O | alarmRecord.proposedRepairActions | Used only in non-security notifications. |
| additionalText | O | alarmRecord.additionalText |  |
| additionalInformation | O | alarmRecord.additionalInformation |  |
| rootCauseIndicator | O | alarmRecord.rootCauseIndicator |  |
| serviceUser | C | alarmRecord.securityServiceUser | Used only in security notifications.  This may contain no information if the identify of the service-user (requesting the service) is not known. |
| serviceProvider | C | alarmRecord.securityServiceProvider | Used only in security notifications.  This shall always identify the service-provider receiving a service request, from serviceUser, that provokes the security alarm. |
| securityAlarmDetector | C | alarmRecord.securityAlarmDetector | Used only in security notifications.  This may contain no information if the detector of the security alarm is the serviceProvider. |

### 9.3.3 Triggering event

#### 9.3.3.1 From-state

noMatchedAlarm.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| noMatchedAlarm | A new alarm is raised.  AlarmList does not contain an alarmRecord that has the following properties:  Its matching-criteria-attributes values are identical to that of the newly generated network alarm and it is involved in relation-AlarmObject-AlarmRecord with the same MonitoredEntity as the one identified by the newly generated network alarm. |

#### 9.3.3.2 To-state

newAlarmInAlarmList.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| newAlarmInAlarmList | AlarmList contains an alarmRecord holding information conveyed by the newly generated network alarm. This AlarmRecord is involved in relation-AlarmObject-AlarmRecord with the same MonitoredEntity as the one identified by the newly generated network alarm.  The information in the notification should be included in the new alarmRecord. |

## 9.4 notifyClearedAlarm

### 9.4.1 Definition

This notification is generated by the MnS producer when the perceivedSeverity of an existing AlarmRecord changes to "CLEARED".

### 9.4.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| notificationType | M | "notifyClearedAlarm" |  |
| eventTime | M | alarmRecord.alarmClearedTime |  |
| alarmId | M | alarmRecord.alarmId |  |
| alarmType | M | alarmRecord.alarmType |  |
| probableCause | M | alarmRecord.probablaCause |  |
| perceivedSeverity | M | alarmRecord.perceivedSeverity | Value shall be "CLEARED" |
| correlatedNotifications | O | alarmRecord.correlatedNotifications |  |
| clearUserId | O | alarmRecord.clearUserId | This parameter shall be present and contain valid information if the AlarmRecord is cleared by the consumer. |
| clearSystemId | O | alarmRecord.clearSystemId | This parameter shall be present if clearUserId is present and if alarmRecord.clearSystemId contains valid information. |

### 9.4.3 Triggering event

#### 9.4.3.1 From-state

alarmMatchedAndCleared OR clearedByConsumer.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmMatchedAndCleared | The matching-criteria-attributes of the newly generated network alarm have values that are identical (matched) with ones in one alarmRecord in AlarmList and the perceivedSeverity of the matched AlarmRecord is not Cleared  AND  The perceivedSeverity of the newly generated network alarm is cleared. |
| clearedByConsumer | The consumer set the severity of the matching alarmRecord.percievedSeverity to “CLEARED” |

#### 9.4.3.2 To-state

AlarmRecordCleared\_1 OR AlarmRecordCleared\_2.

Note the alarmRecord may be removed from the AlarmList.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| AlarmRecordCleared\_1 | Case if From-state is alarmMatchedAndCleared:  The following attributes of the subject alarmRecord are updated:  notificationId, perceivedSeverity (updated to Cleared), clearedTime. |
| AlarmRecordCleared\_2 | Case if From-state is clearedByConsumer:  The following attributes of the subject alarmRecord are updated:  notificationId, clearedTime, perceivedSeverity (updated to CLEARED), clearedUserId, clearedSystemId. |

## 9.5 notifyAlarmListRebuilt

### 9.5.1 Definition

This notification is generated by the MnS producer when the AlarmList has been completely or partially rebuilt.

### 9.5.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| objectClass | M | String | ClassName of the object identified by objectInstance. |
| objectInstance | M | DN | Identifies the part of the alarm scope that has been rebuilt.  If this parameter is equal to the instance carried in systemDN, then all AlarmRecord instances in the AlarmList may have been rebuilt.  If this parameter is equal to some other instance, then only alarmRecords related to this instance and its descendants may have been rebuilt. |
| notificationType | M | "notifyAlarmListRebuilt" |  |
| eventTime | M | DateTime | The time when the alarm list has been rebuilt. |
| reason | M | String  "System-NE communication error", "System restarts", "indeterminate". Other values can be added. | The reason why the system has rebuilt the AlarmList. This may carry different reasons than that carried by the immediate previous notifyPotentialFaultyAlarmList. |
| alarmListAlignmentRequirement | O | "alignmentRequired", "alignmentNotRequired". | It carries an enumeration of "alignmentRequired" and "alignmentNotRequired". |

### 9.5.3 Triggering event

#### 9.5.3.1 From-state

alarmListRebuilt\_0 OR alarmListRebuilt\_1.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmListRebuilt\_0 | MnS producer has cold-started, initialized, re-initialized or rebooted and it has initiated procedure to rebuild its AlarmList. |
| alarmListRebuilt\_1 | MnS producer loses confidence in part or whole of its AlarmList. MnS producer has initiated procedure to repair its AlarmList. |

#### 9.5.3.2 To-state

alarmListRebuilt\_2.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmListRebuilt\_2 | MnS producer rebuilds the whole or part of AlarmList. |

## 9.6 notifyChangedAlarm

### 9.6.1 Definition

This notification is generated by the MnS producer when the perceivedSeverity of an existing AlarmRecord changes (except to the value "CLEARED").

The notification is **deprecated**, use notifyChangedAlarmGeneral instead.

### 9.6.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| notificationType | M | "notifyChangedAlarm" |  |
| eventTime | M | alarmRecord.alarmChangedTime |  |
| alarmId | M | alarmRecord.alarmId |  |
| alarmType | M | alarmRecord.alarmType |  |
| probableCause | M | alarmRecord.probableCause |  |
| perceivedSeverity | M | alarmRecord.perceivedSeverity |  |

### 9.6.3 Triggering event

#### 9.6.3.1 From-state

alarmMatched AND alarmNotCleared AND alarmChanged.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmMatched | The matching-criteria-attributes of the newly generated network alarm has values that are identical (matches) with ones in one alarmRecord in AlarmList. |
| alarmNotCleared | The perceivedSeverity of the newly generated network alarm is not Cleared. |
| alarmChanged | The perceivedSeverity of the newly generated network alarm and of the matched AlarmRecord are different. |

#### 9.6.3.2 To-state

informationUpdate.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| informationUpdate | The alarmRecord identified in alarmMatched in from-state has been updated according to the following rules:  - notificationId is updated;  - alarmChangedTime is updated;  - perceivedSeverity is updated;  - ackTime, ackUserId and ackSystemId are updated to contain no information;  - ackState is updated to "unacknowledged"; |

## 9.7 notifyChangedAlarmGeneral

##### 9.7.1 Definition

This notification is generated by the MnS producer when one or more of the following attributes of an AlarmRecord instance in the AlarmList changes its value: perceivedSeverity, backedUpStatus, backUpObject, trendIndication, thresholdInfo, stateChangeDefinition, monitoredAttributes, proposedRepairActions, additionalText, additionalInformation, serviceUser, serviceProvider or securityAlarmDetector. From the attributes listed above, only those that changed value shall be included in the notification.

The notification parameters depend on the alarmType and are different for non-security and security alarms. If the alarmType is "Communications Alarm", "Processing Error Alarm", "Environmental Alarm". "Quality Of Service Alarm" or "Equipment Alarm" the alarm is considered to be non-security related. If the alarmType is "Integrity Violation", "Operational Violation", "Physical Violation", "Security Service or Mechanism Violation" or "Time Domain Violation" the alarm is considered to be security related.

##### 9.7.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| notificationType | M | "notifyChangedAlarmGeneral" |  |
| eventTime | M | alarmRecord.alarmChangedTime |  |
| alarmId | M | alarmRecord.alarmId |  |
| alarmType | M | alarmRecord.alarmType |  |
| probableCause | O | alarmRecord.probableCause |  |
| specificProblem | O | alarmRecord.specificProblem |  |
| perceivedSeverity | O | alarmRecord.perceivedSeverity |  |
| backedUpStatus | O | alarmRecord.backedUpStatus |  |
| backUpObject | O | alarmRecord.backUpObject | Used only in non-security notifications. |
| trendIndication | O | alarmRecord.trendIndication | Used only in non-security notifications. |
| thresholdInfo | O | alarmRecord.thresholdInfo | Used only in non-security notifications. |
| correlatedNotifications | O | alarmRecord.correlatedNotifications |  |
| stateChangeDefinition | O | alarmRecord.stateChange | Used only in non-security notifications. |
| monitoredAttributes | O | alarmRecord.monitoredAttributes | Used only in non-security notifications. |
| proposedRepairActions | O | alarmRecord.proposedRepairActions | Used only in non-security notifications. |
| additionalText | O | alarmRecord.additionalText |  |
| additionalInformation | O | alarmRecord.additionalInformation |  |
| rootCauseIndicator | O | alarmRecord.rootCauseIndicator |  |
| serviceUser | C | alarmRecord.securityServiceUser | Used only in security notifications.  This may contain no information if the identify of the service-user (requesting the service) is not known. |
| serviceProvider | C | alarmRecord.securityServiceProvider | Used only in security notifications.  This shall always identify the service-provider receiving a service request, from serviceUser, that provokes the security alarm. |
| securityAlarmDetector | C | alarmRecord.securityAlarmDetector | Used only in security notifications.  This may contain no information if the detector of the security alarm is the serviceProvider. |
| changedAlarmAttributes | O | LIST OF SEQUENCE <AttributeName, OldAttributeValue> | The changed alarm attributes (name/value pairs) (with old values). |

##### 9.7.3 Trigger event

###### 9.7.3.1 From-state

alarmMatched AND alarmNotCleared AND alarmChanged.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmMatched | The matching-criteria-attributes of the newly generated network alarm has values that are identical (matches) with ones in one alarmRecord in AlarmList. |
| alarmChanged | One or more of perceivedSeverity, backedUpStatus, backUpObject, trendIndication, thresholdInfo, stateChangeDefinition, monitoredAttributes, proposedRepairActions, additionalText, additionalInformation, serviceUser, serviceProvider or securityAlarmDetector of the newly generated network alarm and of the matched AlarmRecord are different. |

###### 9.7.3.2 To-state

informationUpdate.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| informationUpdate | The AlarmRecord identified in alarmMatched in from-state has been updated according to the following rules: perceivedSeverity, backedUpStatus, backUpObject, trendIndication, thresholdInfo, stateChangeDefinition, monitoredAttributes, proposedRepairActions, additionalText, additionalInformation, serviceUser, serviceProvider or securityAlarmDetector is updated;  notificationId is updated;  alarmChangedTime is updated;  ackTime, ackUserId and ackSystemId are updated to contain no information;  ackState is updated to "unacknowledged"; |

## 9.8 notifyCorrelatedNotificationChanged

##### 9.8.1 Definition

This notification is generated by the MnS producer when the set of correlatedNotifications is created, updated or deleted.

##### 9.8.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| notificationType | M | "notifyCorrelatedNotificationChanged" |  |
| eventTime | M | alarmRecord.alarmChanedTime  It carries the time when the CorrelatedNotification is created, updated or deleted. |  |
| alarmId | M | alarmRecord.alarmId |  |
| correlatedNotifications | M | alarmRecord.correlatedNotifications |  |
| rootCauseIndicator | O | alarmRecord.rootCauseIndicator |  |

##### 9.8.3 Triggering event

###### 9.8.3.1 From-state

newAlarmCorrelationInfoIsAvailable AND AlarmRecordExists.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| newAlarmCorrelationInfoIsAvailable | New alarm correlation information is available. |
| AlarmRecordExists | The alarmRecord is in AlarmList. |

###### 9.8.3.2 To-state

alarmCorrelatedInfoUpdated.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmCorrelatedInfoUpdated | The set of correlatedNotifications network slice instances is created, updated or deleted. |

## 9.9 notifyAckStateChanged

##### 9.9.1 Definition

This notification is generated by the MnS producer when a the acknowledgement state of an alarm changes from "UNACKNOWLEDGED" to "ACKNOWLEDGED" or back from "ACKNOWLEDGED" to "UNACKNOWLEDGED".

##### 9.9.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| notificationType | M | "notifyAckStateChanged" |  |
| eventTime | M | alarmRecord.ackTime |  |
| alarmId | M | alarmRecord.alarmId |  |
| alarmType | M | alarmRecord.alarmType |  |
| probableCause | M | alarmRecord.probableCause |  |
| perceivedSeverity | M | alarmRecord.perceivedSeverity |  |
| ackState | M | alarmRecord.ackState |  |
| ackUserId | M | alarmRecord.ackUserId |  |
| ackSystemId | O | alarmRecord.ackSystemId |  |

##### 9.9.3 Triggering event

###### 9.9.3.1 From-state

ackedByConsumer OR ackedByProvider AND AlarmRecordExists.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| ackedByConsumer | Consumer changes alarmRecord.ackState. |
| ackedByProvider | The provider changes alarmRecord.ackState. |
| AlarmRecordExists | The AlarmRecord exists in AlarmList. |

###### 9.9.3.2 To-state

alarmAckStateHasChanged.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| alarmAckStateHasChanged | The alarmRecord.ackState identified by from-state assertion AlarmRecordExists have been updated. Specifically, the following attributes of the subject AlarmRecord are updated:  -- notificationId, ackTime, ackUserId, ackState, ackSystemId. |

## 9.10 notifyComments

##### 9.10.1 Definition

This notification is generated by the MnS producer when a Comment instance is added to an AlarmRecord instance in the AlarmList.

A MnS producer shall support this notification if it supports the operation setComment.

##### 9.10.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| notificationType | M | "notifyComments" |  |
| eventTime | M | alarmRecord.alarmChangedTime | The time the comment was updated |
| alarmId | M | alarmRecord.alarmId |  |
| alarmType | M | alarmRecord.alarmType |  |
| probableCause | M | alarmRecord.probableCause |  |
| perceived Severity | M | alarmRecord.perceivedSeverity |  |
| comments | M | The Comment instances related to this AlarmRecord.  Type: AlarmComment |  |

##### 9.10.3 Trigger event

###### 9.10.3.1 From-state

commentedByConsumer OR commentedByServiceprovider AND AlarmRecordExists.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| commentedByConsumer | Consumer added an alarmRecord.comments field element. |
| commentedByServiceprovider | Provider updated alarmRecord.comments field element. |
| AlarmRecordExists | The AlarmRecord is in AlarmList. |

###### 9.10.3.2 To-state

commentInserted.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| commentInserted | One AlarmComment has been created and added to alarmRecord.comments. The following attribute fields of the newly created Comment instance shall be populated:  commentTime, commentText, commentUserId and commentSystemId. |

## 9.11 notifyPotentialFaultyAlarmList

##### 9.11.1 Definition

This notification is generated by the MnS producer when the MnS producer looses confidence in the integrity of its alarm list.

The MnS producer may then rebuilt the faulty alarm list. When the alarm List is rebuilt or confidence in the existing alarm list is re-established the MnS producer may generate a notifyAlarmListRebuilt notification.

The parameters objectClass and objectInstance are used to specify if the complete alarm list is unreliable or only parts thereof.

The MnS consumer behaviour, on reception of this notifyPotentialFaultyAlarmList notification, is not specified. The authorized consumer behaviour is considered not essential for the specification of the interface itself. However, the following are recommended actions the uthorized consumer should take, in case it receives this notification.

1) The uthorized consumer should not perform any task requiring the integrity of the AlarmRecord identified as faulty or unreliable by the subject notification.

2) The uthorized consumer should not invoke operations that require integrity of the AlarmList such as getAlarmList., acknolwedgeAlarms operations.

##### 9.11.2 Input parameters

| **Parameter Name** | **S** | **Matching Information/ Information Type / Legal Values** | **Comment** |
| --- | --- | --- | --- |
| objectInstance | M | It identifies the instance identified by systemDN or an instance of MonitoredEntity. | Identifies, together with the objetClass parameter, the part of the alarm scope that may not be reliable.  If this parameter is equal to the instance carried in systemDN, then all AlarmRecord instances in the AlarmList may not be reliable.  If this parameter is equal to some instance represented by MonitoredEntity, then only AlarmRecord related to this instance and its descendants may not be reliable. |
| notificationType | M | "notifyPotentialFaultyAlarmList" |  |
| eventTime | M | DateTime | Time when the MnS producer lost confidence in the integrity of the alarm list |
| reason | M | "serviceprovider-NE communication error", " serviceprovider restarts", "indeterminate". Other values can be added. | Reason why the MnS producer has to rebuild its AlarmList. |

##### 9.11.3 Trigger event

###### 9.11.3.1 From-state

faultyAlarmListDetected.

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| faultyAlarmListDetected | MnS producer detects faults in part or whole of its AlarmList. |

###### 9.11.3.2 To-state

faultyAlarmList

|  |  |
| --- | --- |
| **Assertion Name** | **Definition** |
| faultyAlarmList | MnS producer initiates the AlarmList rebuild process. |

# 10 Stage 3 – Solution Sets

## 10.1 YANG Definitions

### 10.1.1 NRM Definitions

#### 10.1.1 module \_3gpp-common-fm.yang

<CODE BEGINS>

module \_3gpp-common-fm {

yang-version 1.1;

namespace "urn:3gpp:sa5:\_3gpp-common-fm";

prefix "fm3gpp";

import ietf-yang-types { prefix yang; }

import \_3gpp-common-top { prefix top3gpp; }

import \_3gpp-common-yang-types { prefix types3gpp; }

import \_3gpp-common-yang-extensions { prefix yext3gpp; }

organization "3GPP SA5";

contact "https://www.3gpp.org/DynaReport/TSG-WG--S5--officials.htm?Itemid=464";

description "Defines a Fault Management model";

reference "3GPP TS 28.623

Generic Network Resource Model (NRM)

Integration Reference Point (IRP);

Solution Set (SS) definitions

3GPP TS 28.622

Generic Network Resource Model (NRM)

Integration Reference Point (IRP);

Information Service (IS)";

revision 2022-10-24 { reference CR-0196; }

revision 2021-08-08 { reference "CR-0132"; }

revision 2021-06-02 { reference "CR-0130"; }

revision 2020-06-03 { reference "CR-0091"; }

revision 2020-02-24 {

reference "S5-201365";

}

typedef eventType {

type enumeration {

enum COMMUNICATIONS\_ALARM {

value 2;

}

enum QUALITY\_OF\_SERVICE\_ALARM {

value 3;

}

enum PROCESSING\_ERROR\_ALARM {

value 4;

}

enum EQUIPMENT\_ALARM {

value 5;

}

enum ENVIRONMENTAL\_ALARM {

value 6;

}

enum INTEGRITY\_VIOLATION {

value 7;

}

enum OPERATIONAL\_VIOLATION {

value 8;

}

enum PHYSICAL\_VIOLATIONu {

value 9;

}

enum SECURITY\_SERVICE\_OR\_MECHANISM\_VIOLATION {

value 10;

}

enum TIME\_DOMAIN\_VIOLATION {

value 11;

}

}

description "General category for the alarm.";

}

typedef severity-level {

type enumeration {

enum CRITICAL { value 3; }

enum MAJOR { value 4; }

enum MINOR { value 5; }

enum WARNING { value 6; }

enum INDETERMINATE { value 7; }

enum CLEARED { value 8; }

}

description "The possible alarm serverities.

Aligned with ERICSSON-ALARM-MIB.";

}

grouping AlarmRecordGrp {

description "Contains alarm information of an alarmed object instance.

A new record is created in the alarm list when an alarmed object

instance generates an alarm and no alarm record exists with the same

values for objectInstance, alarmType, probableCause and specificProblem.

When a new record is created the MnS producer creates an alarmId, that

unambiguously identifies an alarm record in the AlarmList.

Alarm records are maintained only for active alarms. Inactive alarms are

automatically deleted by the MnS producer from the AlarmList.

Active alarms are alarms whose

a) perceivedSeverity is not CLEARED, or whose

b) perceivedSeverity is CLEARED and its ackState is not ACKNOWLEDED.";

leaf alarmId {

type string;

mandatory true;

description "Identifies the alarmRecord";

yext3gpp:notNotifyable;

}

leaf objectInstance {

type string;

config false ;

mandatory true;

yext3gpp:notNotifyable;

}

leaf notificationId {

type int32;

config false ;

mandatory true;

yext3gpp:notNotifyable;

}

leaf alarmRaisedTime {

type yang:date-and-time ;

config false ;

yext3gpp:notNotifyable;

}

leaf alarmChangedTime {

type yang:date-and-time ;

config false ;

description "not applicable if related alarm has not changed";

yext3gpp:notNotifyable;

}

leaf alarmClearedTime {

type yang:date-and-time ;

config false ;

description "not applicable if related alarm was not cleared";

yext3gpp:notNotifyable;

}

leaf alarmType {

type eventType;

config false ;

description "General category for the alarm.";

yext3gpp:notNotifyable;

}

leaf probableCause {

type string;

config false ;

yext3gpp:notNotifyable;

}

leaf specificProblem {

type string;

config false ;

reference "ITU-T Recommendation X.733 clause 8.1.2.2.";

yext3gpp:notNotifyable;

}

leaf perceivedSeverity {

type severity-level;

description "This is Writable only if producer supports consumer

to set perceivedSeverity to CLEARED";

yext3gpp:notNotifyable;

}

leaf backedUpStatus {

type string;

config false ;

description "Indicates if an object (the MonitoredEntity) has a back

up. See definition in ITU-T Recommendation X.733 clause 8.1.2.4.";

yext3gpp:notNotifyable;

}

leaf backUpObject {

type string;

config false ;

yext3gpp:notNotifyable;

}

leaf trendIndication {

type string;

config false ;

description "Indicates if some observed condition is getting better,

worse, or not changing. ";

reference "ITU-T Recommendation X.733 clause 8.1.2.6.";

yext3gpp:notNotifyable;

}

grouping ThresholdPackGrp {

leaf thresholdLevel {

type string;

}

leaf thresholdValue {

type string;

}

leaf hysteresis {

type string;

description "The hysteresis has a threshold high and a threshold

low value that are different from the threshold value.

A hysteresis, therefore, defines the threshold-high and

threshold-low levels within which the measurementType value is

allowed to oscillate without triggering the threshold crossing

notification.";

}

}

grouping ThresholdInfoGrp {

leaf measurementType {

type string;

mandatory true;

}

leaf direction {

type enumeration {

enum INCREASING;

enum DECREASING;

}

mandatory true;

description "

If it is 'Increasing', the threshold crossing notification is

triggered when the measurement value equals or exceeds a

thresholdValue.

If it is 'Decreasing', the threshold crossing notification is

triggered when the measurement value equals or below a

thresholdValue.";

}

uses ThresholdPackGrp;

}

list thresholdInfo {

config false ;

yext3gpp:notNotifyable;

uses ThresholdInfoGrp;

}

leaf stateChangeDefinition {

type string;

config false ;

description "Indicates MO attribute value changes. See definition

in ITU-T Recommendation X.733 clause 8.1.2.11.";

yext3gpp:notNotifyable;

}

leaf monitoredAttributes {

type string;

config false ;

description "Indicates MO attributes whose value changes are being

monitored.";

reference "ITU-T Recommendation X.733 clause 8.1.2.11.";

yext3gpp:notNotifyable;

}

leaf proposedRepairActions {

type string;

config false ;

description "Indicates proposed repair actions. See definition in

ITU-T Recommendation X.733 clause 8.1.2.12.";

yext3gpp:notNotifyable;

}

leaf additionalText {

type string;

config false ;

yext3gpp:notNotifyable;

}

anydata additionalInformation {

config false ;

yext3gpp:notNotifyable;

}

leaf rootCauseIndicator {

type enumeration {

enum YES;

enum NO;

}

config false ;

description "It indicates that this AlarmInformation is the root cause

of the events captured by the notifications whose identifiers are in

the related CorrelatedNotification instances.";

yext3gpp:notNotifyable;

}

leaf ackTime {

type yang:date-and-time ;

config false ;

description "It identifies the time when the alarm has been

acknowledged or unacknowledged the last time, i.e. it registers the

time when ackState changes.";

yext3gpp:notNotifyable;

}

leaf ackUserId {

type string;

description "It identifies the last user who has changed the

Acknowledgement State.";

yext3gpp:notNotifyable;

}

leaf ackSystemId {

type string;

description "It identifies the system (Management System) that last

changed the ackState of an alarm, i.e. acknowledged or unacknowledged

the alarm.";

yext3gpp:notNotifyable;

}

leaf ackState {

type enumeration {

enum ACKNOWLEDGED {

description "The alarm has been acknowledged.";

}

enum UNACKNOWLEDGED {

description "The alarm has unacknowledged or the alarm has never

been acknowledged.";

}

}

yext3gpp:notNotifyable;

}

leaf clearUserId {

type string;

description "Carries the identity of the user who invokes the

clearAlarms operation.";

yext3gpp:notNotifyable;

}

leaf clearSystemId {

type string;

yext3gpp:notNotifyable;

}

leaf serviceUser {

type string;

config false ;

description "It identifies the service-user whose request for service

provided by the serviceProvider led to the generation of the

security alarm.";

yext3gpp:notNotifyable;

}

leaf serviceProvider {

type string;

config false ;

description "It identifies the service-provider whose service is

requested by the serviceUser and the service request provokes the

generation of the security alarm.";

yext3gpp:notNotifyable;

}

leaf securityAlarmDetector {

type string;

config false ;

yext3gpp:notNotifyable;

}

}

grouping AlarmListGrp {

description "Represents the AlarmList IOC.";

leaf administrativeState {

type types3gpp:AdministrativeState ;

default LOCKED;

description "When set to UNLOCKED, the alarm list is updated.

When the set to LOCKED, the existing alarm records are not

updated, and new alarm records are not added to the alarm list.";

}

leaf operationalState {

type types3gpp:OperationalState ;

default DISABLED;

config false;

description "The producer sets this attribute to ENABLED, indicating

that it has the resource and ability to record alarm in AlarmList

else, it sets the attribute to DISABLED.";

}

leaf numOfAlarmRecords {

type uint32 ;

config false;

mandatory true;

description "The number of alarm records in the AlarmList";

yext3gpp:notNotifyable;

}

leaf lastModification {

type yang:date-and-time ;

config false;

description "The last time when an alarm record was modified";

yext3gpp:notNotifyable;

}

list alarmRecords {

key alarmId;

description "List of alarmRecords";

yext3gpp:notNotifyable;

uses AlarmRecordGrp;

}

}

grouping FmSubtree {

description "Contains FM related classes.

Should be used in all classes (or classes inheriting from)

- SubNetwork

- ManagedElement

If some YAM wants to augment these classes/list/groupings they must

augment all user classes!";

list AlarmList {

key id;

max-elements 1;

description "The AlarmList represents the capability to store and manage

alarm records. The management scope of an AlarmList is defined by all

descendant objects of the base managed object, which is the object

name-containing the AlarmList, and the base object itself.

AlarmList instances are created by the system or are pre-installed.

They cannot be created nor deleted by MnS consumers.

When the alarm list is locked or disabled, the existing alarm records

are not updated, and new alarm records are not added to the alarm list";

uses top3gpp:Top\_Grp ;

container attributes {

uses AlarmListGrp ;

}

}

}

}

<CODE ENDS>

### 10.1.2 Notifications

The Netconf-YANG solution set uses the JSON/OpenApi notifications.

## 10.2 OpenApi Definitions

### 10.1.1 NRM Definitions

#### 10.2.1 OpenAPI document "TS28545\_FmNrm.yaml"

TODO: Extract FM parts from 28.623 clause C.4.3 OpenAPI document "TS28623\_GenericNrm.yaml"

### 10.21.2 Notifications

TODO: Add FM notifications from 28.532 clause 12.2 Generic fault supervision management service

# Annex X (informative): Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2018-09 | SA#81 |  |  |  |  | Upgrade to change control version | 15.0.0 |
| 2018-12 | SA#82 | SP-181045 | 0001 | 1 | F | Correction for Requirements of Fault Supervision (FS) Service | 15.1.0 |
| 2020-07 | SA#88-e | SP-200501 | 0006 | 1 | F | Update clause 8 virtualized resource alarm correlation | 15.2.0 |
| 2020-07 | SA#88-e | SP-200490 | 0005 | 1 | B | Add description for MnS components used for configurable FM control | 16.0.0 |
| 2020-09 | SA#89e | SP-200724 | 0007 | 1 | F | Update description of MnS components used for configurable FM control | 16.1.0 |
| 2021-06 | SA#92e | SP-210407 | 0010 | 2 | B | Add missing Alarm Requirements and Use Cases | 17.0.0 |
| 2023-02 |  |  |  |  |  | Restructure document |  |

X.1 Change Log – to be removed before formal submission

### V11 to v12

Updated clause 4 Concepts and overview

1. Updated and added alarm state diagram.
2. Separated state diagram for consumer acknowledgement and producer automatic acknowledgement
3. Added basic concepts

Outstanding items:

* 10.2 JSON Definitions – Olaf’s text

### V10 to v11

Updated according to the endorsed S5-232771 and other offline discussions.

1. Reformatted Requirements
2. Added stage 2 and stage 3 for YANG, OpenApi yet missing.
3. Removed old use-cases, renamed new usage descriptions to use-case.
4. Move sections “main proposals” and “General Editorial Questions” into the discussion paper clause 3.
5. Add AlarmList.unreliableAlarmScope to NRM
6. Added data types AlarmComment and CorrelatedNotification
7. Added attribute definitions for all attributes in AlarmList, AlarmRecord, AlarmComment and CorrelatedNotification
8. Added notifyPotentialFaultyAlarmList
9. Added new references.