**3GPP TSG-SA5 Meeting #131e *S5-203389***

**e-meeting 25th May-3rd June 2020**

|  |
| --- |
| *CR-Form-v12.0* |
| **CHANGE REQUEST** |
|  |
|  | **28.623** | **CR** | **0091** | **rev** | **rev** | **Current version:** | **16.3.2** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network | **X** | Core Network | **X** |

|  |
| --- |
|  |
| ***Title:***  | Update FM control fragment (YANG definitions) |
|  |  |
| ***Source to WG:*** | Ericsson |
| ***Source to TSG:*** | S5 |
|  |  |
| ***Work item code:*** | eNRM |  | ***Date:*** | 2020-06-05 |
|  |  |  |  |  |
| ***Category:*** | **F** |  | ***Release:*** | Rel-16 |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-8 (Release 8)Rel-9 (Release 9)Rel-10 (Release 10)Rel-11 (Release 11)Rel-12 (Release 12)**Rel-13 (Release 13)Rel-14 (Release 14)Rel-15 (Release 15)Rel-16 (Release 16)* |
|  |  |
| ***Reason for change:*** | S5-203275 / S5-203424 Rel-16 CR 28.622 Update FM control fragment (stage 2) changed the stage 2 of the FM fragment. This is the corresponding YANG definition change |
|  |  |
| ***Summary of change:*** |  |
|  |  |
| ***Consequences if not approved:*** | Mismatch between Stage 2 and stage 3 |
|  |  |
| ***Clauses affected:*** |  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  | **X** |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  | **X** |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  | **X** |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** | YANG checked locally with pyang --strict |
|  |  |
| ***This CR's revision history:*** |  |

|  |
| --- |
| **First modification** |

## D.2.9 module \_3gpp-common-fm.yang

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; }

 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 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;

 config false ;

 mandatory true;

 description "Identifies the alarmRecord";

 }

 leaf objectInstance {

 type string;

 config false ;

 mandatory true;

 }

 leaf notificationId {

 type string;

 config false ;

 mandatory true;

 }

 leaf alarmRaisedTime {

 type yang:date-and-time ;

 config false ;

 }

 leaf alarmChangedTime {

 type yang:date-and-time ;

 config false ;

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

 }

 leaf alarmClearedTime {

 type yang:date-and-time ;

 config false ;

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

 }

 leaf alarmType {

 type eventType;

 config false ;

 description "General category for the alarm.";

 }

 leaf probableCause {

 type string;

 config false ;

 }

 leaf specificProblem {

 type string;

 config false ;

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

 }

 leaf perceivedSeverity {

 type severity-level;

 description "This is Writable only if producer supports consumer

 to set perceivedSeverity to CLEARED";

 }

 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.";

 }

 leaf backUpObject {

 type string;

 config false ;

 }

 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.";

 }

 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 ;

 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.";

 }

 leaf monitoredAttributes {

 type string;

 config false ;

 description "Indicates MO attributes whose value changes are being

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

 }

 leaf proposedRepairActions {

 type string;

 config false ;

 description "Indicates proposed repair actions. See definition in

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

 }

 leaf additionalText {

 type string;

 config false ;

 }

 leaf additionalInformation {

 type string;

 config false ;

 }

 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.";

 }

 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.";

 }

 leaf ackUserId {

 type string;

 description "It identifies the last user who has changed the

 Acknowledgement State.";

 }

 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.";

 }

 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.";

 }

 }

 }

 leaf clearUserId {

 type string;

 description "Carries the identity of the user who invokes the

 clearAlarms operation.";

 }

 leaf clearSystemId {

 type string;

 }

 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.";

 }

 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.";

 }

 leaf securityAlarmDetector {

 type string;

 config false ;

 }

 }

 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";

 }

 leaf lastModification {

 type yang:date-and-time ;

 config false;

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

 }

 list alarmRecords {

 key alarmId;

 description "List of alarmRecords";

 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 ;

 }

 }

 }

}

|  |
| --- |
| **Next modification** |

|  |
| --- |
| **End of modification** |