TSG-RAN Working Group3 meeting#6
Sophia Antipolis, 23-27 August 1999

TSG R3#6(99)961

Agenda Item: 21

Source: Siemens / Italtel

Title: Info Model and State Management Functions for NodeB logical O&M

Document for: Approval

---

1 Abstract

This contribution proposes to introduce an information model as basis for state management functions for the logical resources in NodeB according to ITU-T X.731. This contribution shows only the info model of the object oriented interface. The protocol of the object oriented interface is demonstrated in a separate contribution.

The contribution comprises the following parts:

1.) Information Model

It is shown how the logical O&M (logical resources) interface between RNC and NodeB can be described by means of an information model instead of a message based description (NBAP procedures). We will identify the main managed object classes of the NodeB logical O&M based on the logical model of NodeB shown in TS25.430. We will give a high level description of the classes, showing the behaviour, the main attributes, operations and notifications. A containment tree for the identified classes will also be provided. The description level will be sufficient to introduce the main concepts and especially the state management functions.

In our opinion the info model is the appropriate description method for an O&M interface. It contains information which is not provided in a simple message based description (e.g. containment hierarchy, relationships, initial values of attributes, auto creation of contained objects, auto deletion of contained objects, manager – agent role). In addition on basis of the info model existing management standards can be easily used (e.g. state management ITU-T X.731)

2.) State Management

On basis of the information model we propose to introduce state management according to ITU-T X.731. We will at least request:

- introduce administrative state, operational state, usage state, availability status, alarm status as attributes for the logical resources
- administrative state of NodeB logical resources can be set via C-RNC by the operator
- Node B logical resources notify changes of operational state by state change event reports, according ITU-T X.731.
- the states and the status attributes of the logical resources in NodeB can be requested by C-RNC

We will show the benefit of using state management by means of following procedures for:

- fault propagation
- creation / deletion of logical resources
- alignment procedures between NodeB and RNC.

2 Information Model

2.1 UTRAN O&M Resource Model

Figure 1 presents the distribution of managed objects modelling implementation dependent resources and logical resources for NodeB and RNC.
Implementation dependent resources of the Node B are managed directly by the NodeB Manager via the implementation dependent management interface. NodeB logical resources are managed from the RNC Manager via the RNC or managed from the RNC itself.
Implementation dependent resources and logical resources of the RNC are controlled by the RNC Manager via the implementation dependent management interface. Logical resources within the RNC consist of the logical resources of the RNC itself and of the logical resources of the connected Node Bs. They are referenced therefore in the figure as RNS Logical resources.
The dotted lines between the implementation dependent resources and the logical resources represent the relationship between the physical resources and the associated logical resources that they support.
The logical resources on NodeB are copies from the RNS logical resources. The logical resources on NodeB are controlled by RNC, i.e. the RNC is the 'master' of the configuration data of the NodeB logical resources.

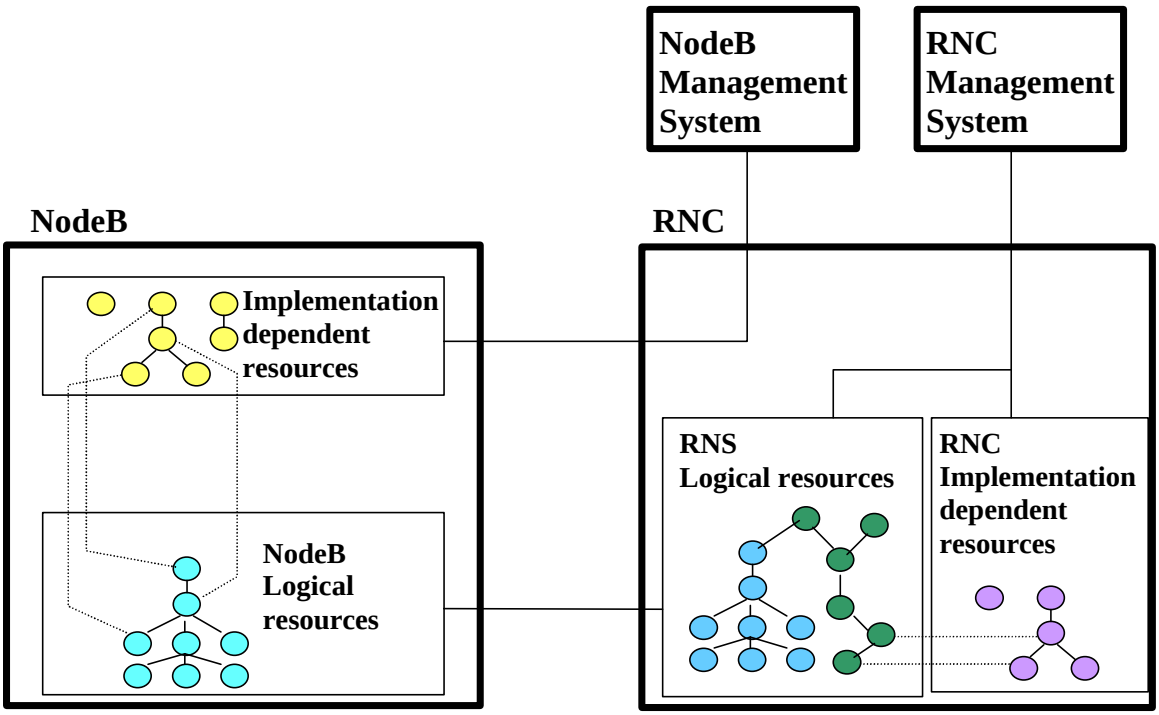

Figure 1: UTRAN O&M Resource Model

2.2 NodeB Info Model

The NodeB logical resources are modelled as managed objects within an information model. The types of the managed objects are called managed object classes (MOCs). The information model is given by the entirety of all object classes. By the information model the interface between RNC and NodeB concerning logical O&M is specified on a higher abstraction layer than a message based description. The focus is shifted towards the subject of management which are the managed resources. The static aspects of the resources are defined by their attributes, by their management operations and their notifications. The management operations can be invoked on the resources, the notifications are send by the resources.

Furthermore the information is available in an appropriate format that facilitates processing on the higher layers of network management, i.e. the effort for mediation of the management information to the higher management layers (network layer) in the network element manager can be limited using the object oriented approach thorougly.

2.2.1 Identify Managed Object Classes

The managed object classes are derived from 'elements of the logical model' of TS 25.430 and from the functionality which is provided by the currently defined NBAP O&M procedures.

It is not trivial that the point of view of the 'elements of the logical model' matches the management point of view. In fact we could have that some logical resources are very important for the telecommunication services but they are not manageable at all. In this case there is no need to model these resources as managed objects.

From point of view of management, the resources which are dynamically created and deleted on a per call basis are not visible to the management. They shall not be modelled as MOCs.
Only the statically configured resources *Node-B*, *Cell*, *Common Control Transport Channels (BCCH, PCCH, FACH, RACH)* are visible from point of view of management and are identified as managed object classes (MOCs).

The statically configured resource *Node-B Control Port* represents the interface to the Transport Network and is not considered as managed object class within this contribution.

Apart from the MOCs which are directly derived from the NodeB logical resources of TS 25.430 there will be auxiliary MOCs like the NodeBScanner, which models the performance measurement functionality which RNC can request on NodeB (see TS 25.433).

Furthermore we introduce the MOC *Carrier*, which allows to model a cell which consists of several RF carriers.

2.2.2 Containment Tree

The containment tree shows the containment hierarchy of the managed object classes. This means a *NodeB* managed object contains n *Cell* managed objects. A *Cell* managed object contains n *Carrier* managed objects and so on.

A description of the managed object classes can be found in chapter 2.2.3.

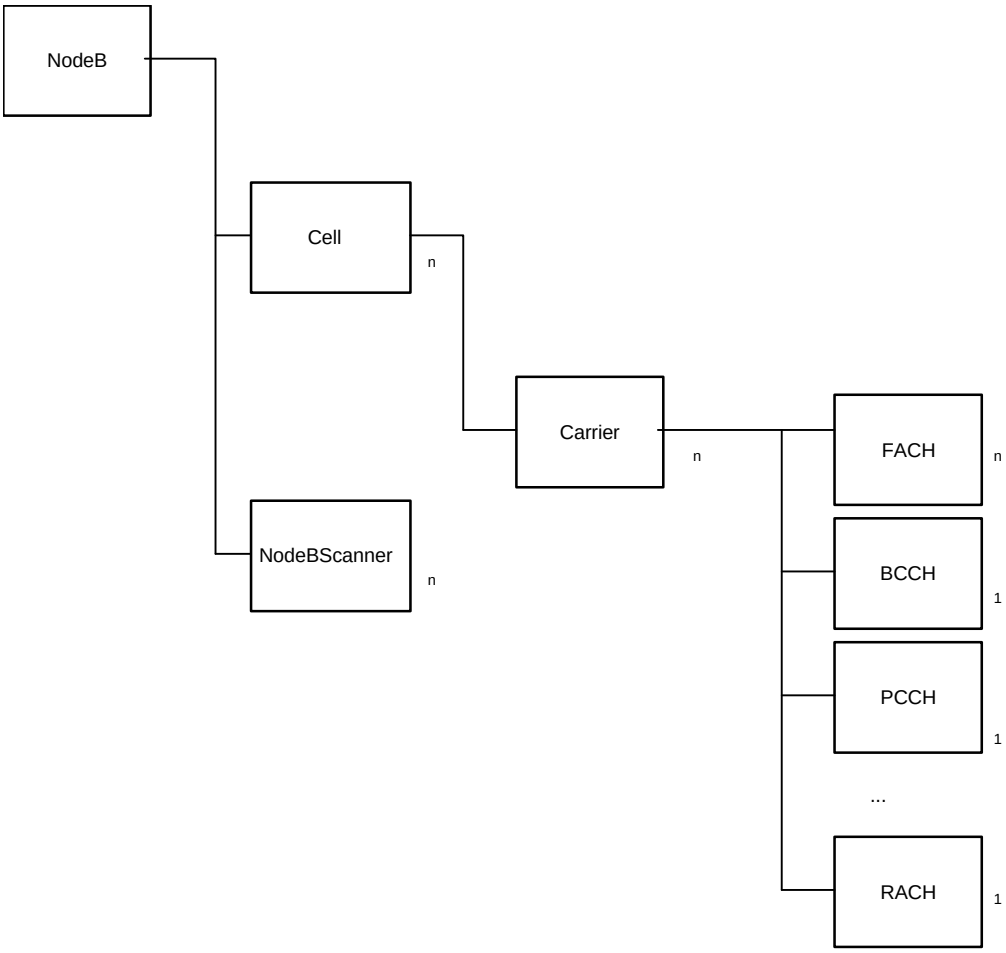

Figure 2: NodeB Managed Object Classes, Containment Tree

2.2.3 MOC Description

The description notation is a high level textual notation and does not contain any data types. The description comprises the class name, a textual description, the attributes, operations and the notifications of a managed object class.

[Editor's note] The description notation is presently a high level textual notation, which does not contain any data types.

2.2.3.1 MOC: NodeB

Description:
This class represents the whole NodeB (cell site) from point of view of RNC. The *NodeB* contains one or more *Cells*. An instance from *NodeB* is created / deleted by the operator when a

cell site is added to / deleted from RNS. *NodeB* also provides operations which are requested by RNC to inform *NodeB* about an RNC restart and operations to request an alignment procedure for state and alarm information.
NodeB provides state management functionality according ITU-T.X.731. When a *NodeB* performs a restart, this is indicated by sending the notification *StateChangeEventReport.*

Attributes:
AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);
ProceduralStatus (Get);

Contained Objects:
Cells (0 – n)

Operations:
Create(IN: AdminState, OUT: Result);
Delete(OUT:Result);
Get();
Replace();
RNCRestart(IN: ServiceImpactLevel);
RNCRestartComplete();
StateAlignmentRequest();
AlarmAlignmentRequest();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.2 MOC: Cell

Description:
Cell is contained in *NodeB. Cell* represents a collection of radio links. An instance from *Cell* is created / deleted by the operator when a *NodeB* (cell site) or an RF carrier is added to / deleted from a *NodeB*. The attribute *Cell Id* is used to identify the *Cell* within call processing procedures. The attribute *Local Cell Id* is used to model the relationship of *Cell* to the Implementation-dependent resources. *Cell* also provides state management functionality according to ITU-T X.731. A *Cell* can indicate when it waits for a change of its AdminState to 'locked' by sending the notification *WaitForBlockResource.*

Attributes:
LocalCellId (Get, Replace);
CellId (Get, Replace);
DLScramblingCode (Get, Replace);

AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);

Contained Objects:
Carrier (0 –n);

Operations:
Create(IN: LocalCellId, CellId, DLScramblingCode, AdminState, OUT: Result);
Delete(OUT: Result);
Get();
Replace();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
ConfigEventReport(AdminState; LocalCellId, CellId, DLScramblingCode, CarrierData[]);
WaitForBlockResource(PriorityIndicator);
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.3 MOC: Carrier

Description:
This class represents an RF carrier within a *Cell.* The *Carrier* instances are either created / during creation of a *Cell* or are created within an existing *Cell* when an RF carrier is added to this *Cell*. The same is valid for the deletion of a *Carrier*. The *Carrier* contains at least a complete set of Common Control Transport Channels (*RACH, PCCH, BCCH, FACH*). *Carrier* also provides state management functions according to ITU-T X.731. A *Carrier* can indicate when it waits for a change of its AdminState to 'locked' by sending the notification *WaitForBlockResource.*

Attributes:
MaxTransmissionPower (Get, Replace);
Frequency (Get, Replace);
AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);

Contained Objects:
RACH (1);
PCCH (1);
BCCH (1);
FACH (1-n),

Operations:
Create(IN: MaxTransmissionPower, Frequency, AdminState, OUT: Result);
Delete(OUT: Result);

Get();
Replace();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
WaitForBlockResource(Priority Indicator);
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.4 MOC: RACH

Description:
This class represents the UL Common Control Transport Channel RACH. An instance from *RACH* is implicitly created during the creation of a *Carrier* or explicitly created within an existing *Carrier*. An instance from *RACH* is deleted by the operator or implicitly by RNC when a *Carrier* is deleted. *RACH* also provides state management functions according to ITU-T X.731. A *RACH* can indicate when it waits for a change of its AdminState to 'locked' by sending the notification *WaitForBlockResource*

Attributes:
PreambleSpreadingCode (Get, Replace);
AllowedPreambleSignatures (Get, Replace);
AllowedSpreadingFactorsForTheMessagePart (Get, Replace);
PreambleToPreambleTiming (Get, Replace);
AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);
...

Operations:
Create(IN: AdminState,ChannelData OUT: Result);
Delete(OUT: Result);
Get();
Replace();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
WaitForBlockResource;
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.5 MOC: FACH

Description:
This class represents the DL Common Control Transport Channel FACH. An instance from *FACH* is implicitly created during the creation of a *Carrier* or explicitly created within an existing *Carrier*. An instance from *FACH* is deleted by the operator or implicitly by RNC when a *Carrier* is deleted. *FACH* also provides state management functions according to ITU-T X.731. A *FACH* can indicate when it waits for a change of its AdminState to 'locked' by sending the notification *WaitForBlockResource*

Attributes:
DLScramblingCodeID (Get, Replace);
DLChannelisationCodeNumber (Get, Replace);
DLChannelisationCodeSpreadingFactor (Get, Replace);
AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);
...

Operations:
Create(IN: AdminState,ChannelData OUT: Result);
Delete(OUT: Result);
Get();
Replace();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
WaitForBlockResource;
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.6 MOC: BCCH

Description:
This class represents the DL Common Control Transport Channel BCCH. An instance from *BCCH* is implicitly created during the creation of a *Carrier* or explicitly created within an existing *Carrier*. An instance from *BCCH* is deleted by the operator or implicitly by RNC when a *Carrier* is deleted. *BCCH* also provides state management functions according to ITU-T X.731. A *BCCH* can indicate when it waits for a change of its AdminState to 'locked' by sending the notification *WaitForBlockResource*

Attributes:
DLScramblingCodeID (Get, Replace);
DLChannelisationCodeNumber (Get, Replace);
DLChannelisationCodeSpreadingFactor (Get, Replace);
BCCHPower (Get, Replace);
AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);

...

Operations:
Create(IN: AdminState,ChannelData OUT: Result);
Delete(OUT: Result);
Get();
Replace();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
WaitForBlockResource;
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.7 MOC: PCCH

Description:
This class represents the DL Common Control Transport Channel PCCH. An instance from *PCCH* is implicitly created during the creation of a *Carrier* or explicitly created within an existing *Carrier*. An instance from *PCCH* is deleted by the operator or implicitly by RNC when a *Carrier* is deleted. *PCCH* also provides state management functions according to ITU-T X.731. A *PCCH* can indicate when it waits for a change of its AdminState to 'locked' by sending the notification *WaitForBlockResource*

Attributes:
DLScramblingCodeID (Get, Replace);
DLChannelisationCodeNumber (Get, Replace);
DLChannelisationCodeSpreadingFactor (Get, Replace);
PCCHPower (Get, Replace);
AdminState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);
...

Operations:
Create(IN: AdminState,ChannelData OUT: Result);
Delete(OUT: Result);
Get();
Replace();

Notifications:
StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
WaitForBlockResource;
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

2.2.3.8 MOC: NodeBScanner

Description:
This class models a performance measurement job in NodeB. This object shall at least comprise the functionality of the performance measurement procedures currently defined in TS 25.433. A NodeBScanner collects and reports measurement data for the specified measurement types, measurement characteristics and report characteristics. The modelling is based on the class *scanner* in ITU-T X.738. *NodeBScanner* also provides state management functions according to ITU-T X.731.

Attributes:
AdministrativeState (Get, Replace);
OperationalState (Get);
AvailabilityStatus (Get);
AlarmStatus (Get);
...
NodeBMeasurementType (Get-Replace);
The type of the measurement (e.g. 'interference of the uplink', 'DL Cell Power Load')
MeasurementCharacteristics (Get-Replace);
This defines how the measurement shall be performed. For example measurement frequency, timing information, filtering information
ReportCharacteristics (Get-Replace);
The reporting could be any of the following classes:
Periodic: Reports should be delivered in a periodic matter with some frequency. In this case the update frequency have to be specified.
Event Triggered: Reports should be delivered upon a specific event in Node B e.g. Performance threshold crossing. In this case the event have to be specified.
Immediate Reporting: A report should be delivered immediately. Only one measurement report should be sent and after that the measurement is automatically cancelled.

Operations:

Create(IN: AdministrativeState, MeasurementObjectID, NodeBMeasurementType, MeasurementCharacteristics, ReportCharacteristics;
OUT: Result);
Delete(OUT: Result);
Get();
Replace();

Notifications:

StateChangeEventReport (acc. ITU_T X.731);
FailureEventReport (acc. ITU_T X.733);
ScanReport(TimeReference, MeasurementValue);
CreateEventReport(ObjectID, Result);
DeleteEventReport(Result)

3 State Management

On basis of the information model the state management according to ITU-T X.731 is introduced. The state information and status attributes are added to all managed object classes. State changes are notified by related managed objects by the notification *StateChangeEventReport*.
By means of the introduced states the operator is able to monitor the following aspects of a resource:
Operability (operational state): whether or not the resource is physically installed and working, i.e. is able to provide service;
Usage (usage state): whether or not the resource is actively in use at a specific instant, and if so, whether or not it has spare capacity for additional users at that instant.

Furthermore the permission to use or prohibition against using the resource can be administered by the operator (administrative state).

The operational state and the associated status attributes (availability status, alarm status) is set by the resource itself. They can only be read by the operator but can not be set.
They indicate when the resource is not able to provide service or suffers from service degradation due to an network element internal problem. The usage state indicates if a resource is already used by other managed objects and if there is still spare capacity for other managed objects available. The usage state is determined by the network element itself, it can be read but not set by the operator.
The administrative state allows to prevent a resource from providing service. The administrative state is set by the operator.

3.1 State Management Procedures

Besides the benefit for the operator that he can monitor the operability and usage information, the introduced states can be used for:

Fault propagation:
Within the information model all types of relationships (e.g. containment, service-user ⇔ service-provider, co-operation) can be modelled very easily. The scope of standardisation is only on the info model for the NodeB logical resources. Within this info model only relationships between logical resources themselves can be modelled. But the implementation dependant resources can be modelled in an Implementation dependent info model by each vendor as well. When connecting both information models, the relationships between the logical resources and their supporting equipment are given. This is the precondition to provide fault propagation from equipment resources to logical resources.
Every change of the operational state of one managed object must be propagated by an internal mechanism to all related managed objects (e.g. see Figure 3: Fault Propagation). Because the operational state and its status attributes have a fixed set of values, a managed object knows how it is affected by a state change of a related object.

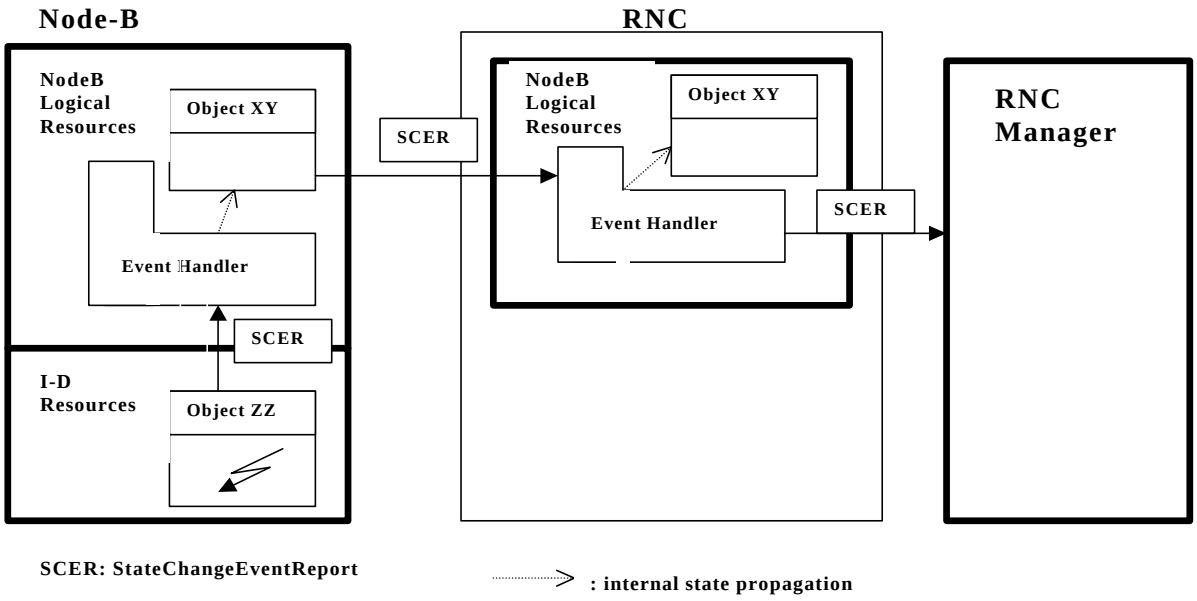

Figure 3: Fault Propagation

Creation and configuration of managed objects in advance:
A managed object can be created and configured before the real hardware is available. The operational state / availability status is set to 'disabled / offline' by the managed object itself when it is created and configured, before the related hardware is available. Furthermore the operator can set the administrative state of the managed object to *'locked'* when the related hardware is not available in the system. So he prevents the resource from sending any alarms or state changes before the hardware is available.

Alignment:
After restart of RNC or of NodeB or after recovery of the O&M link between RNC and NodeB the object representations between RNC and NodeB must be aligned. This comprises the configuration data of the logical resources which must be aligned from RNC to NodeB and the spontaneous network element information which must be aligned from NodeB to RNC.

The spontaneous network element information is given by the operational state, usage state and the associated status attributes. The alignment can be done by sending a *StateChangeEventReport* to RNC for every logical resource managed object which differ from a default state. The default state shall indicate that the resource is fully operational without any degradation. The start of the alignment must be requested by the C-RNC. The end of the alignment procedure must be indicated by Node-B either implicitly by a parameter within the last *StateChangeEventReport* or by a separate notification.

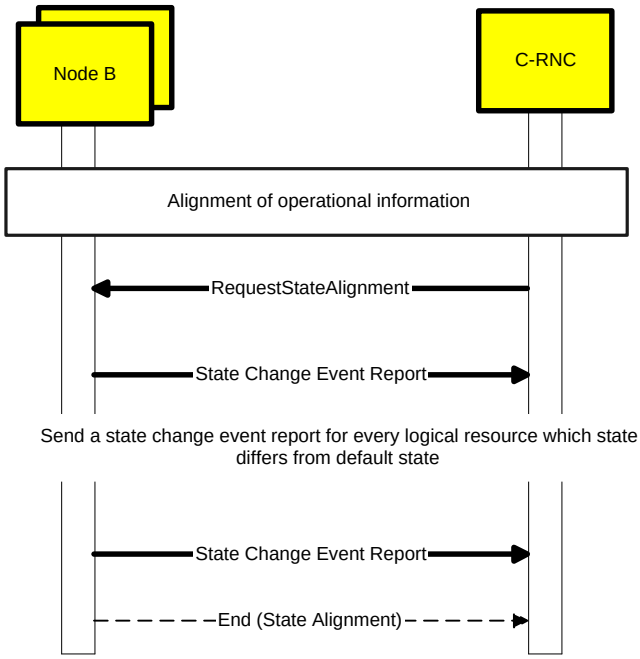

Figure 4: State Alignment Procedure

4 Conclusion

It is proposed:

- to insert the content of chapter 2.2 in this contribution to chapter 9.1.x in document TS25.433.
- to insert the content of chapter 3 in this contribution to chapter 5.x in document I3.05.