INTERNATIONAL TELECOMMUNICATION UNION

# ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

# Q.2630.2

**Annex D**

(04/2002)

SERIES Q: SWITCHING AND SIGNALLING

Broadband ISDN – Common aspects of B-ISDN application protocols for access signalling and network signalling and interworking

## AAL type 2 signalling protocol – Capability set 2

## Annex D: SDL definition of the AAL type 2 signalling protocol CS-2

ITU-T Recommendation Q.2630.2 – Annex D

ITU-T Q-SERIES RECOMMENDATIONS

**SWITCHING AND SIGNALLING**

| | |
|---|---|
| SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE | Q.1–Q.3 |
| INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING | Q.4–Q.59 |
| FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN | Q.60–Q.99 |
| CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS | Q.100–Q.119 |
| SPECIFICATIONS OF SIGNALLING SYSTEM No. 4 | Q.120–Q.139 |
| SPECIFICATIONS OF SIGNALLING SYSTEM No. 5 | Q.140–Q.199 |
| SPECIFICATIONS OF SIGNALLING SYSTEM No. 6 | Q.250–Q.309 |
| SPECIFICATIONS OF SIGNALLING SYSTEM R1 | Q.310–Q.399 |
| SPECIFICATIONS OF SIGNALLING SYSTEM R2 | Q.400–Q.499 |
| DIGITAL EXCHANGES | Q.500–Q.599 |
| INTERWORKING OF SIGNALLING SYSTEMS | Q.600–Q.699 |
| SPECIFICATIONS OF SIGNALLING SYSTEM No. 7 | Q.700–Q.799 |
| Q3 INTERFACE | Q.800–Q.849 |
| DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1 | Q.850–Q.999 |
| PUBLIC LAND MOBILE NETWORK | Q.1000–Q.1099 |
| INTERWORKING WITH SATELLITE MOBILE SYSTEMS | Q.1100–Q.1199 |
| INTELLIGENT NETWORK | Q.1200–Q.1699 |
| SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000 | Q.1700–Q.1799 |
| SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC) | Q.1900–Q.1999 |
| BROADBAND ISDN | Q.2000–Q.2999 |
| General aspects | Q.2000–Q.2099 |
| Signalling ATM adaptation layer (SAAL) | Q.2100–Q.2199 |
| Signalling network protocols | Q.2200–Q.2299 |
| **Common aspects of B-ISDN application protocols for access signalling and network signalling and interworking** | **Q.2600–Q.2699** |
| B-ISDN application protocols for the network signalling | Q.2700–Q.2899 |
| B-ISDN application protocols for access signalling | Q.2900–Q.2999 |

*For further details, please refer to the list of ITU-T Recommendations.*

# ITU-T Recommendation Q.2630.2

## AAL type 2 signalling protocol – Capability set 2

## Annex D

## SDL definition of the AAL type 2 signalling protocol CS-2

### Summary

This new annex contains a comprehensive definition of the AAL type 2 signalling procedures in SDL. The data structures required for the SDL definition as well as the data structure for the messages in the main part of the Recommendation are specified in ASN.1.

Although, the SDL definitions contain more detail than the prose definition in clause 8, the definitions in clause 8 take precedence. Nevertheless, this new annex clearly indicates which part of the procedures specified in clause 8 are left to individual implementations and which part of the procedures must be implemented precisely for conformance with this Recommendation.

In addition, this new annex demonstrates that semi-formal definitions for signalling protocols are possible.

### Source

Annex D to ITU-T Recommendation Q.2630.2 was prepared by ITU-T Study Group 11 (2001-2004) and approved under the WTSA Resolution 1 procedure on 13 April 2002.

## FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

## NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

## INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

# CONTENTS

# ITU-T Recommendation Q.2630.2

# AAL type 2 signalling protocol – Capability set 2

# Annex D

# SDL definition of the AAL type 2 signalling protocol CS-2

The SDL definitions may contain more detail than the prose definition in clause 8 of the main body of this Recommendation. Nevertheless, should there exist any technical difference between this Annex and clause 8, then the definitions in clause 8 take precedence.

## D.1 Introduction

The SDL definitions of the AAL type 2 signalling protocol described in this Recommendation depend on the SDL system and block structure diagrams defined in this annex.

The SDL definition in this annex assumes that only a single event occurs at a given time, hence, no racing condition within the AAL type 2 signalling entity are considered; resolution of such collisions and racing conditions remains implementation dependent.

## D.2 The SDL system diagram

The SDL system diagram is depicted in Figure B.1/Q.2630.1; it is unchanged from the definition in ITU-T Rec. Q.2630.1.

## D.3 The SDL block structure diagram

The SDL block structure diagrams are depicted in Figure B.2/Q.2630.1 (parts 1 to 4 of 4); several signal lists are extended from the ones defined in ITU-T Rec. Q.2630.1.

NOTE 1 – The block USER and its process USER (not shown) are not part of the AAL type 2 signalling entity but used to indicate different served user entities.

NOTE 2 – The procedures located in process NodalF2 and called by the process NodalF1 are not elaborated further in this annex.

NOTE 3 – The procedure calls by the process NodalF1 to procedures located in process NodalF2 evoke an exchange of implicit signals between the processes NodalF1 and NodalF2.

NOTE 4 – One STI entity exists per signalling transport converter. These converters are known by nodal function 2 with their (SDL) ProcessID. The addition or removal of signalling relations together with the creation or destruction of the STI and STC processes is not shown in these SDL diagrams of this annex.

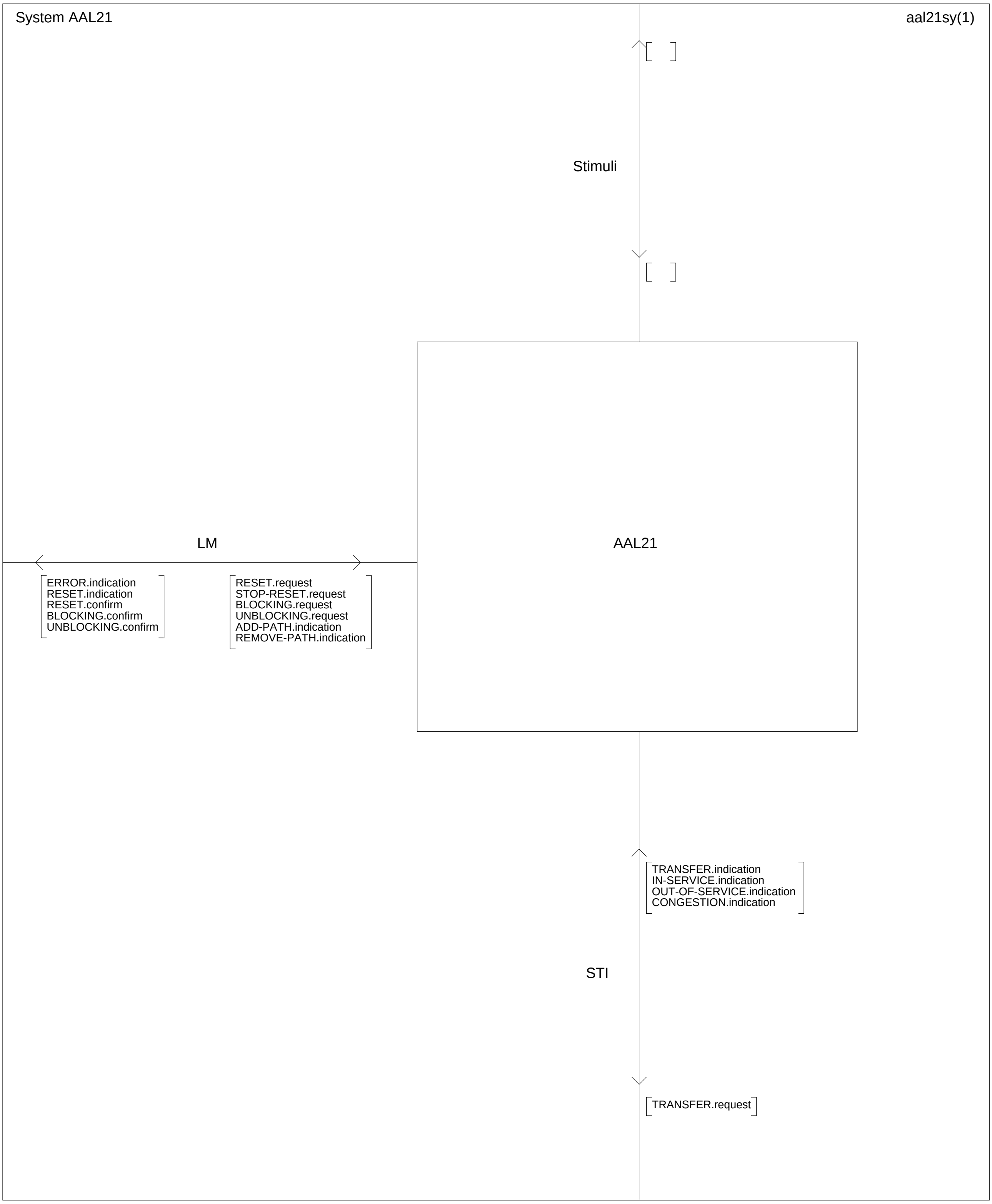


**Figure D.1/Q.2630.2 – SDL system of the AAL type 2 signalling entity**

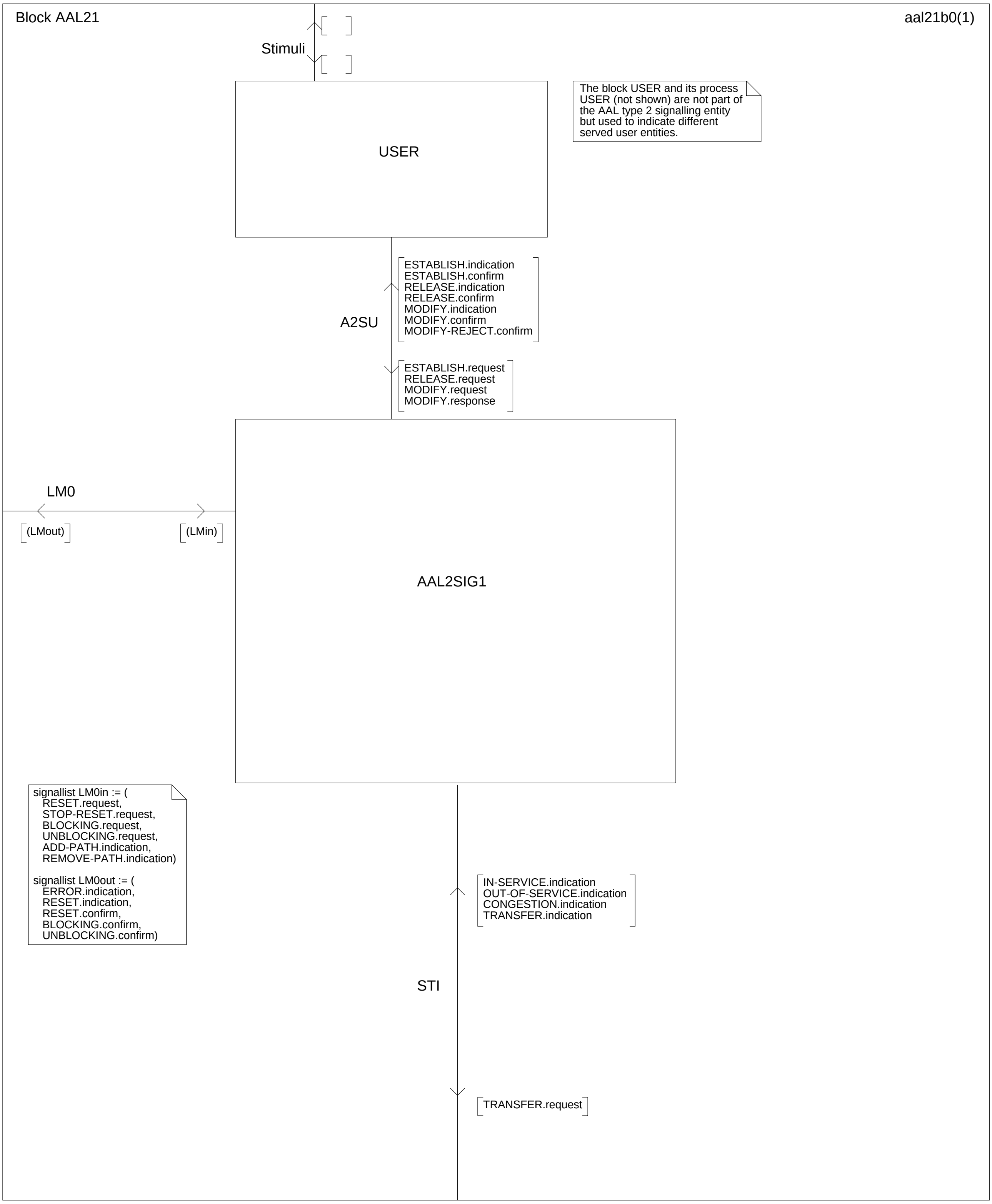


**Figure D.2/Q.2630.2 – SDL block structure of the AAL type 2 signalling entity *(part 1 of 4)***

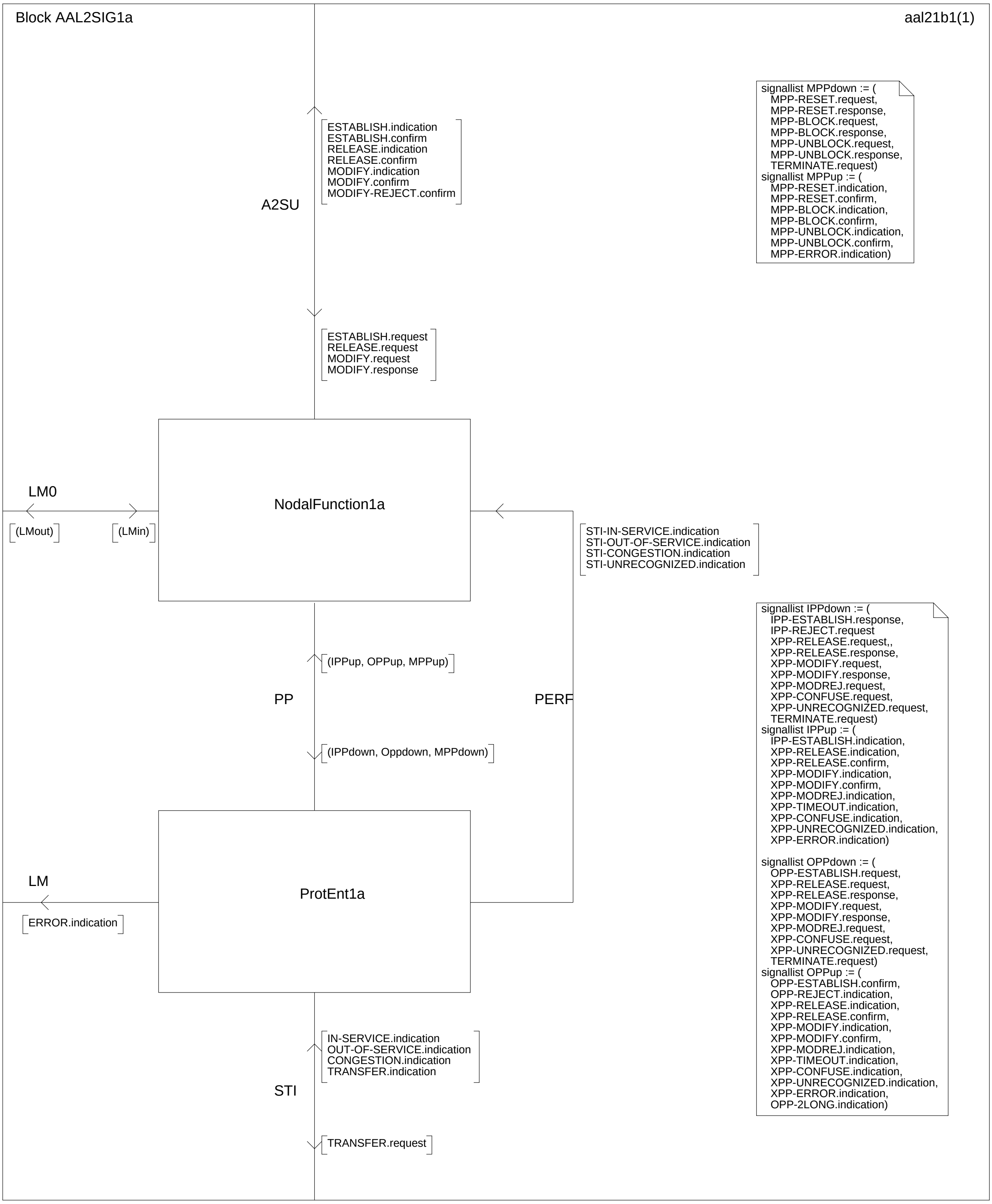


**Figure D.2/Q.2630.2 – SDL block structure of the AAL type 2 signalling entity *(part 2 of 4)***

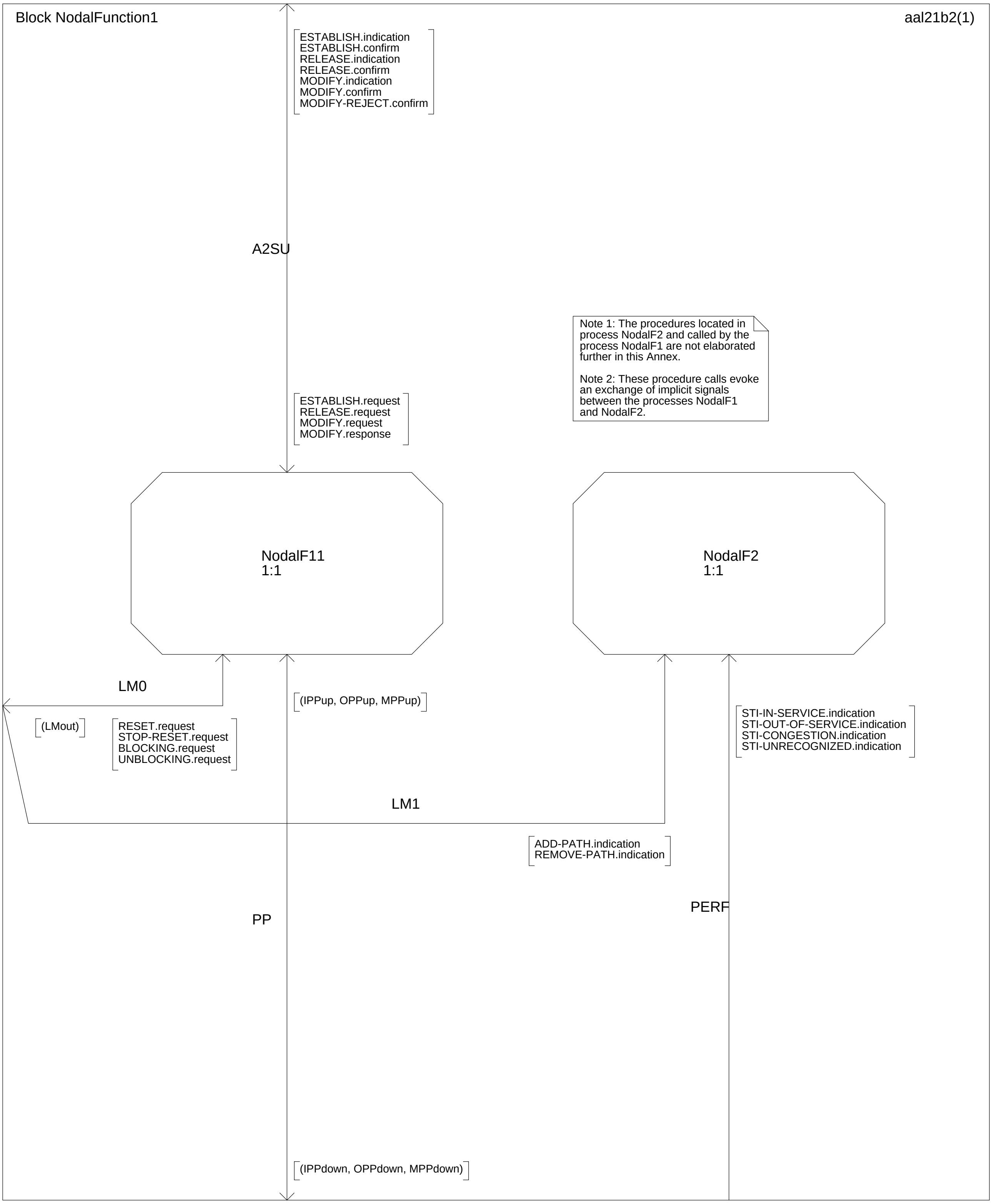


**Figure D.2/Q.2630.2 – SDL block structure of the AAL type 2 signalling entity *(part 3 of 4)***

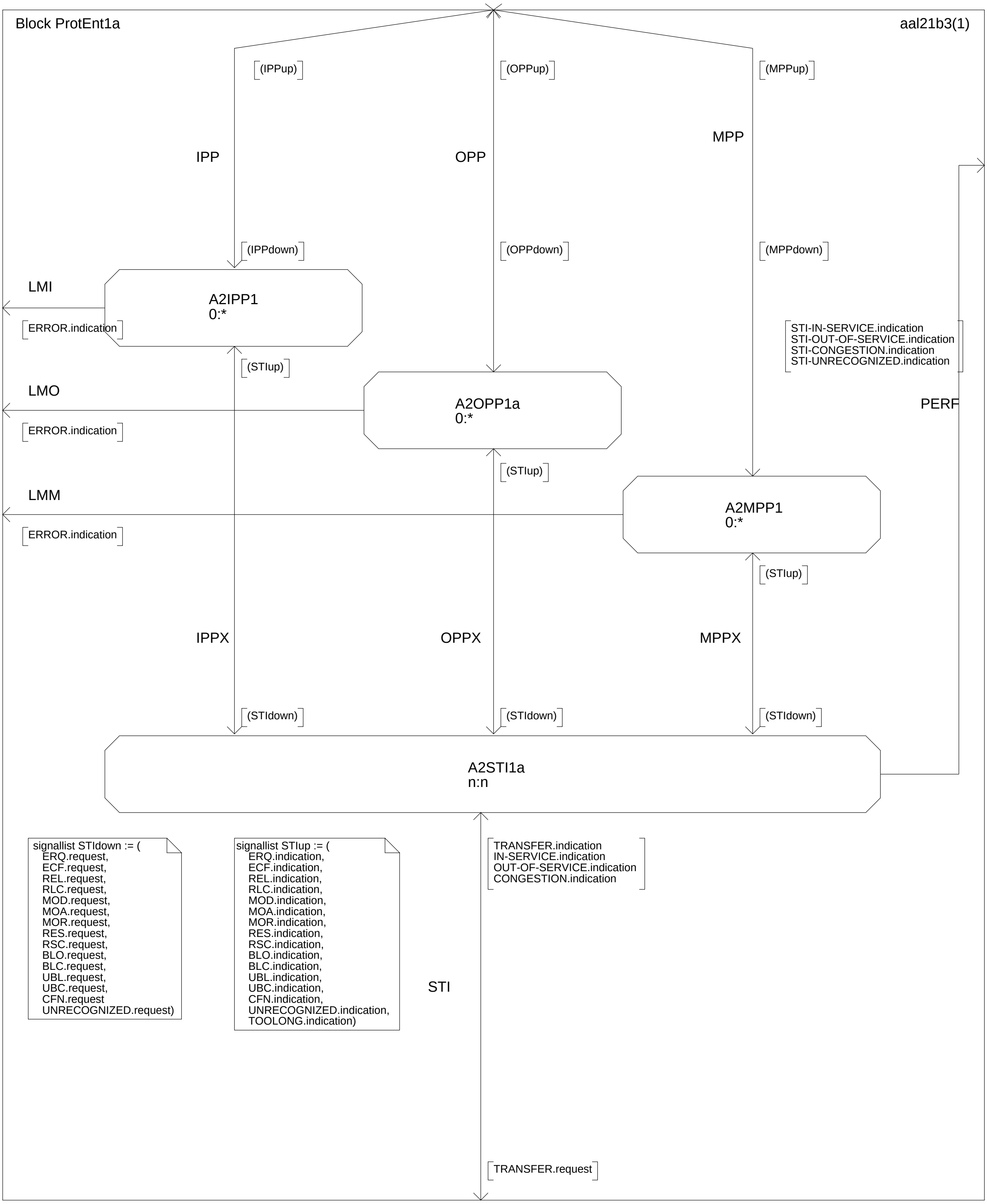


**Figure D.2/Q.2630.2 – SDL block structure of the AAL type 2 signalling entity *(part 4 of 4)***

## D.4 SDL specification for the nodal function

### D.4.1 Introduction

B.4.1/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D.

### D.4.2 SDL diagrams for the nodal function 1

#### D.4.2.1 Data structures

B.4.2.1/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modifications are required:

Replace Table B.1/Q.2630.1 with Table D.1/Q.2630.2:

**Table D.1/Q.2630.2 – Status values for CRec records**

| Description | Status Value | Incoming part describes | Outgoing part describes |
|---|---|---|---|
| Establishment Pending | 2 | preceeding link | succeeding link |
| | 3 | succeeding link | preceeding link |
| Connection established | 4 | preceeding link | succeeding link |
| | 5 | succeeding link | preceeding link |
| Release Pending | 6 | preceeding link | succeeding link |
| | 7 | succeeding link | preceeding link |
| Modify Pending | 8 | preceeding link | succeeding link |
| | 9 | succeeding link | preceeding link |

The structure of the record of type "CRec" is defined in the ASN.1 fragment below:

```
CRec ::= SEQUENCE {
     status                           CRecStatus,    -- connection status
     curresrce   CHOICE (                           -- current allocated resource
             lc                   LC,            -- link characteristics
             ssis                 SSIS },        -- SSCS information
     reqresrce   CHOICE {                           -- requested resource
             lctype   SEQUENCE {
                   lc             LC,            -- link characteristics
                   plc            LC OPTIONAL }, -- preferred link
                                                -- characteristics
             ssistype   SEQUENCE {
                   ssis           SSIS,          -- SSCS information
                   pssis          PSSIS } },     -- preferred SSCS information
     reqresrcetype                ResourceType,  -- type of requested resource
     incoming                     HRec,          -- details incoming link
     outgoing                     HRec }         -- details outgoing link

SSIS ::= CHOICE {                               -- SSCS information
                   ssia           SSIA,          -- service specific
                                                -- information (audio)
                   ssiae          SSIAE,         -- service specific
                                                -- information
                                                -- (audio ext.)
                   ssim           SSIA,          -- service specific info.
                                                -- (multirate)
                   ssime          SSIAE,         -- service specific info.
                                                -- (multirate ext.)
                   ssisa          SSIA,          -- service specific info.
                                                -- (SAR-assured)
                   ssisu          SSIA }         -- service specific info.
                                                -- (SAR-unassured)

PSSIS ::= CHOICE {                              -- preferred SSCS information
       ssiae                     SSIAE,         -- service specific
                                                -- information
                                                -- (audio ext.)
       ssime                     SSIAE }        -- service specific info.
                                                -- (multirate ext.)
```

```
HRec ::= SEQUENCE {
     peer          ENUMERATED {user,remote,none},
     ppus          PID,                    -- ID of protocol entity or user
     srid          PID,                    -- signalling relation identifier
                                           -- is ID of signalling relation
     ceid          CEID }                  -- connection element identifier
```

Add the following paragraph and Table D.2/Q.2630.2 after the definition of "CRec":

The type of requested resource is summarized in Table D.2.

**Table D.2/Q.2630.2 – Type of requested resource**

| Requested Resource Type | Parameter in ERQ Message | Type Value | Modification |
|---|---|---|---|
| | Default | 0 | not permitted |
| Resource based on SSIS | Default | 1 | SSIS change permitted |
| Resource based on LC | Default | 2 | LC change permitted |
| Resource based on SSIS | SSIS | 3 | not permitted |
| Resource based on SSIS | SSIS, MSSSI | 4 | SSIS change permitted |
| Resource based on SSIS | SSIS, MSSSI, PSSIS | 5 | SSIS change permitted |
| Resource based on LC | LC | 6 | not permitted |
| Resource based on LC | LC, MSLC | 7 | LC change permitted |
| Resource based on LC | LC, MSLC, PLC | 8 | LC change permitted |

Add the following paragraph and Table D.2/Q.2630.2 after the definition of "MRec":

The status values for MRec records is summarized in Table D.3.

**Table D.3/Q.2630.2 – Status values for MRec records**

| Description | Status Value | Record describes | Preceeding Link |
|---|---|---|---|
| Outgoing Block Pending | 2 | succeeding link | does not exist |
| Outgoing Unblock Pending | 4 | succeeding link | does not exist |
| Outgoing Reset Pending | 6 | succeeding link | does not exist |

### D.4.2.2 Primitives between Nodal Function 1 and the Protocol Entities

B.4.2.2/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modifications are required:

Replace Table B.2/Q.2630.1 with Table D.4/Q.2630.2:

**Table D.4/Q.2630.2 – Primitives and parameters exchanged between the Nodal Function 1 entity and the Protocol Entities**

| Primitive<br>Generic Name | Type | | | |
|---|---|---|---|---|
| | **Request** | **Indication** | **Response** | **Confirm** |
| IPP-ESTABLISH | Not defined | ERQmsg, sri | ECFmsg | Not defined |
| IPP-REJECT | RLCmsg | Not defined | Not defined | Not defined |
| OPP-ESTABLISH | ERQmsg | Not defined | Not defined | ECFmsg |
| OPP-REJECT | Not defined | RLCmsg | Not defined | Not defined |
| XPP-RELEASE | RELmsg | RELmsg | RLCmsg | RLCmsg |
| XPP-MODIFY | MODmsg | MODmsg | MOAmsg | MOAmsg |
| XPP-MODREJ | MORmsg | MORmsg | Not defined | Not defined |
| XPP-TIMEOUT | Not defined | – | Not defined | Not defined |
| XPP-CONFUSE | CNFmsg | CNFmsg | Not defined | Not defined |
| XPP-UNRECOGNIZED | msg | msg | Not defined | Not defined |
| MPP-BLOCK | BLOmsg, sqc | BLOmsg, sri | BLCmsg | BLCmsg |
| MPP-UNBLOCK | UBLmsg | UBLmsg, sri | UBCmsg | UBCmsg |
| MPP-RESET | RESmsg | RESmsg, sri | RSCmsg, sqc | RSCmsg |
| MPP-ERROR | Not defined | cause | Not defined | Not defined |
| TERMINATE | – | Not defined | Not defined | Not defined |
| – This primitive has no parameters | | | | |

Add the following items to the list:

n) **XPP-RELEASE**

An incoming or outgoing protocol entity is instructed to start modify procedures with a modify request (MOD) message, this message is conveyed to the protocol entity (with the request primitive). If an incoming or outgoing protocol entity receives such a modify request (MOD) message, this message is conveyed to nodal function 1 (with the indication primitive). After receipt of the modify request (MOD) message, the nodal function confirms a successful modify with a modify confirm (MOC) message; this message is conveyed to the protocol entity (with the response primitive). If an incoming or outgoing protocol entity receives a modify confirm (MOC) message, this message is conveyed to nodal function 1 (with the confirm primitive).

o) **XPP-MODREJ**

The nodal function indicates an unsuccessful modify with a modify reject (MOR) message; this message is conveyed to the protocol entity (with the request primitive). If an incoming or outgoing protocol entity receives a modify reject (MOR) message, this message is conveyed to nodal function 1 (with the indication primitive).

The reaction to input signal events is described in parts 1 to 21 (of 37) in Figure D.3/Q.2630.2. The following modifications were applied to the input signal processing in B.3/Q.2630.1:

In part 1 of Figure D.3/Q.2630.2 the declaration of variables for the new messages (MOD, MOA, and MOR) have been added.

Parts 2 to 7 (establishment procedure) have been amended to cater for the new but backward compatible CAC procedures. Also, Note 2 in part 7 was amended to permit the optional redirection procedure.

Parts 8 and 9 (Modify Request), parts 10 and 11 (Modify Acknowledge), and part 12 (Modify Reject) have been added.

Parts 8 and 9 of B.3/Q.2630.1 apply unmodified as parts 13 and 14 of Figure D.3/Q.2630.2.

From part 10 of Figure B.3/Q.2630.1, the specification for XPP-CONFUSE.indication and XPP-ERROR.indication applies unmodified as part 15 of D.3/Q.2630.2.

The timeout procedures from part 10 of B.3/Q.2630.1 are amended to cater for the timeout on the MOD (Modify Request) message in the established phase as part 16 of Figure D.3/Q.2630.2.

Part 11 of B.3/Q.2630.1 applies unmodified as part 17 of Figure D.3/Q.2630.2.

The management procedures in parts 12 to 15 of B.3/Q.2630.1 apply unmodified as parts 18 to 21 of Figure D.3/Q.2630.2.

### D.4.2.3 Procedures

B.4.2.3/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modifications are required:

The procedures are described in parts 22 to 27 (of 37) in Figure D.3/Q.2630.2. The following modifications were applied to the procedures in B.3/Q.2630.1:

Parts 16, 17, 19, 20, and 21 of B.3/Q.2630.1 apply unmodified as parts 22, 23, 25 26, and 27 of Figure D.3/Q.2630.2.

Part 18 of B.3/Q.2630.1 applies with the additional specification that a reset of a link may interrupt a modification procedure as part 24 of Figure D.3/Q.2630.2.

### D.4.2.4 Macros

B.4.2.3/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modifications are required:

The macros are described in parts 28 to 37 (of 37) of Figure D.3/Q.2630.2.

The following macros already defined in ITU-T Rec. Q.2630.1 needed amendments as follows:

The macros "**Construct ERQmsg**" (in part 28 of Figure D.3/Q.2630.2), "**Construct ECFmsg**", "**Extract ECFparams**" (in part 29 of Figure D.3/Q.2630.2), and "**Extract ERQparams**" (in part 30 of Figure D.3/Q.2630.2), have been modified to allow the treatment of the new parameters.

The following macros are defined in addition to those in ITU-T Rec. Q.2630.1.

The macro "**ReturnModRej**" (in part 31 of Figure D.3/Q.2630.2) returns a MOR (Modify Reject) message towards the originator of the modify request.

The macro "**Construct MORmsg**" (in part 33 of Figure D.3/Q.2630.2) indicates that the respective message is constructed containing a cause (CAU) parameter.

The macro "**DetCACParam**" (in part 34 of Figure D.3/Q.2630.2) determines the desired CAC parameters based on the presence of the parameters LC, MSLC, and PLC (or, as a network option, SSIS, MSSSIS, and PSSIS).

The macro "**AdjCACParam**" (in part 35 of Figure D.3/Q.2630.2) adjusts the CAC parameters based on the parameters in the ECF (Establish Confirm) or MOA (Modify Acknowledge) messages.

The macro "**Construct MODmsg**" (in part 36 of Figure D.3/Q.2630.2) provides the necessary details for the construction of the MOD (Modify) message. In particular, parameters are added to the message dependent on the parameters in the MODIFY.request primitive from the served user.

The macro "**Extract MODParams**" (in part 36 of Figure D.3/Q.2630.2) extracts the parameters for the MODIFY.indication primitive from the parameters of the MOD (Modify) message.

The macro "**Construct MOAmsg**" (in part 36 of Figure D.3/Q.2630.2) provides the necessary details for the construction of the MOA (Modify Acknowledge) message. In particular, parameters are added to the message dependent on the parameters in the MODIFY.response primitive from the served user.

The macro "**Extract MOAParams**" (in part 36 of Figure D.3/Q.2630.2) extracts the parameters for the MODIFY.confirm primitive from the parameters of the MOA (Modify Acknowledge) message.

The macro "**ModCACParam**" (in part 37 of Figure D.3/Q.2630.2).

The macro "**Construct MOAmsg**" (in part 31 of Figure D.3/Q.2630.2) indicates that the respective message is constructed without any parameters.

### D.4.3 Procedures in the nodal function

B.4.2.3/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modifications are required:

Replace the definitions for the functions "PathRes", "SelectRoute", and "SwitchRoute" with the following definition:

The function "**PathRes**" performs a connection admission control followed by the resource reservation on an incoming link (the preceeding link) during connection establishment. It also performs a connection admission control followed by the resource reservation on an incoming or outgoing link during resource modification.

The function returns a value that is defined by the following ASN.1 structure:

```
PathRes ::= SEQUENCE {                         -- return value of procedure
                                               -- PathResource
     course            ENUMERATED {success,fail},
     cause             CAUSE }                  -- cause if "course = fail"
```

NOTE 1 – The handling of Link Characteristics and SSCS Information is defined in Annex C.

NOTE 2 – Connection admission control and resource reservations are not specified in detail in this Recommendation.

The function "**SelectRoute**" performs a routing decision followed by the resource reservation on the outgoing link (the succeeding link) during connection establishment.

The nodal function analyses the routing information and selects a route with sufficient AAL type 2 path resources on a path with the requested path type to the succeeding AAL type 2 node. It then selects an AAL type 2 path from within that route which is able to accommodate the new connection.

Routing typically is based on:

- Addressing information;
- The Test Connection Indicator;
- Link Characteristics;
- Requested Path Type; and
- SSCS information (if Link Characteristics are not present).

This function returns a value that is defined by the following ASN.1 structure:

```
Route ::= SEQUENCE {                               -- return value of procedure
                                                   -- SelectRoute
     course             ENUMERATED {remote,local,fail},
     ceid               CEID,                      -- connection element
                                                   -- identifier
     sri                PID,                       -- nodal signalling association
                                                   -- identifier
     cause              CAUSE }                    -- cause if "fail"
```

NOTE 3 – The handling of Link Characteristics and SSCS Information is defined in Annex C.

The function "**SwitchRoute**" performs a routing decision followed by the resource reservation inside an AAL type 2 node.

This route is established between the requesting AAL type 2 served user or the incoming (preceeding) link on the one hand and the destination AAL type 2 served user or the outgoing link (succeeding) link on the other hand during connection establishment. It returns a value that is defined by the following ASN.1 structure:

```
Switch ::= SEQUENCE {                         -- return value of procedure
                                              -- SwitchRoute
     course             ENUMERATED {success,fail},
     cause              CAUSE }               -- cause if "fail"
```

NOTE 4 – The handling of Link Characteristics and SSCS Information is defined in Annex C.

NOTE 5 – Not all implementations require such internal path reservations; in such cases, "SwitchRes" is a null function that always returns the value "success".

The following procedures are defined in addition to those in ITU-T Rec. Q.2630.1.

The function "**SwitchRes**" performs a resource reservation inside an AAL type 2 node during resource modification. The resources reservation involves the path between the requesting AAL type 2 served user or the incoming link on the one hand and the destination AAL type 2 served user or the outgoing link on the other hand. It returns a value that is defined by the ASN.1 structure "Switch" above.

NOTE 6 – The handling of Link Characteristics and SSCS Information is defined in Annex C.

NOTE 7 – Not all implementations require such internal path reservations; in such cases, "SwitchRes" is a null function that always returns the value "success".

The procedure "**PathAdj**" is used to adjust resources on an incoming or outgoing link after situations where the "most demanding" resources had to be reserved during connection establishment or resource modification (see C.1).

The procedure "**SwitchAdj**" is used to adjust resources inside an AAL type 2 node after situations where the "most demanding" resources had to be reserved during connection establishment or resource modification (see C.1).

NOTE 8 – Not all implementations require such internal path reservations; in such cases, "SwitchAdj" is a null procedure without functionality.

### D.4.4 Data structures of AAL type 2 signalling messages and parameters

The SDL diagrams make use of the following ASN.1 structure and definition of the AAL type 2 signalling messages and parameters:

### D.4.4.1 General message and parameter structure

B.4.4.1/Q.2630.1 applies. Further, the following specifications need to be added:

```
mod          MessageID ::= ‘00001110’H          -- Modify request
moa          MessageID ::= ‘00001100’H          -- Modify acknoledgement
mor          MessageID ::= ‘00001101’H          -- Modify reject
```

### D.4.4.2 Detailed parameter structure

B.4.4.2/Q.2630.1 applies. Further, the following specifications need to be added:

```
-- The following parameters are handled but never interpreted in the SDL
-- definition, hence, no details are needed
PT ::= OCTET STRING (SIZE (1))         -- definition of the essentials of the
                                       -- PT parameter
SSIAE ::= OCTET STRING (SIZE (8))      -- definition of the essentials of the
                                       -- SSIAE parameter
SSIME ::= OCTET STRING (SIZE (3))      -- definition of the essentials of the
                                       -- SSIME parameter
LC ::= OCTET STRING (SIZE (12))        -- definition of the essentials of the
                                       -- LC parameter
PLC ::= OCTET STRING (SIZE (12))       -- definition of the essentials of the
                                       -- PLC parameter
MSLC ::= OCTET STRING (SIZE (0))       -- definition of the essentials of the
                                       -- MSLC parameter
MSSSI ::= OCTET STRING (SIZE (0))      -- definition of the essentials of the
                                       -- MSSSI parameter
SUCI ::= OCTET STRING (SIZE (1))       -- definition of the essentials of the
                                       -- SUCI parameter
```

The parameter "ALC" is replaced by parameter "LC", therefore, the definition for "ALC" shall be deleted.

### D.4.4.3 Detailed parameter list structure for the messages

B.4.4.3/Q.2630.1 applies. Further, the following modifications are required:

Replace the definition of the ERQ message with the following definition:

```
ERQmsg ::= SEQUENCE {   -- definition of the essentials of the ERQ message
     ceid               CEID,              -- connection element identifier
     a2ea   CHOICE {                       -- AAL type 2 endpoint address
            esea        ESEA,              -- destination E.164 endpoint
                                           -- address
            nsea        NSEA },            -- destination NSAP endpoint
                                           -- address
     lc                 LC OPTIONAL,       -- link characteristics
     plc                LC OPTIONAL,       -- preferred link characteristics
     mslc               MSLC OPTIONAL,     -- modify support for link
                                           -- characteristics
     msssi              MSSSI OPTIONAL,    -- modify support for service
                                           -- specific info.
     osaid              OSAID,             -- originating signalling
                                           -- association ID
     pt                 PT OPTIONAL,       -- path type
     sugr               SUGR OPTIONAL,     -- served user generated reference
     sut                SUT OPTIONAL,      -- serverd user transport
     ssis               SSIS OPTIONAL,     -- SSCS information
     pssis              PSSIS OPTIONAL,    -- preferred SSCS information
     tci                TCI OPTIONAL }     -- test connection indicator
```

Replace the definition of the ECF message with the following definition:

```
ECFmsg ::= SEQUENCE {   -- definition of the essentials of the ECF message
     mslc               MSLC OPTIONAL,      -- modify support for link
                                            -- characteristics
     msssi              MSSSI OPTIONAL,     -- modify support for service
                                            -- specific info.
     osaid              OSAID }             -- originating signalling
                                            -- association ID
```

Replace the definition of the RLC message with the following definition:

```
RLCmsg ::= SEQUENCE {   -- definition of the essentials of the RLC message
     cause              CAU OPTIONAL,       -- cause
     ceid               CEID OPTIONAL }     -- connection element identifier
```

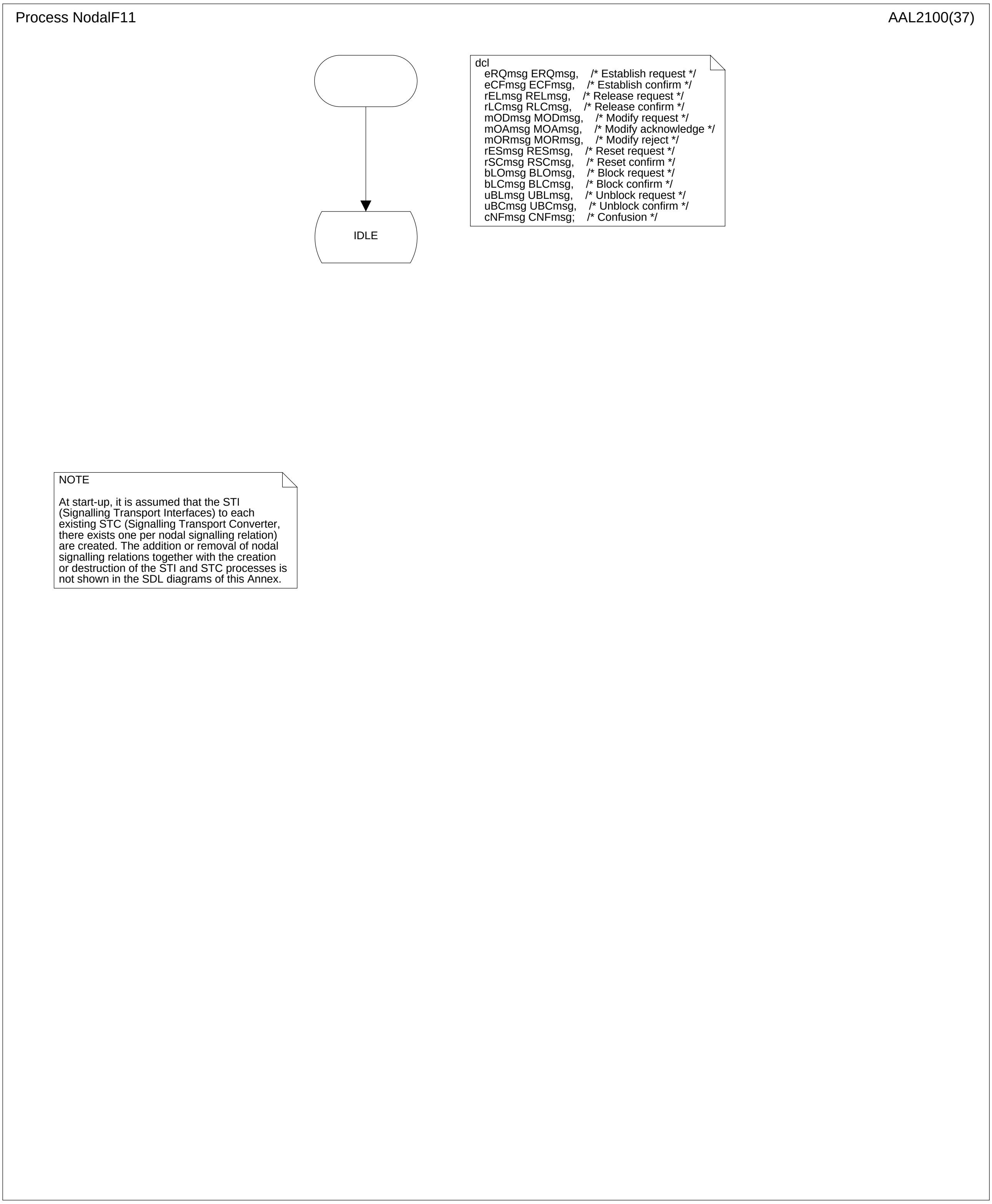
Add the definition of the MOD, MOA, and MOR messages as follows:

```
MODmsg ::= SEQUENCE {        -- definition of the essentials of the MOD message
     lc                      LC OPTIONAL,        -- link characteristics
     ssis   CHOICE {                             -- SSCS information
     ssiae                   SSIAE,              -- service specific information
                                                 -- (audio ext.)
     ssime                   SSIAE } OPTIONAL,   -- service specific info.
                                                 -- (multirate ext.)
     suci                    SUCI OPTIONAL }     -- served user correlation ID
```

```
MOAmsg ::= SEQUENCE {        -- definition of the essentials of the MOA message
     suci                    SUCI OPTIONAL }      -- served user correlation ID
```

```
MORmsg ::= SEQUENCE {   -- definition of the essentials of the MOR message
     cause              CAU }               -- cause
```

NOTE – The optionality of the parameters of ERQ and MOD messages is indicated in Table 7-6/Q.2630.2 and specified in 8.2.1.1.1.1 and 8.2.1.1.5 as well as Annex C.

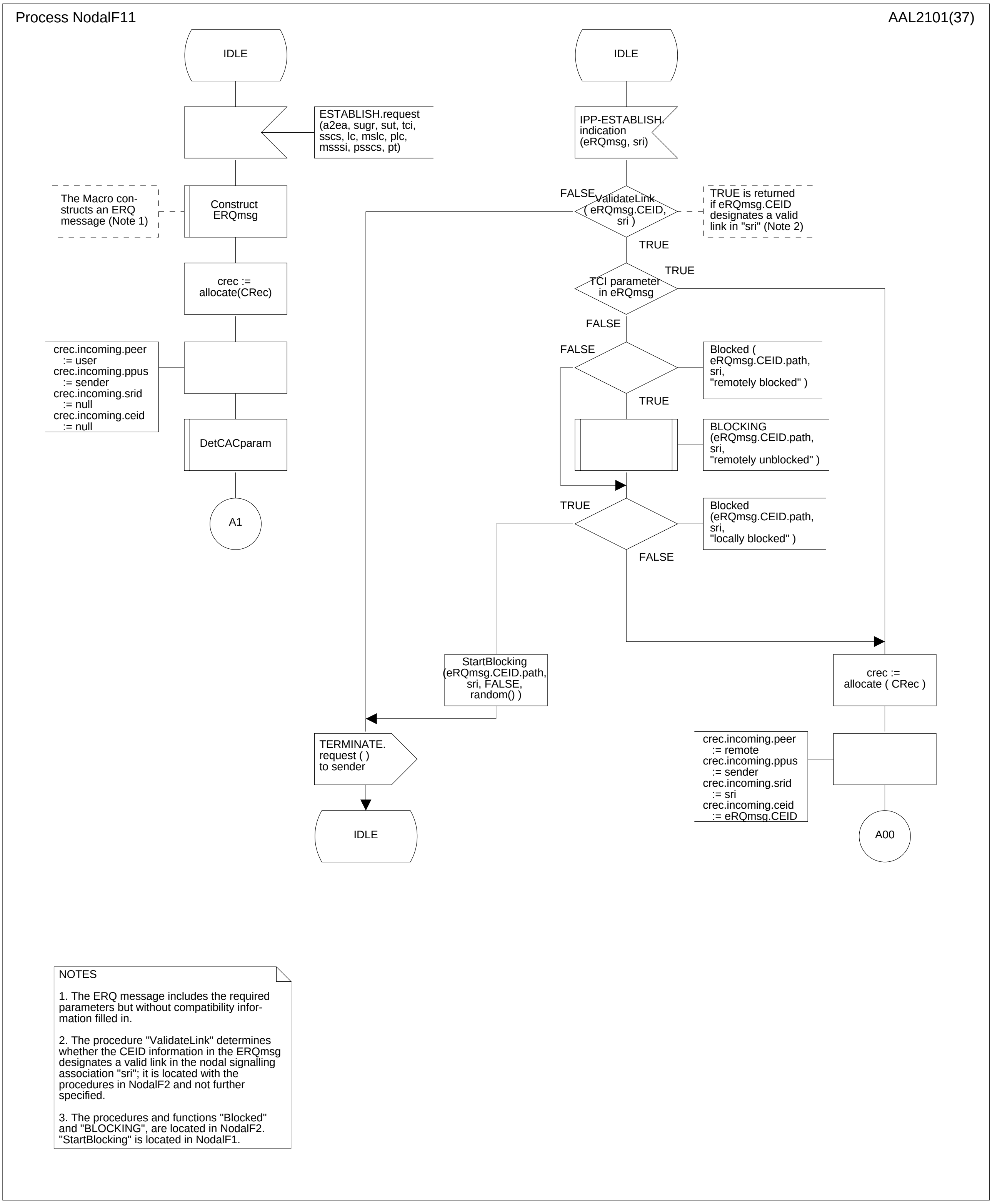


```
dcl
   eRQmsg ERQmsg,    /* Establish request */
   eCFmsg ECFmsg,    /* Establish confirm */
   rELmsg RELmsg,    /* Release request */
   rLCmsg RLCmsg,    /* Release confirm */
   mODmsg MODmsg,    /* Modify request */
   mOAmsg MOAmsg,    /* Modify acknowledge */
   mORmsg MORmsg,    /* Modify reject */
   rESmsg RESmsg,    /* Reset request */
   rSCmsg RSCmsg,    /* Reset confirm */
   bLOmsg BLOmsg,    /* Block request */
   bLCmsg BLCmsg,    /* Block confirm */
   uBLmsg UBLmsg,    /* Unblock request */
   uBCmsg UBCmsg,    /* Unblock confirm */
   cNFmsg CNFmsg;    /* Confusion */
```

NOTE

At start-up, it is assumed that the STI (Signalling Transport Interfaces) to each existing STC (Signalling Transport Converter, there exists one per nodal signalling relation) are created. The addition or removal of nodal signalling relations together with the creation or destruction of the STI and STC processes is not shown in the SDL diagrams of this Annex.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1** ***(part 1 of 37)***


Process NodalF11
AAL2101(37)
IDLE
ESTABLISH.request (a2ea, sugr, sut, tci, sscs, lc, mslc, plc, msssi, psscs, pt)
The Macro constructs an ERQ message (Note 1)
Construct ERQmsg
crec := allocate(CRec)
crec.incoming.peer := user
crec.incoming.ppus := sender
crec.incoming.srid := null
crec.incoming.ceid := null
DetCACparam
A1
IDLE
IPP-ESTABLISH. indication (eRQmsg, sri)
FALSE
ValidateLink ( eRQmsg.CEID, sri )
TRUE is returned if eRQmsg.CEID designates a valid link in "sri" (Note 2)
TRUE
TRUE
TCI parameter in eRQmsg
FALSE
FALSE
Blocked ( eRQmsg.CEID.path, sri, "remotely blocked" )
TRUE
BLOCKING (eRQmsg.CEID.path, sri, "remotely unblocked" )
TRUE
Blocked (eRQmsg.CEID.path, sri, "locally blocked" )
FALSE
StartBlocking (eRQmsg.CEID.path, sri, FALSE, random() )
crec := allocate ( CRec )
TERMINATE. request ( ) to sender
IDLE
crec.incoming.peer := remote
crec.incoming.ppus := sender
crec.incoming.srid := sri
crec.incoming.ceid := eRQmsg.CEID
A00

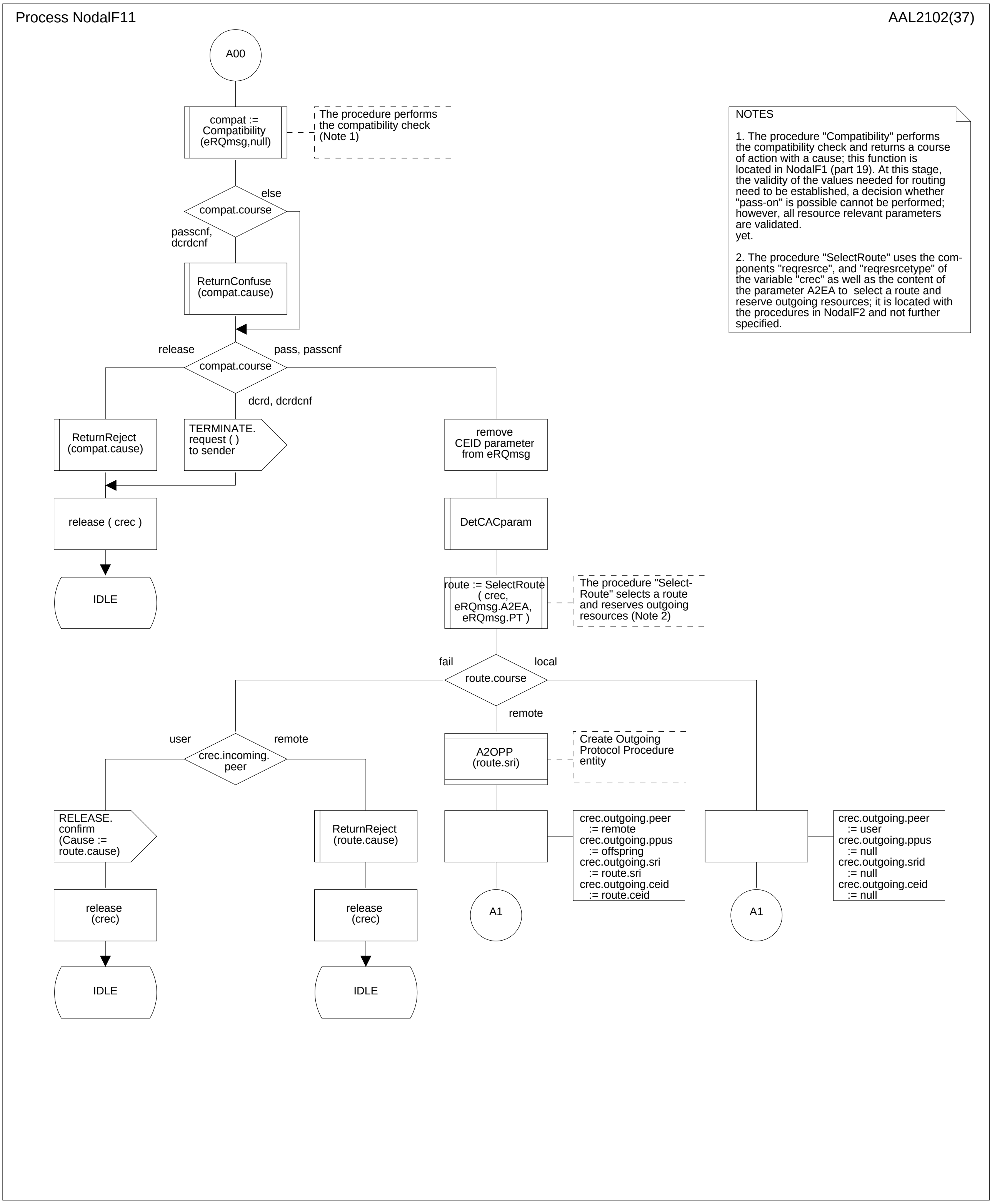
NOTES

1. The ERQ message includes the required parameters but without compatibility information filled in.

2. The procedure "ValidateLink" determines whether the CEID information in the ERQmsg designates a valid link in the nodal signalling association "sri"; it is located with the procedures in NodalF2 and not further specified.

3. The procedures and functions "Blocked" and "BLOCKING", are located in NodalF2. "StartBlocking" is located in NodalF1.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 2 of 37)***

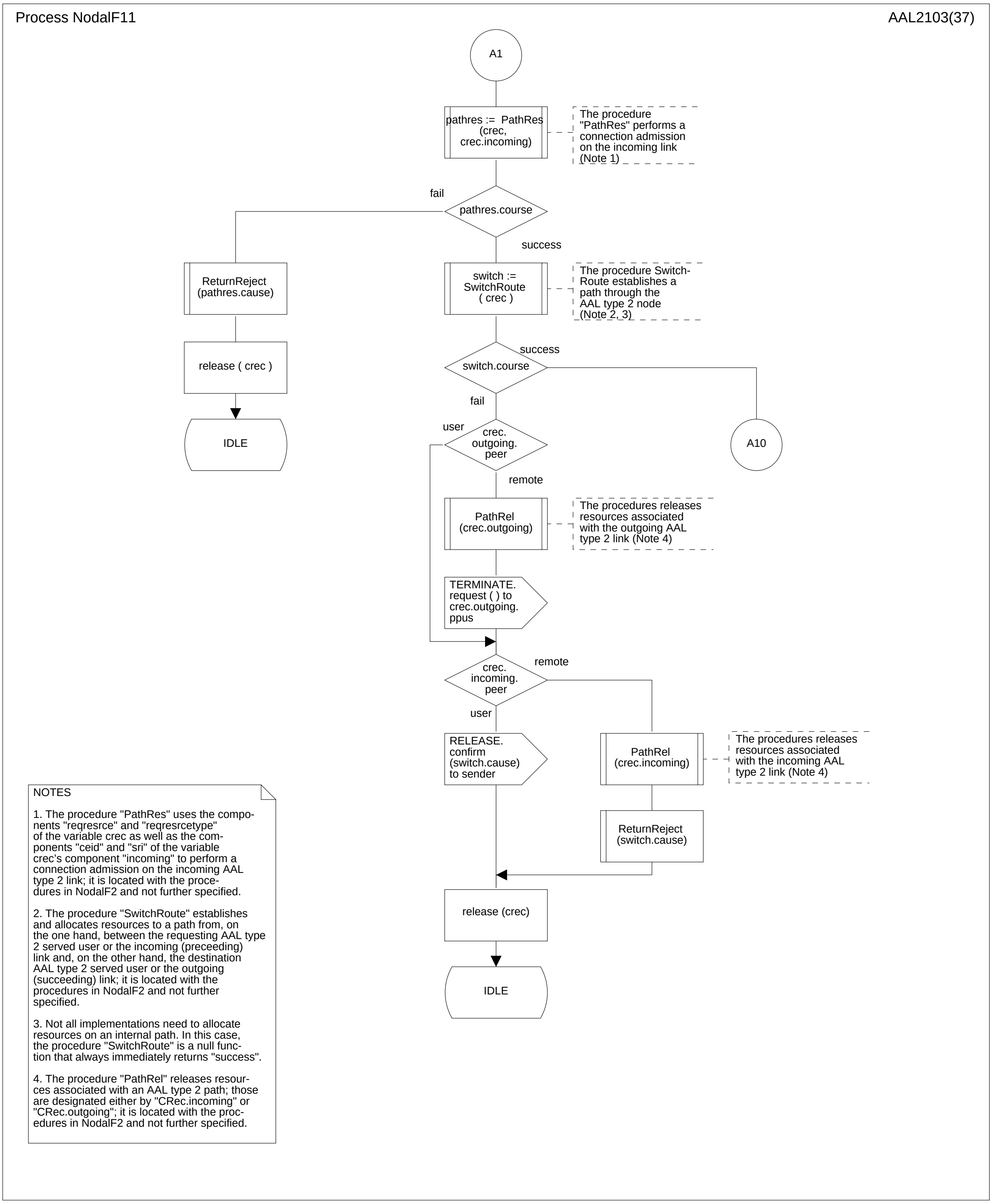

Process NodalF11
AAL2102(37)
A00
compat := Compatibility (eRQmsg,null)
The procedure performs the compatibility check (Note 1)
compat.course
else
passcnf, dcrdcnf
ReturnConfuse (compat.cause)
release
pass, passcnf
compat.course
dcrd, dcrdcnf
ReturnReject (compat.cause)
TERMINATE. request ( ) to sender
remove CEID parameter from eRQmsg
release ( crec )
DetCACparam
IDLE
route := SelectRoute ( crec, eRQmsg.A2EA, eRQmsg.PT )
The procedure "Select-Route" selects a route and reserves outgoing resources (Note 2)
fail
local
route.course
remote
user
remote
crec.incoming. peer
A2OPP (route.sri)
Create Outgoing Protocol Procedure entity
RELEASE. confirm (Cause := route.cause)
ReturnReject (route.cause)
crec.outgoing.peer := remote
crec.outgoing.ppus := offspring
crec.outgoing.sri := route.sri
crec.outgoing.ceid := route.ceid
crec.outgoing.peer := user
crec.outgoing.ppus := null
crec.outgoing.srid := null
crec.outgoing.ceid := null
release (crec)
release (crec)
A1
A1
IDLE
IDLE

NOTES

1. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). At this stage, the validity of the values needed for routing need to be established, a decision whether "pass-on" is possible cannot be performed; however, all resource relevant parameters are validated.
yet.

2. The procedure "SelectRoute" uses the components "reqresrce", and "reqresrcetype" of the variable "crec" as well as the content of the parameter A2EA to select a route and reserve outgoing resources; it is located with the procedures in NodalF2 and not further specified.

Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 3 of 37)*


Process NodalF11
AAL2103(37)
A1
pathres := PathRes (crec, crec.incoming)
The procedure "PathRes" performs a connection admission on the incoming link (Note 1)
fail
pathres.course
success
ReturnReject (pathres.cause)
switch := SwitchRoute ( crec )
The procedure Switch-Route establishes a path through the AAL type 2 node (Note 2, 3)
release ( crec )
success
switch.course
fail
IDLE
user
crec. outgoing. peer
A10
remote
PathRel (crec.outgoing)
The procedures releases resources associated with the outgoing AAL type 2 link (Note 4)
TERMINATE. request ( ) to crec.outgoing. ppus
remote
crec. incoming. peer
user
RELEASE. confirm (switch.cause) to sender
PathRel (crec.incoming)
The procedures releases resources associated with the incoming AAL type 2 link (Note 4)
ReturnReject (switch.cause)
release (crec)
IDLE

NOTES

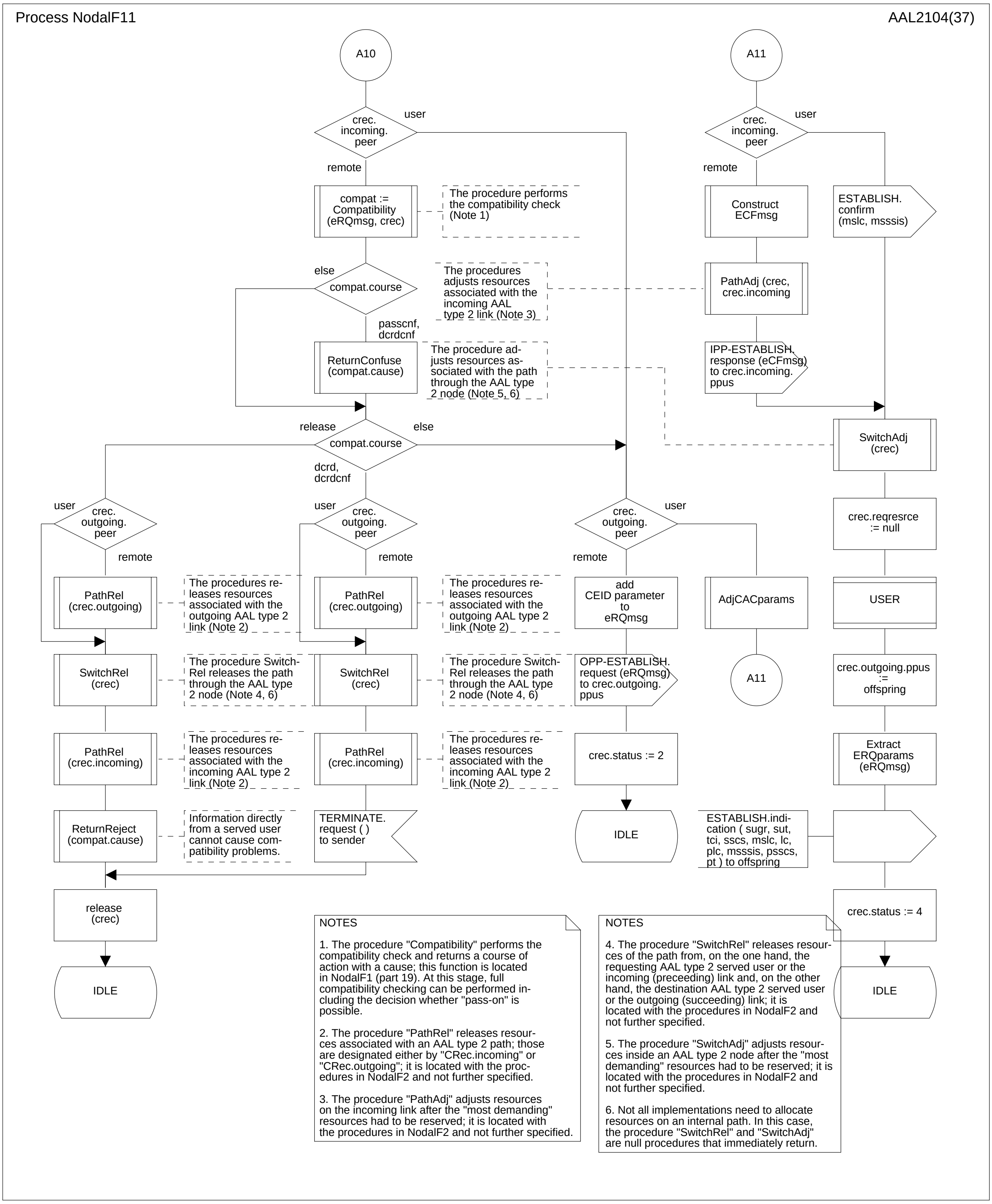
1. The procedure "PathRes" uses the components "reqresrce" and "reqresrcetype" of the variable crec as well as the components "ceid" and "sri" of the variable crec's component "incoming" to perform a connection admission on the incoming AAL type 2 link; it is located with the procedures in NodalF2 and not further specified.

2. The procedure "SwitchRoute" establishes and allocates resources to a path from, on the one hand, between the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

3. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRoute" is a null function that always immediately returns "success".

4. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 4 of 37)***



NOTES

1. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). At this stage, full compatibility checking can be performed including the decision whether "pass-on" is possible.

2. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified.

3. The procedure "PathAdj" adjusts resources on the incoming link after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

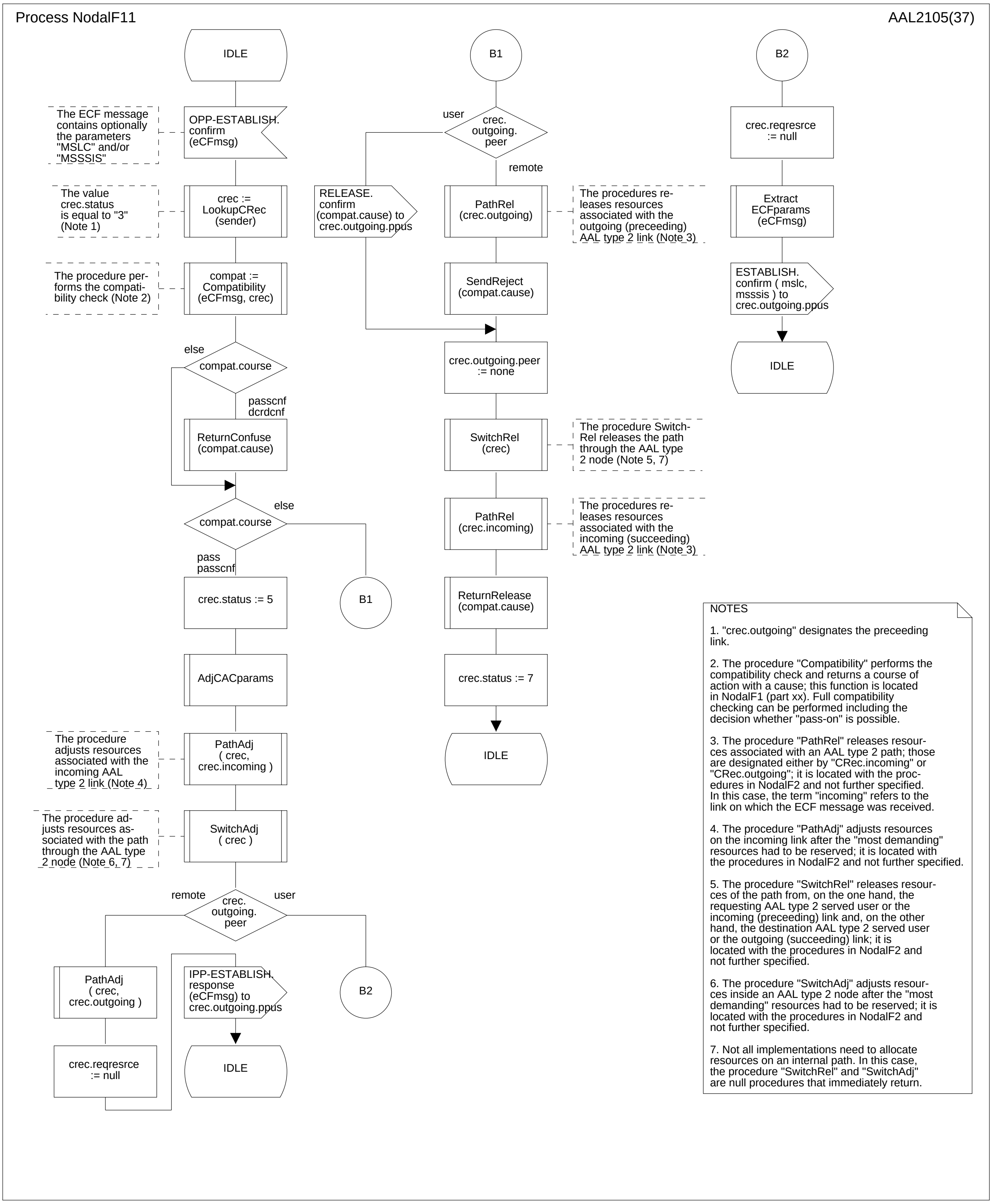
NOTES

4. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

5. The procedure "SwitchAdj" adjusts resources inside an AAL type 2 node after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

6. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" and "SwitchAdj" are null procedures that immediately return.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 5 of 37)***


Process NodalF11
AAL2105(37)
IDLE
B1
B2
The ECF message contains optionally the parameters "MSLC" and/or "MSSSIS"
OPP-ESTABLISH. confirm (eCFmsg)
The value crec.status is equal to "3" (Note 1)
crec := LookupCRec (sender)
The procedure performs the compatibility check (Note 2)
compat := Compatibility (eCFmsg, crec)
else
compat.course
passcnf dcrdcnf
ReturnConfuse (compat.cause)
else
compat.course
pass passcnf
crec.status := 5
B1
AdjCACparams
The procedure adjusts resources associated with the incoming AAL type 2 link (Note 4)
PathAdj ( crec, crec.incoming )
The procedure adjusts resources associated with the path through the AAL type 2 node (Note 6, 7)
SwitchAdj ( crec )
remote
user
crec. outgoing. peer
PathAdj ( crec, crec.outgoing )
crec.reqresrce := null
IPP-ESTABLISH. response (eCFmsg) to crec.outgoing.ppus
IDLE
B2
user
crec. outgoing. peer
remote
RELEASE. confirm (compat.cause) to crec.outgoing.ppus
PathRel (crec.outgoing)
The procedures releases resources associated with the outgoing (preceeding) AAL type 2 link (Note 3)
SendReject (compat.cause)
crec.outgoing.peer := none
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 5, 7)
PathRel (crec.incoming)
The procedures releases resources associated with the incoming (succeeding) AAL type 2 link (Note 3)
ReturnRelease (compat.cause)
crec.status := 7
IDLE
crec.reqresrce := null
Extract ECFparams (eCFmsg)
ESTABLISH. confirm ( mslc, msssis ) to crec.outgoing.ppus
IDLE

NOTES

1. "crec.outgoing" designates the preceeding link.

2. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part xx). Full compatibility checking can be performed including the decision whether "pass-on" is possible.

3. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the ECF message was received.

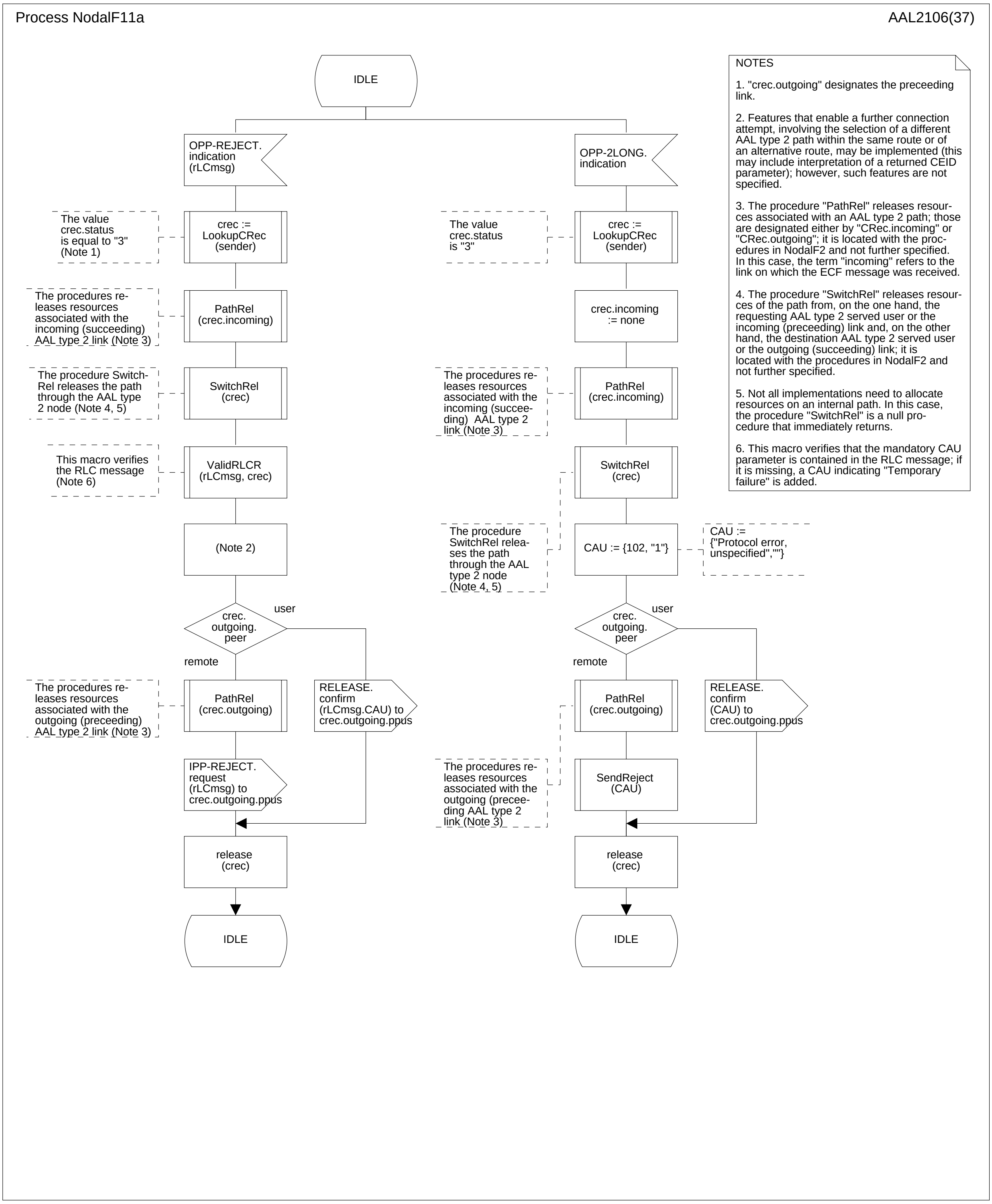
4. The procedure "PathAdj" adjusts resources on the incoming link after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

5. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

6. The procedure "SwitchAdj" adjusts resources inside an AAL type 2 node after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

7. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" and "SwitchAdj" are null procedures that immediately return.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 6 of 37)***


Process NodalF11a
AAL2106(37)
IDLE
OPP-REJECT. indication (rLCmsg)
OPP-2LONG. indication
The value crec.status is equal to "3" (Note 1)
crec := LookupCRec (sender)
The value crec.status is "3"
crec := LookupCRec (sender)
The procedures releases resources associated with the incoming (succeeding) AAL type 2 link (Note 3)
PathRel (crec.incoming)
crec.incoming := none
The procedure SwitchRel releases the path through the AAL type 2 node (Note 4, 5)
SwitchRel (crec)
The procedures releases resources associated with the incoming (succeeding) AAL type 2 link (Note 3)
PathRel (crec.incoming)
This macro verifies the RLC message (Note 6)
ValidRLCR (rLCmsg, crec)
SwitchRel (crec)
(Note 2)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 4, 5)
CAU := {102, "1"}
CAU := {"Protocol error, unspecified",""}
crec. outgoing. peer
user
remote
crec. outgoing. peer
user
remote
The procedures releases resources associated with the outgoing (preceeding) AAL type 2 link (Note 3)
PathRel (crec.outgoing)
RELEASE. confirm (rLCmsg.CAU) to crec.outgoing.ppus
PathRel (crec.outgoing)
RELEASE. confirm (CAU) to crec.outgoing.ppus
IPP-REJECT. request (rLCmsg) to crec.outgoing.ppus
The procedures releases resources associated with the outgoing (preceeding AAL type 2 link (Note 3)
SendReject (CAU)
release (crec)
release (crec)
IDLE
IDLE

NOTES

1. "crec.outgoing" designates the preceeding link.

2. Features that enable a further connection attempt, involving the selection of a different AAL type 2 path within the same route or of an alternative route, may be implemented (this may include interpretation of a returned CEID parameter); however, such features are not specified.

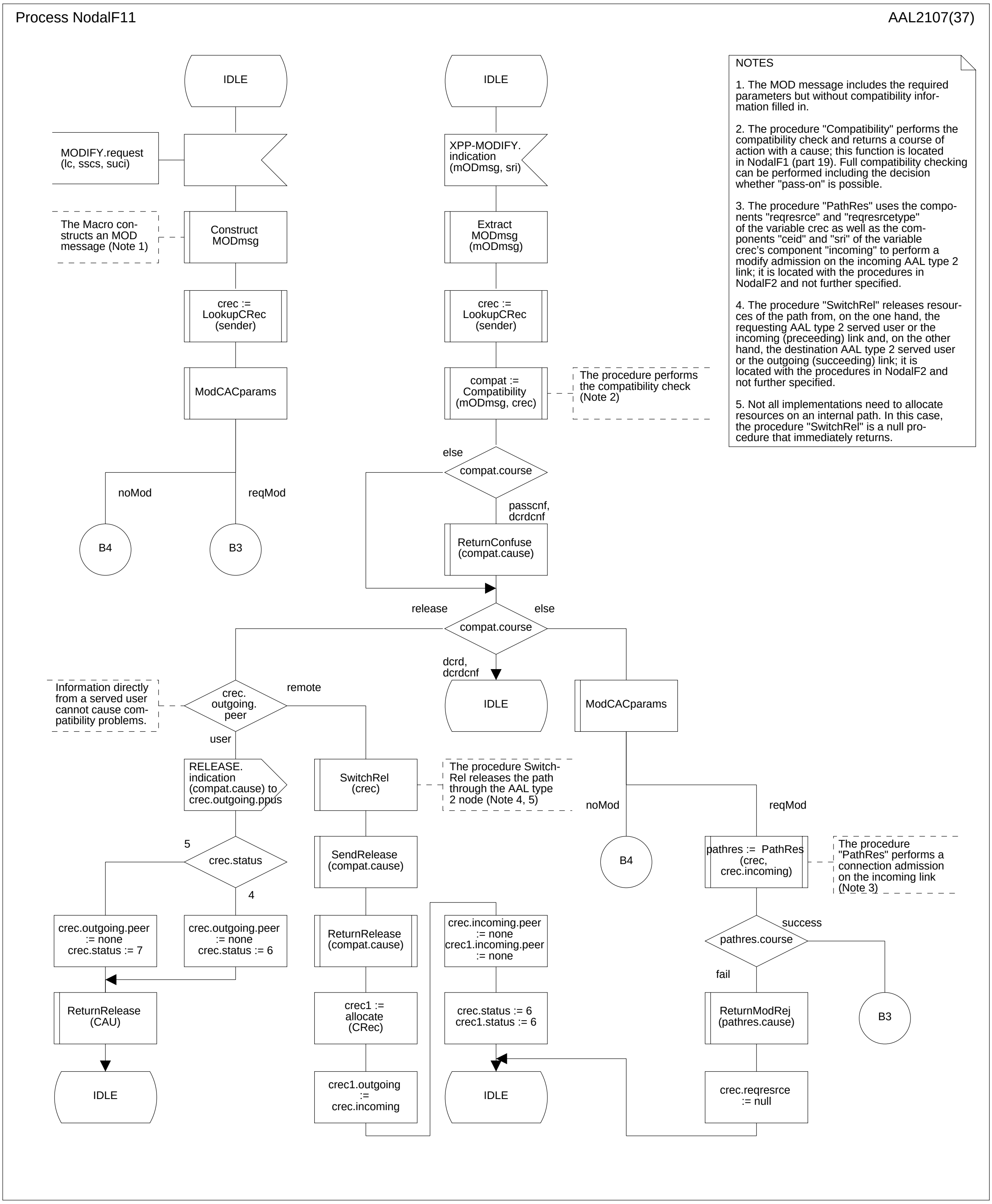
3. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the ECF message was received.

4. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

5. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

6. This macro verifies that the mandatory CAU parameter is contained in the RLC message; if it is missing, a CAU indicating "Temporary failure" is added.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 7 of 37)***


Process NodalF11
AAL2107(37)
IDLE
MODIFY.request (lc, sscs, suci)
The Macro constructs an MOD message (Note 1)
Construct MODmsg
crec := LookupCRec (sender)
ModCACparams
noMod
reqMod
B4
B3
IDLE
XPP-MODIFY. indication (mODmsg, sri)
Extract MODmsg (mODmsg)
crec := LookupCRec (sender)
compat := Compatibility (mODmsg, crec)
The procedure performs the compatibility check (Note 2)
else
compat.course
passcnf, dcrdcnf
ReturnConfuse (compat.cause)
release
else
compat.course
dcrd, dcrdcnf
IDLE
ModCACparams
Information directly from a served user cannot cause compatibility problems.
crec. outgoing. peer
remote
user
RELEASE. indication (compat.cause) to crec.outgoing.ppus
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 4, 5)
noMod
reqMod
5
crec.status
4
SendRelease (compat.cause)
B4
pathres := PathRes (crec, crec.incoming)
The procedure "PathRes" performs a connection admission on the incoming link (Note 3)
crec.outgoing.peer := none crec.status := 7
crec.outgoing.peer := none crec.status := 6
ReturnRelease (compat.cause)
crec.incoming.peer := none crec1.incoming.peer := none
success
pathres.course
fail
ReturnRelease (CAU)
crec1 := allocate (CRec)
crec.status := 6 crec1.status := 6
ReturnModRej (pathres.cause)
B3
IDLE
crec1.outgoing := crec.incoming
IDLE
crec.reqresrce := null

NOTES

1. The MOD message includes the required parameters but without compatibility information filled in.

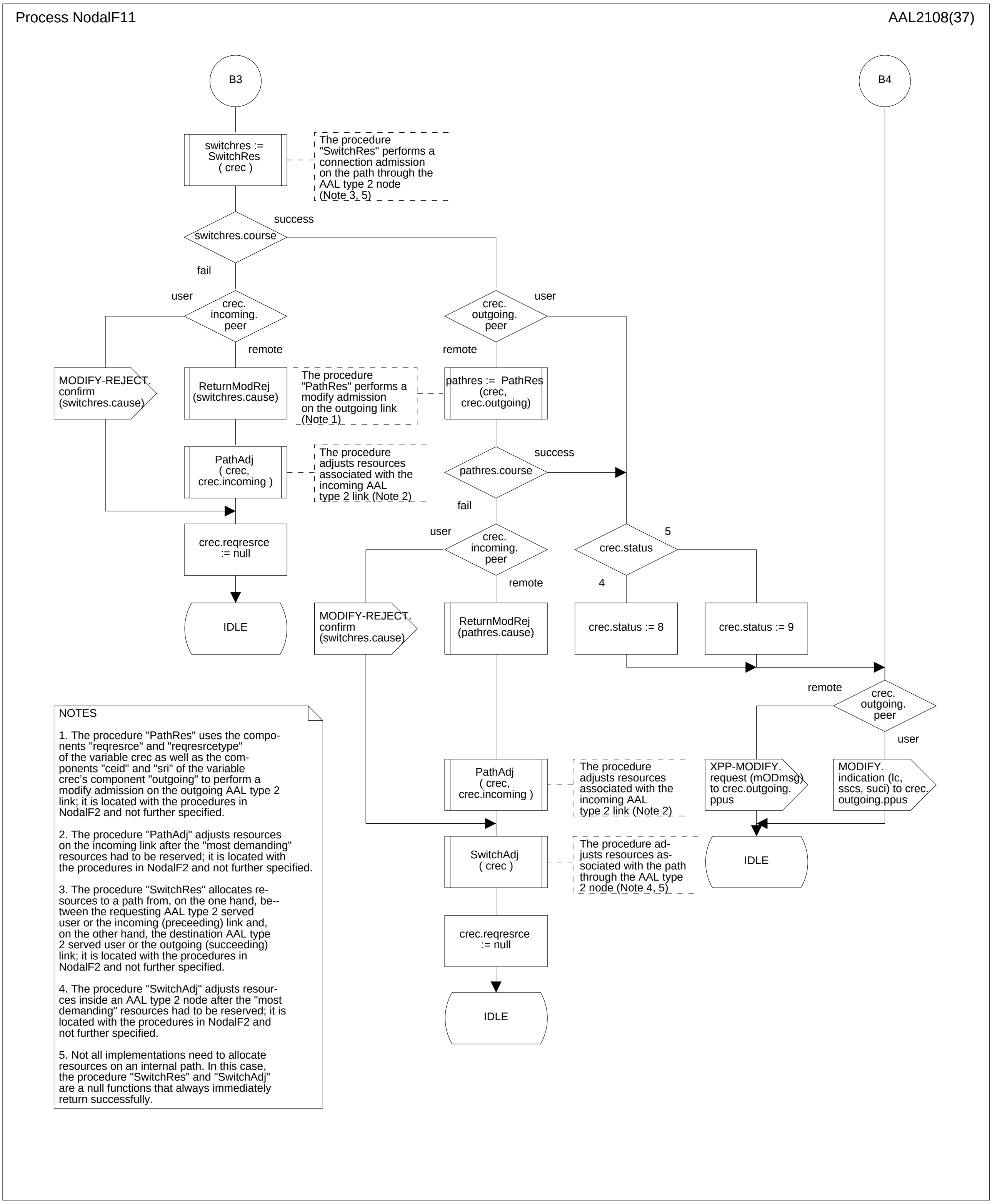
2. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). Full compatibility checking can be performed including the decision whether "pass-on" is possible.

3. The procedure "PathRes" uses the components "reqresrce" and "reqresrcetype" of the variable crec as well as the components "ceid" and "sri" of the variable crec's component "incoming" to perform a modify admission on the incoming AAL type 2 link; it is located with the procedures in NodalF2 and not further specified.

4. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

5. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 8 of 37)***


Process NodalF11
AAL2108(37)
B3
B4
switchres := SwitchRes ( crec )
The procedure "SwitchRes" performs a connection admission on the path through the AAL type 2 node (Note 3, 5)
switchres.course
success
fail
user
crec. incoming. peer
remote
crec. outgoing. peer
user
remote
MODIFY-REJECT. confirm (switchres.cause)
ReturnModRej (switchres.cause)
The procedure "PathRes" performs a modify admission on the outgoing link (Note 1)
pathres := PathRes (crec, crec.outgoing)
PathAdj ( crec, crec.incoming )
The procedure adjusts resources associated with the incoming AAL type 2 link (Note 2)
pathres.course
success
fail
crec.reqresrce := null
user
crec. incoming. peer
remote
crec.status
5
4
IDLE
MODIFY-REJECT. confirm (switchres.cause)
ReturnModRej (pathres.cause)
crec.status := 8
crec.status := 9
remote
crec. outgoing. peer
user
PathAdj ( crec, crec.incoming )
The procedure adjusts resources associated with the incoming AAL type 2 link (Note 2)
XPP-MODIFY. request (mODmsg) to crec.outgoing. ppus
MODIFY. indication (lc, sscs, suci) to crec. outgoing.ppus
SwitchAdj ( crec )
The procedure adjusts resources associated with the path through the AAL type 2 node (Note 4, 5)
IDLE
crec.reqresrce := null
IDLE
NOTES

NOTES

1. The procedure "PathRes" uses the components "reqresrce" and "reqresrcetype" of the variable crec as well as the components "ceid" and "sri" of the variable crec's component "outgoing" to perform a modify admission on the outgoing AAL type 2 link; it is located with the procedures in NodalF2 and not further specified.

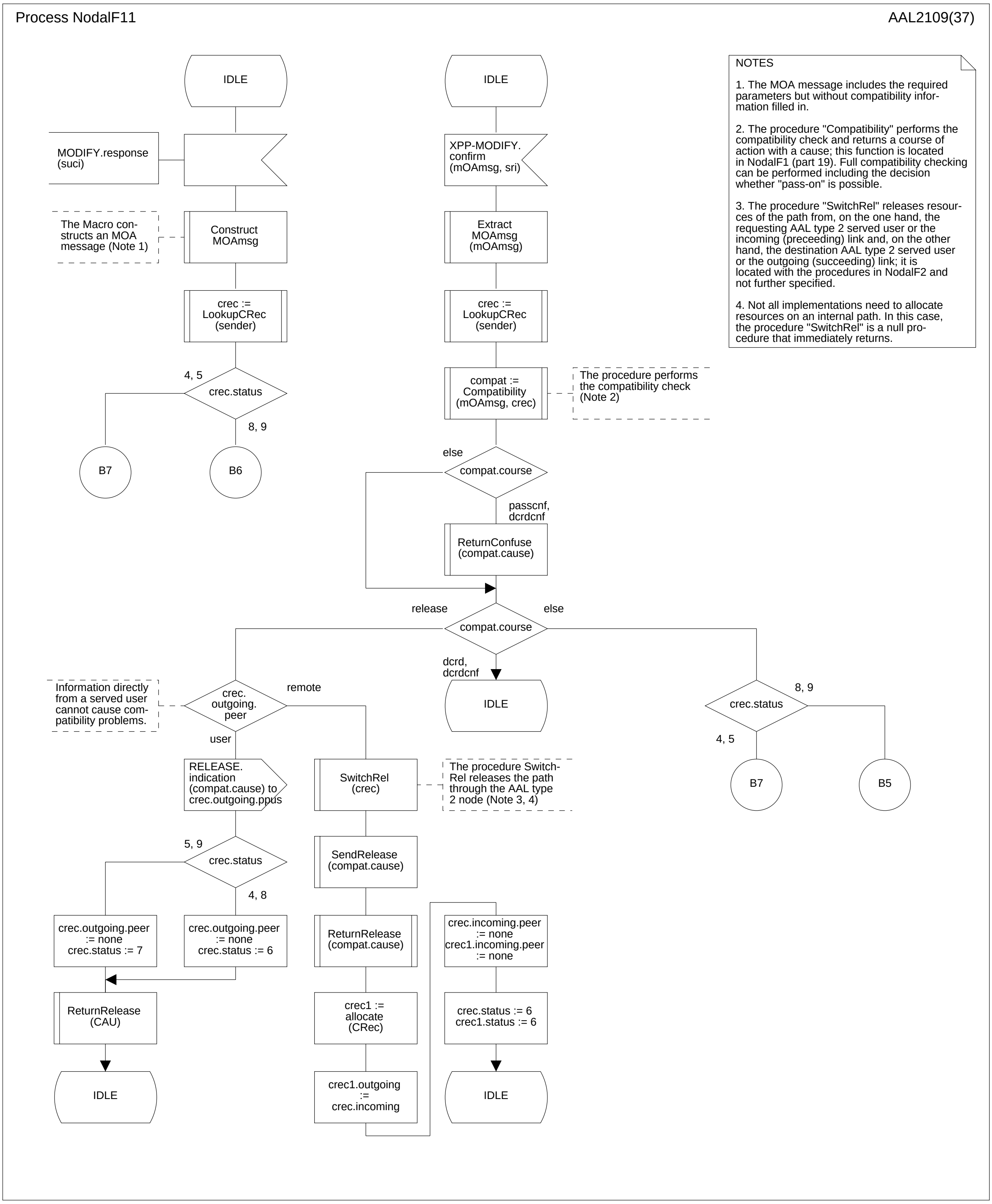
2. The procedure "PathAdj" adjusts resources on the incoming link after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

3. The procedure "SwitchRes" allocates resources to a path from, on the one hand, be--tween the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

4. The procedure "SwitchAdj" adjusts resources inside an AAL type 2 node after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

5. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRes" and "SwitchAdj" are a null functions that always immediately return successfully.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 9 of 37)***


Process NodalF11
AAL2109(37)
IDLE
MODIFY.response (suci)
The Macro constructs an MOA message (Note 1)
Construct MOAmsg
crec := LookupCRec (sender)
4, 5
crec.status
8, 9
B7
B6
IDLE
XPP-MODIFY. confirm (mOAmsg, sri)
Extract MOAmsg (mOAmsg)
crec := LookupCRec (sender)
compat := Compatibility (mOAmsg, crec)
The procedure performs the compatibility check (Note 2)
else
compat.course
passcnf, dcrdcnf
ReturnConfuse (compat.cause)
release
compat.course
else
dcrd, dcrdcnf
IDLE
Information directly from a served user cannot cause compatibility problems.
crec. outgoing. peer
remote
user
RELEASE. indication (compat.cause) to crec.outgoing.ppus
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 3, 4)
8, 9
crec.status
4, 5
B7
B5
5, 9
crec.status
4, 8
crec.outgoing.peer := none crec.status := 7
crec.outgoing.peer := none crec.status := 6
ReturnRelease (CAU)
IDLE
SendRelease (compat.cause)
ReturnRelease (compat.cause)
crec1 := allocate (CRec)
crec1.outgoing := crec.incoming
crec.incoming.peer := none crec1.incoming.peer := none
crec.status := 6 crec1.status := 6
IDLE

NOTES

1. The MOA message includes the required parameters but without compatibility information filled in.

2. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). Full compatibility checking can be performed including the decision whether "pass-on" is possible.

3. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

4. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1** ***(part 10 of 37)***

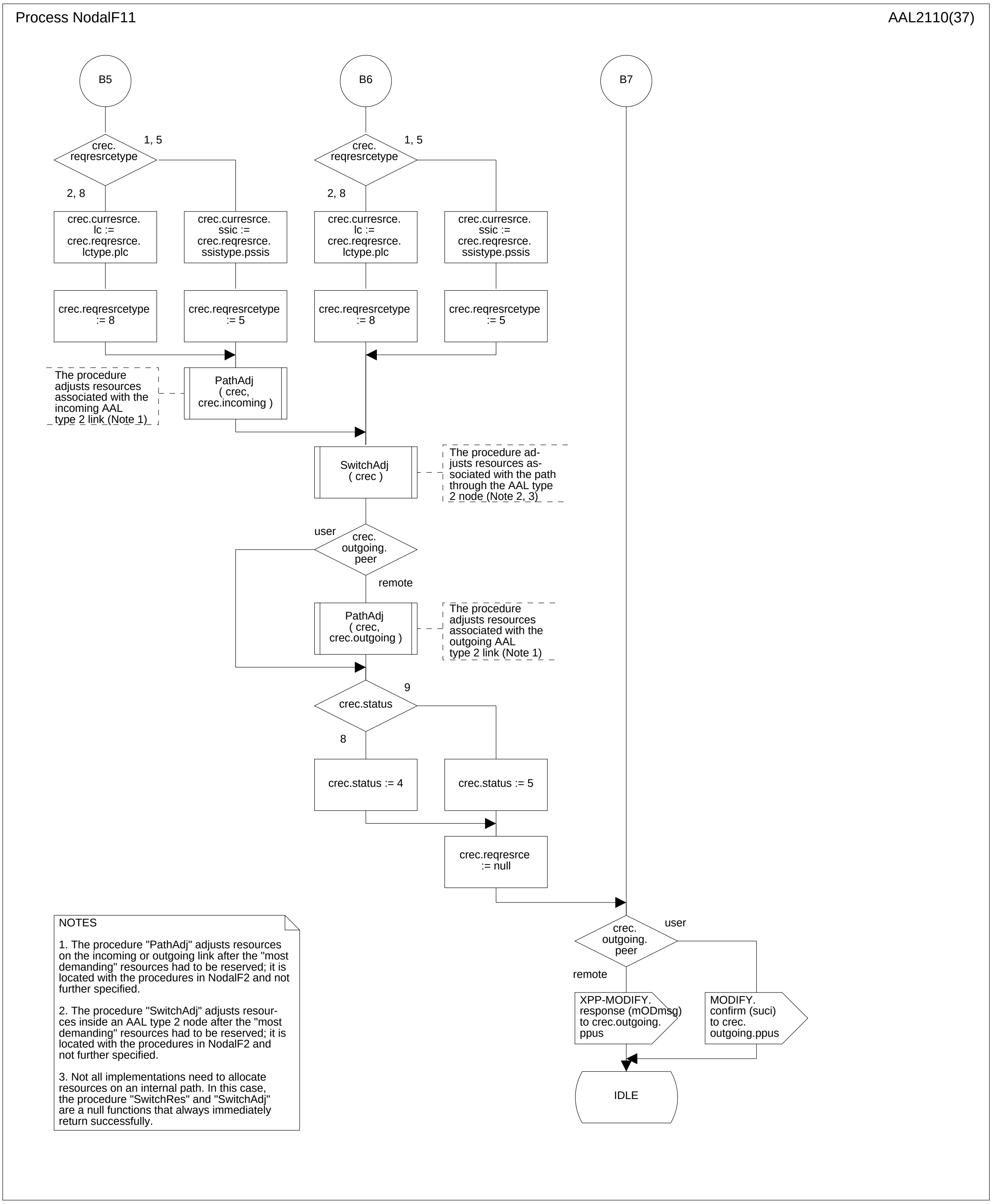
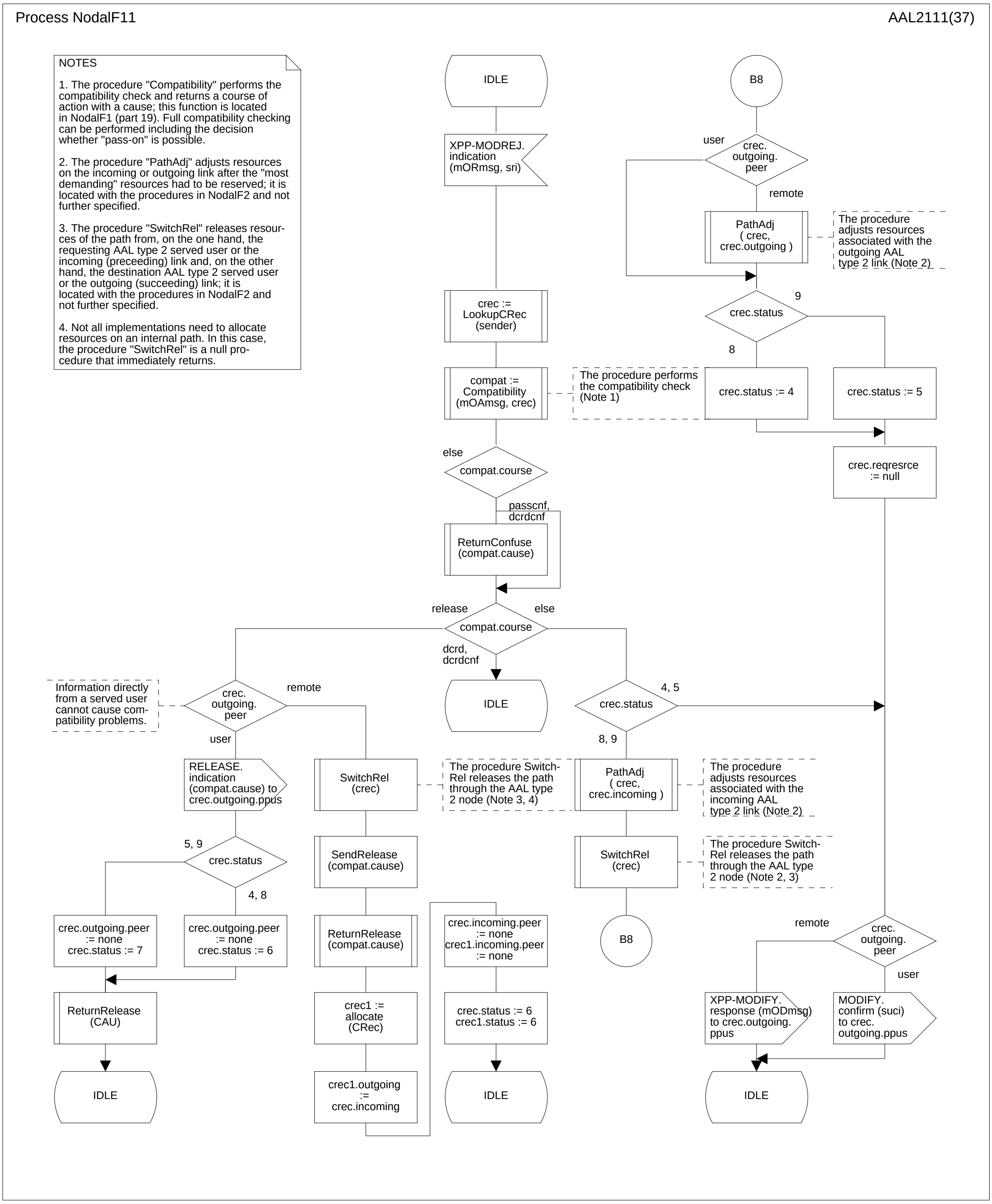


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 11 of 37)***



NOTES

1. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). Full compatibility checking can be performed including the decision whether "pass-on" is possible.

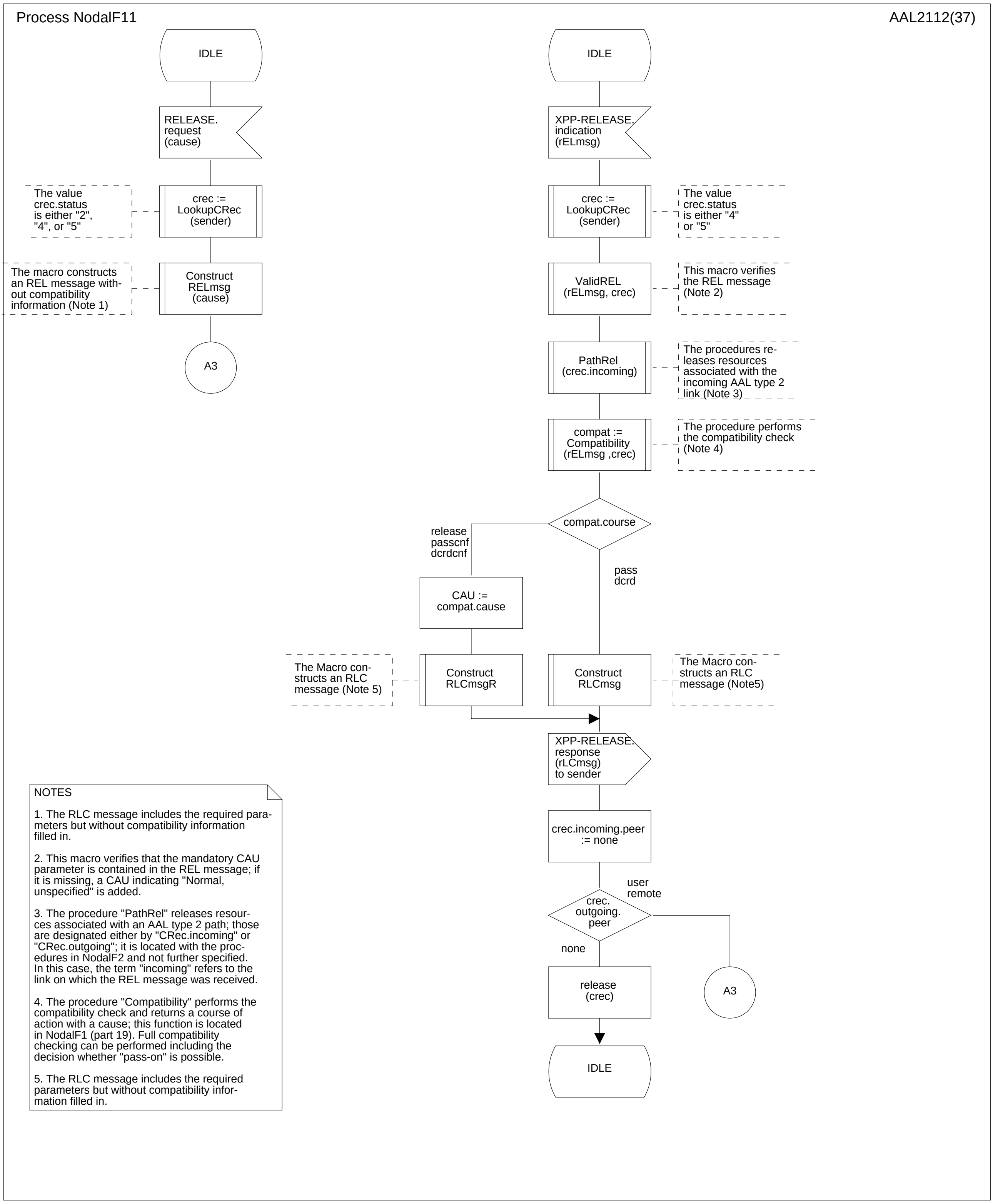
2. The procedure "PathAdj" adjusts resources on the incoming or outgoing link after the "most demanding" resources had to be reserved; it is located with the procedures in NodalF2 and not further specified.

3. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

4. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

Process NodalF11
AAL2111(37)
IDLE
XPP-MODREJ. indication (mORmsg, sri)
crec := LookupCRec (sender)
compat := Compatibility (mOAmsg, crec)
The procedure performs the compatibility check (Note 1)
else
compat.course
passcnf, dcrdcnf
ReturnConfuse (compat.cause)
release
else
compat.course
dcrd, dcrdcnf
IDLE
Information directly from a served user cannot cause compatibility problems.
crec. outgoing. peer
remote
user
RELEASE. indication (compat.cause) to crec.outgoing.ppus
5, 9
crec.status
4, 8
crec.outgoing.peer := none crec.status := 7
crec.outgoing.peer := none crec.status := 6
ReturnRelease (CAU)
IDLE
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 3, 4)
SendRelease (compat.cause)
ReturnRelease (compat.cause)
crec1 := allocate (CRec)
crec1.outgoing := crec.incoming
crec.incoming.peer := none crec1.incoming.peer := none
crec.status := 6 crec1.status := 6
IDLE
crec.status
4, 5
8, 9
PathAdj ( crec, crec.incoming )
The procedure adjusts resources associated with the incoming AAL type 2 link (Note 2)
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 2, 3)
B8
B8
user
crec. outgoing. peer
remote
PathAdj ( crec, crec.outgoing )
The procedure adjusts resources associated with the outgoing AAL type 2 link (Note 2)
9
crec.status
8
crec.status := 4
crec.status := 5
crec.reqresrce := null
remote
crec. outgoing. peer
user
XPP-MODIFY. response (mODmsg) to crec.outgoing. ppus
MODIFY. confirm (suci) to crec. outgoing.ppus
IDLE

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 12 of 37)***


Process NodalF11
AAL2112(37)
IDLE
RELEASE. request (cause)
crec := LookupCRec (sender)
The value crec.status is either "2", "4", or "5"
Construct RELmsg (cause)
The macro constructs an REL message with-out compatibility information (Note 1)
A3
IDLE
XPP-RELEASE. indication (rELmsg)
crec := LookupCRec (sender)
The value crec.status is either "4" or "5"
ValidREL (rELmsg, crec)
This macro verifies the REL message (Note 2)
PathRel (crec.incoming)
The procedures re-leases resources associated with the incoming AAL type 2 link (Note 3)
compat := Compatibility (rELmsg ,crec)
The procedure performs the compatibility check (Note 4)
compat.course
release passcnf dcrdcnf
pass dcrd
CAU := compat.cause
Construct RLCmsgR
The Macro con-structs an RLC message (Note 5)
Construct RLCmsg
The Macro con-structs an RLC message (Note5)
XPP-RELEASE. response (rLCmsg) to sender
crec.incoming.peer := none
crec. outgoing. peer
user remote
none
release (crec)
A3
IDLE

NOTES

1. The RLC message includes the required parameters but without compatibility information filled in.

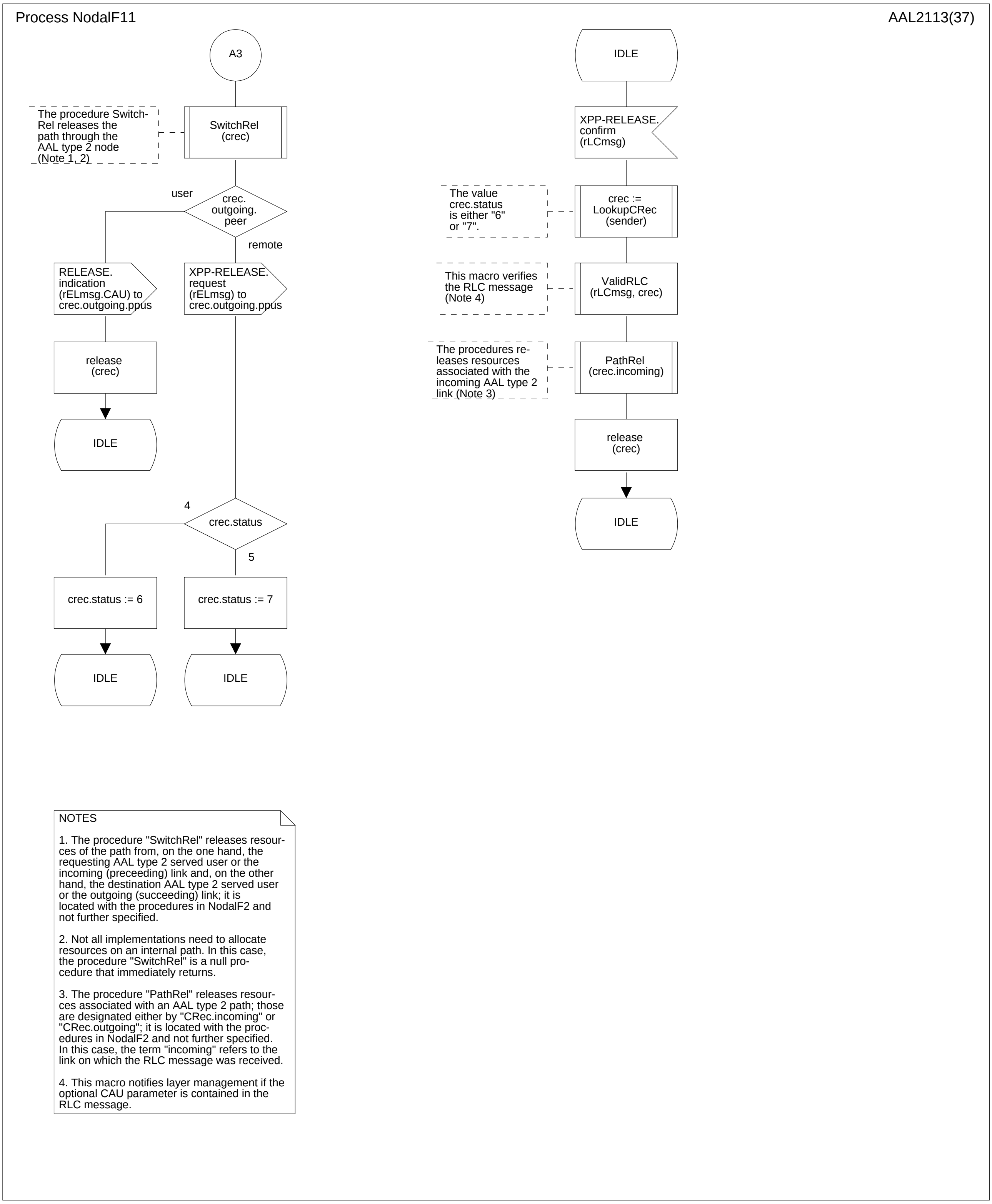
2. This macro verifies that the mandatory CAU parameter is contained in the REL message; if it is missing, a CAU indicating "Normal, unspecified" is added.

3. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the REL message was received.

4. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). Full compatibility checking can be performed including the decision whether "pass-on" is possible.

5. The RLC message includes the required parameters but without compatibility information filled in.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 13 of 37)***


Process NodalF11
AAL2113(37)
A3
SwitchRel (crec)
The procedure Switch-Rel releases the path through the AAL type 2 node (Note 1, 2)
crec. outgoing. peer
user
remote
RELEASE. indication (rELmsg.CAU) to crec.outgoing.ppus
XPP-RELEASE. request (rELmsg) to crec.outgoing.ppus
release (crec)
IDLE
4
crec.status
5
crec.status := 6
crec.status := 7
IDLE
IDLE
IDLE
XPP-RELEASE. confirm (rLCmsg)
The value crec.status is either "6" or "7".
crec := LookupCRec (sender)
This macro verifies the RLC message (Note 4)
ValidRLC (rLCmsg, crec)
The procedures re-leases resources associated with the incoming AAL type 2 link (Note 3)
PathRel (crec.incoming)
release (crec)
IDLE

NOTES

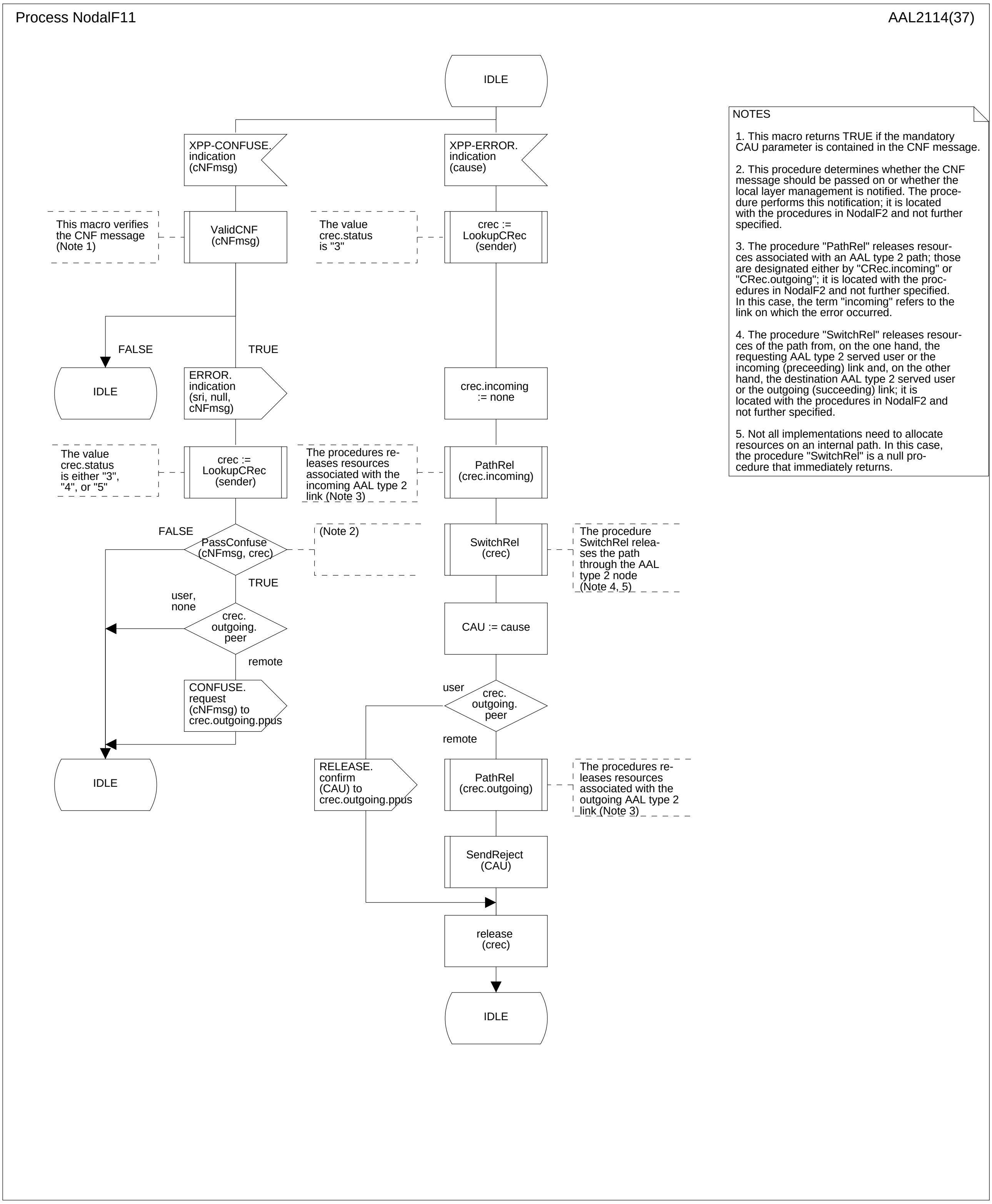
1. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

2. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

3. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the RLC message was received.

4. This macro notifies layer management if the optional CAU parameter is contained in the RLC message.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 14 of 37)***


Process NodalF11
AAL2114(37)
IDLE
XPP-CONFUSE. indication (cNFmsg)
XPP-ERROR. indication (cause)
This macro verifies the CNF message (Note 1)
ValidCNF (cNFmsg)
The value crec.status is "3"
crec := LookupCRec (sender)
FALSE
TRUE
IDLE
ERROR. indication (sri, null, cNFmsg)
crec.incoming := none
The value crec.status is either "3", "4", or "5"
crec := LookupCRec (sender)
The procedures re-leases resources associated with the incoming AAL type 2 link (Note 3)
PathRel (crec.incoming)
FALSE
PassConfuse (cNFmsg, crec)
(Note 2)
SwitchRel (crec)
The procedure SwitchRel relea-ses the path through the AAL type 2 node (Note 4, 5)
TRUE
user, none
crec. outgoing. peer
CAU := cause
remote
CONFUSE. request (cNFmsg) to crec.outgoing.ppus
user
crec. outgoing. peer
remote
IDLE
RELEASE. confirm (CAU) to crec.outgoing.ppus
PathRel (crec.outgoing)
The procedures re-leases resources associated with the outgoing AAL type 2 link (Note 3)
SendReject (CAU)
release (crec)
IDLE

NOTES

1. This macro returns TRUE if the mandatory CAU parameter is contained in the CNF message.

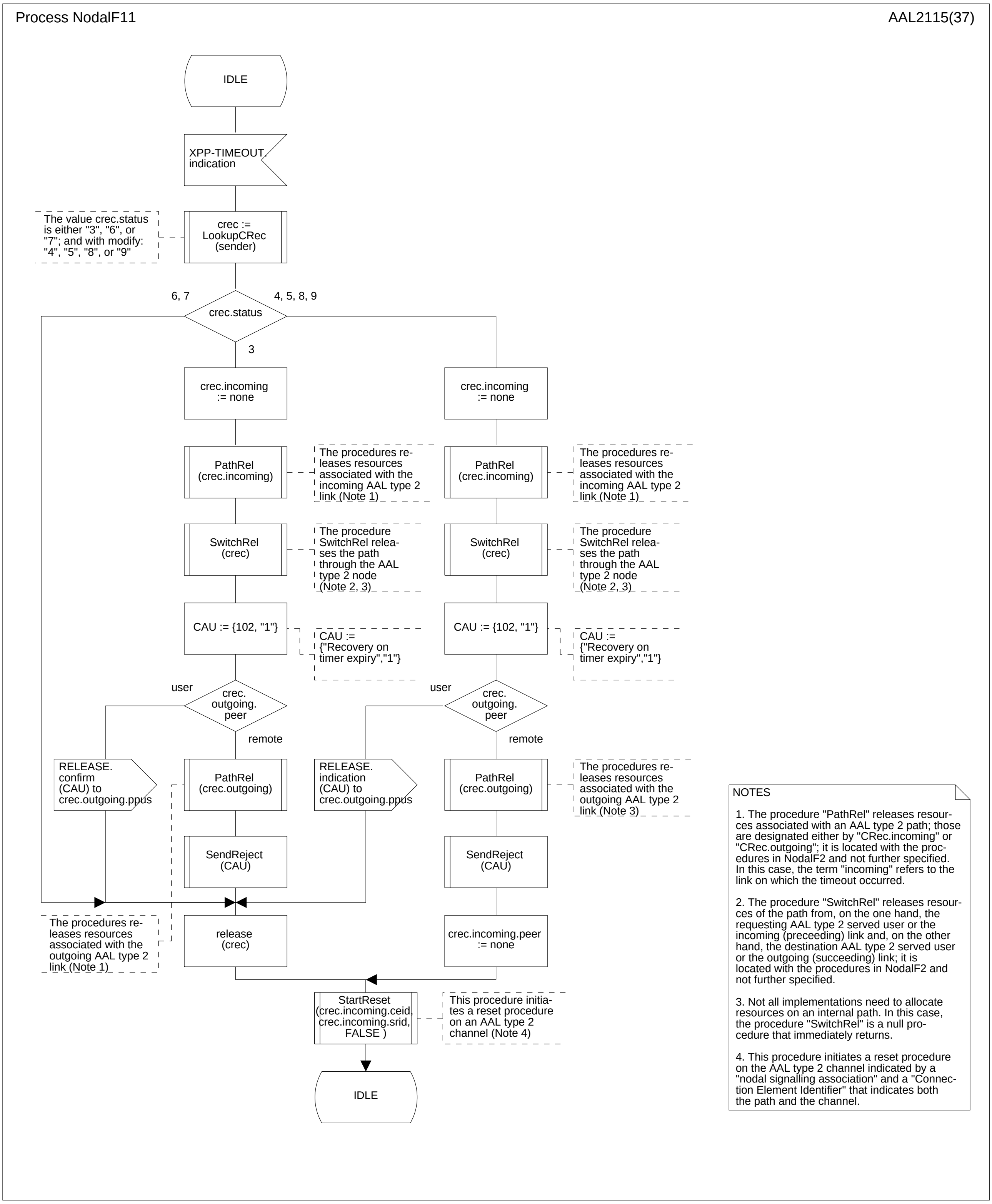
2. This procedure determines whether the CNF message should be passed on or whether the local layer management is notified. The procedure performs this notification; it is located with the procedures in NodalF2 and not further specified.

3. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the error occurred.

4. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

5. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 15 of 37)***


Process NodalF11
AAL2115(37)
IDLE
XPP-TIMEOUT. indication
The value crec.status is either "3", "6", or "7"; and with modify: "4", "5", "8", or "9"
crec := LookupCRec (sender)
6, 7
4, 5, 8, 9
crec.status
3
crec.incoming := none
crec.incoming := none
PathRel (crec.incoming)
The procedures releases resources associated with the incoming AAL type 2 link (Note 1)
PathRel (crec.incoming)
The procedures releases resources associated with the incoming AAL type 2 link (Note 1)
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 2, 3)
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 2, 3)
CAU := {102, "1"}
CAU := {"Recovery on timer expiry","1"}
CAU := {102, "1"}
CAU := {"Recovery on timer expiry","1"}
user
crec. outgoing. peer
remote
user
crec. outgoing. peer
remote
RELEASE. confirm (CAU) to crec.outgoing.ppus
PathRel (crec.outgoing)
RELEASE. indication (CAU) to crec.outgoing.ppus
PathRel (crec.outgoing)
The procedures releases resources associated with the outgoing AAL type 2 link (Note 3)
SendReject (CAU)
SendReject (CAU)
The procedures releases resources associated with the outgoing AAL type 2 link (Note 1)
release (crec)
crec.incoming.peer := none
StartReset (crec.incoming.ceid, crec.incoming.srid, FALSE )
This procedure initiates a reset procedure on an AAL type 2 channel (Note 4)
IDLE

NOTES

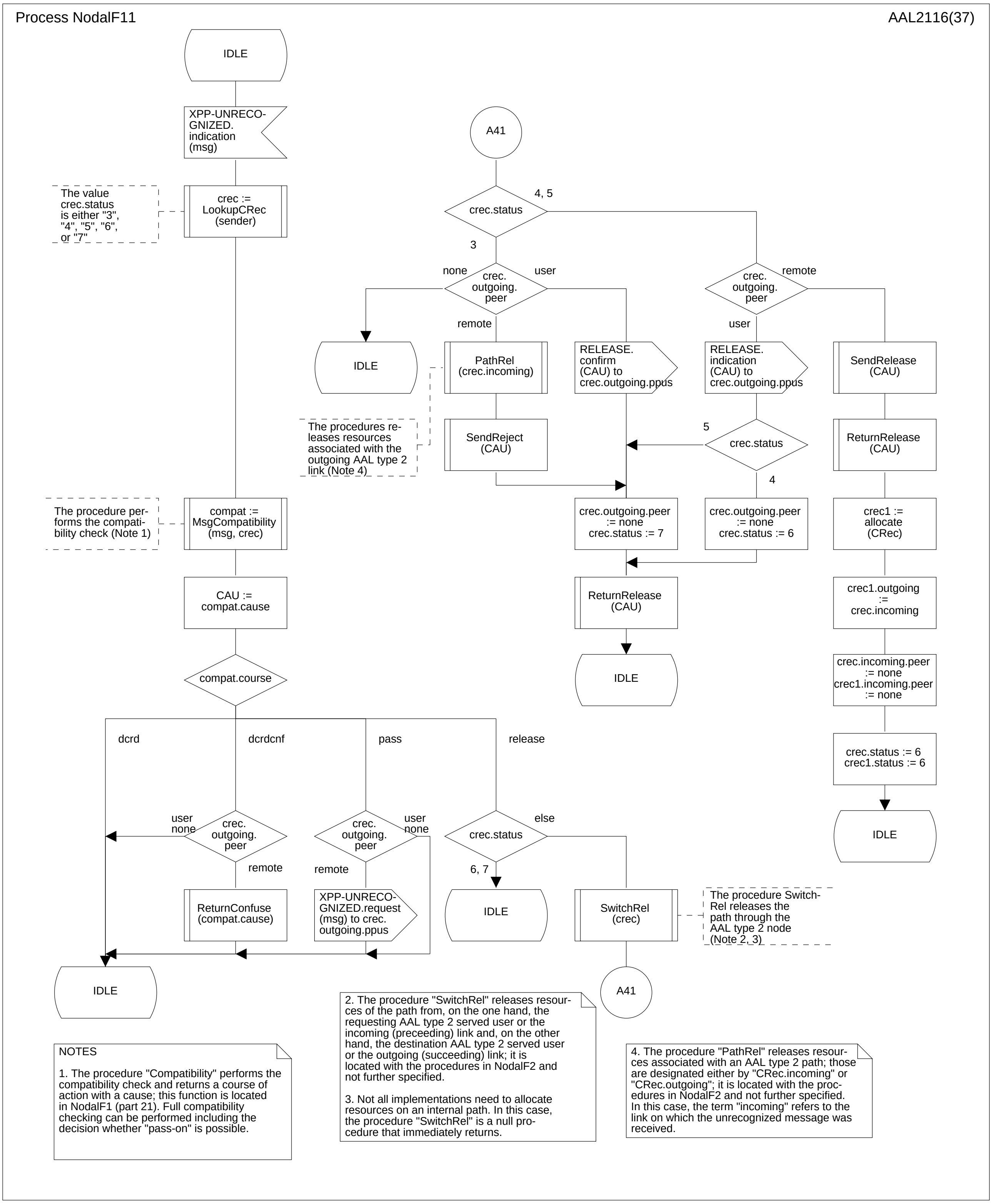
1. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the timeout occurred.

2. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

3. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

4. This procedure initiates a reset procedure on the AAL type 2 channel indicated by a "nodal signalling association" and a "Connection Element Identifier" that indicates both the path and the channel.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 16 of 37)***


Process NodalF11
AAL2116(37)
IDLE
XPP-UNRECO-GNIZED. indication (msg)
The value crec.status is either "3", "4", "5", "6", or "7"
crec := LookupCRec (sender)
The procedure performs the compatibility check (Note 1)
compat := MsgCompatibility (msg, crec)
CAU := compat.cause
compat.course
dcrd
dcrdcnf
pass
release
user
none
crec. outgoing. peer
remote
ReturnConfuse (compat.cause)
crec. outgoing. peer
user
none
remote
XPP-UNRECO-GNIZED.request (msg) to crec. outgoing.ppus
crec.status
else
6, 7
IDLE
SwitchRel (crec)
The procedure SwitchRel releases the path through the AAL type 2 node (Note 2, 3)
A41
IDLE
A41
crec.status
4, 5
3
none
crec. outgoing. peer
user
remote
IDLE
PathRel (crec.incoming)
The procedures releases resources associated with the outgoing AAL type 2 link (Note 4)
SendReject (CAU)
RELEASE. confirm (CAU) to crec.outgoing.ppus
crec.outgoing.peer := none crec.status := 7
ReturnRelease (CAU)
IDLE
crec. outgoing. peer
remote
user
RELEASE. indication (CAU) to crec.outgoing.ppus
5
crec.status
4
crec.outgoing.peer := none crec.status := 6
SendRelease (CAU)
ReturnRelease (CAU)
crec1 := allocate (CRec)
crec1.outgoing := crec.incoming
crec.incoming.peer := none crec1.incoming.peer := none
crec.status := 6 crec1.status := 6
IDLE

NOTES

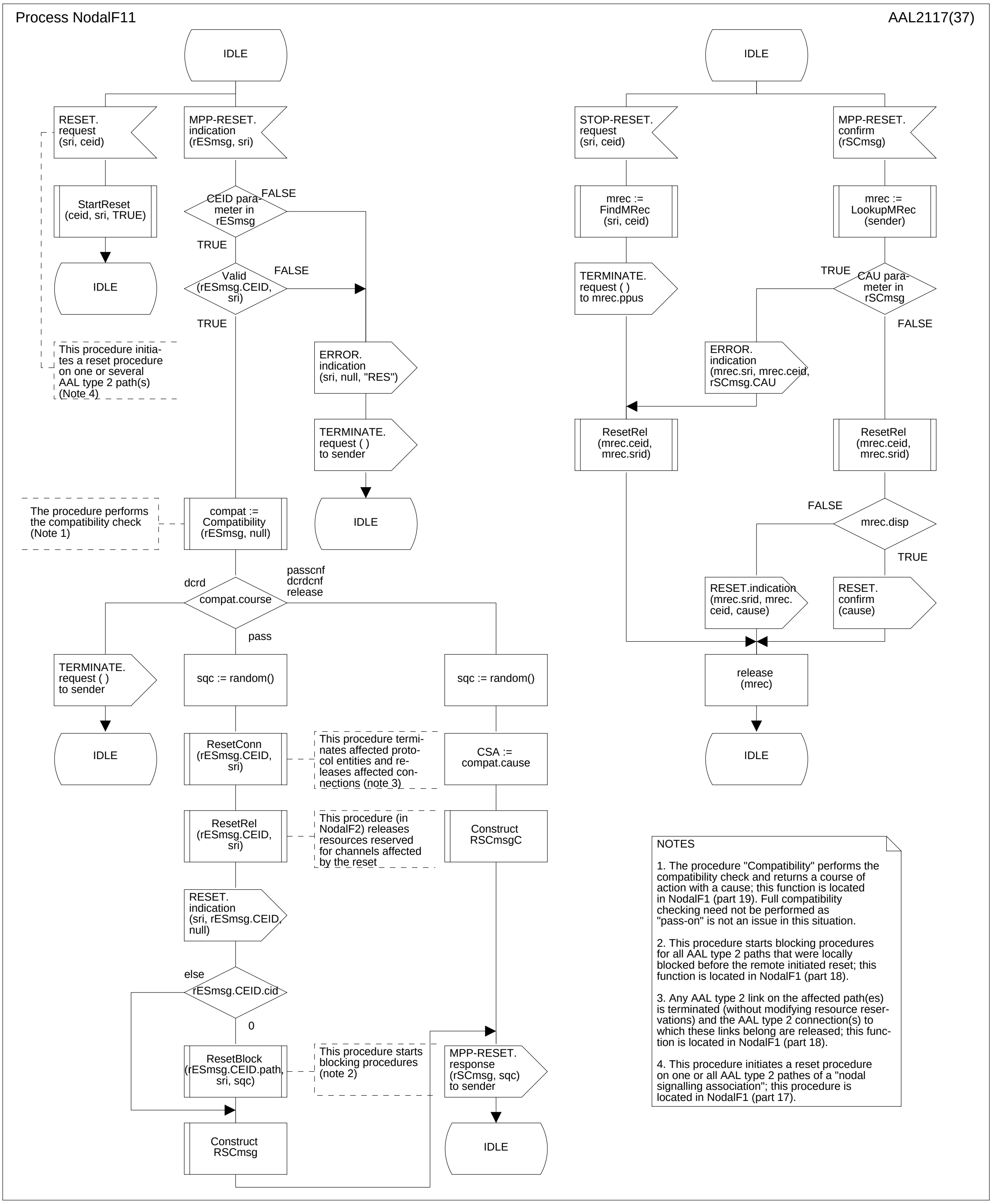
1. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 21). Full compatibility checking can be performed including the decision whether "pass-on" is possible.

2. The procedure "SwitchRel" releases resources of the path from, on the one hand, the requesting AAL type 2 served user or the incoming (preceeding) link and, on the other hand, the destination AAL type 2 served user or the outgoing (succeeding) link; it is located with the procedures in NodalF2 and not further specified.

3. Not all implementations need to allocate resources on an internal path. In this case, the procedure "SwitchRel" is a null procedure that immediately returns.

4. The procedure "PathRel" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing"; it is located with the procedures in NodalF2 and not further specified. In this case, the term "incoming" refers to the link on which the unrecognized message was received.

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1** ***(part 17 of 37)***



NOTES

1. The procedure "Compatibility" performs the compatibility check and returns a course of action with a cause; this function is located in NodalF1 (part 19). Full compatibility checking need not be performed as "pass-on" is not an issue in this situation.

2. This procedure starts blocking procedures for all AAL type 2 paths that were locally blocked before the remote initiated reset; this function is located in NodalF1 (part 18).

3. Any AAL type 2 link on the affected path(es) is terminated (without modifying resource reservations) and the AAL type 2 connection(s) to which these links belong are released; this function is located in NodalF1 (part 18).

4. This procedure initiates a reset procedure on one or all AAL type 2 pathes of a "nodal signalling association"; this procedure is located in NodalF1 (part 17).

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 18 of 37)***

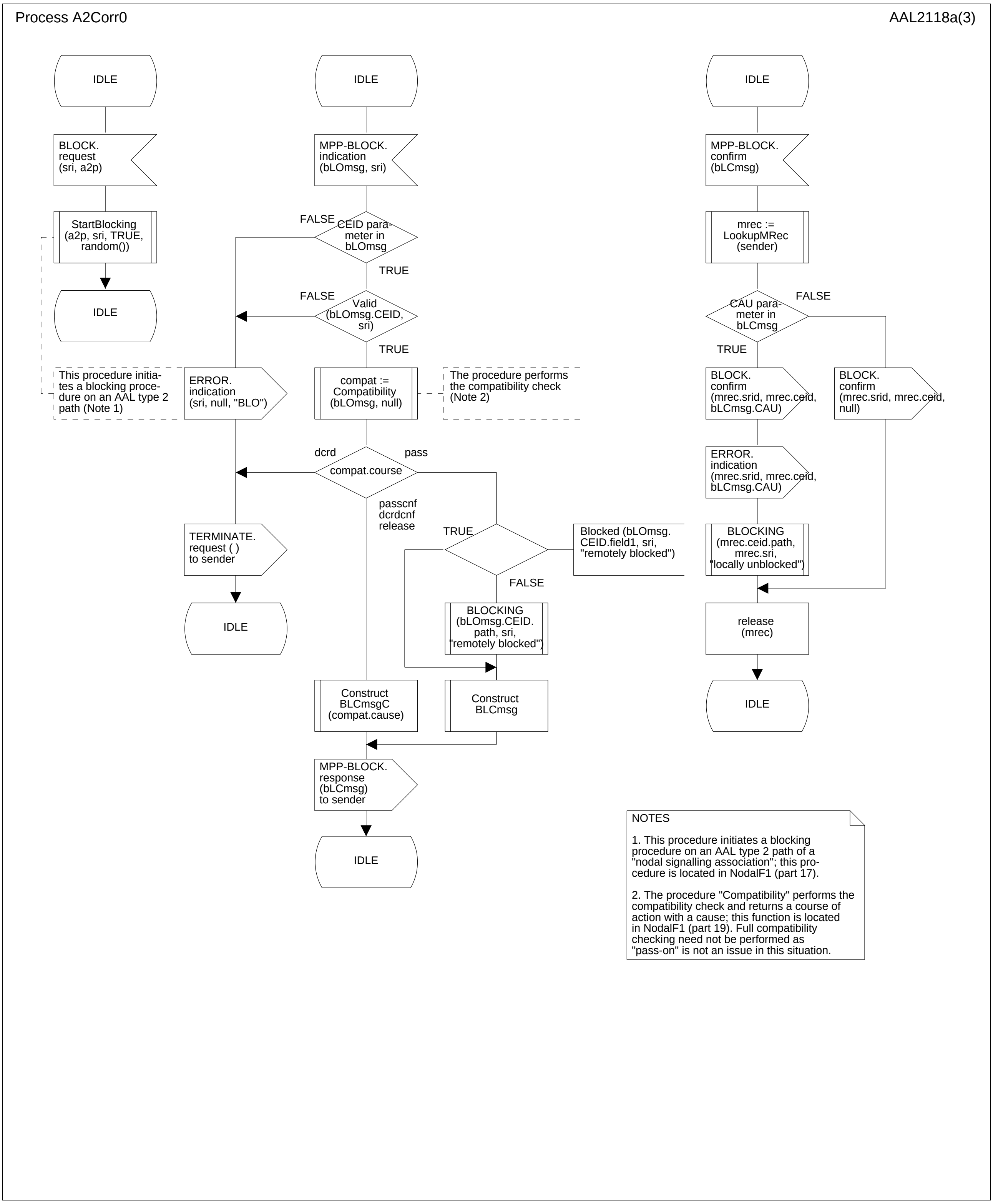


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1** ***(part 19 of 37)***

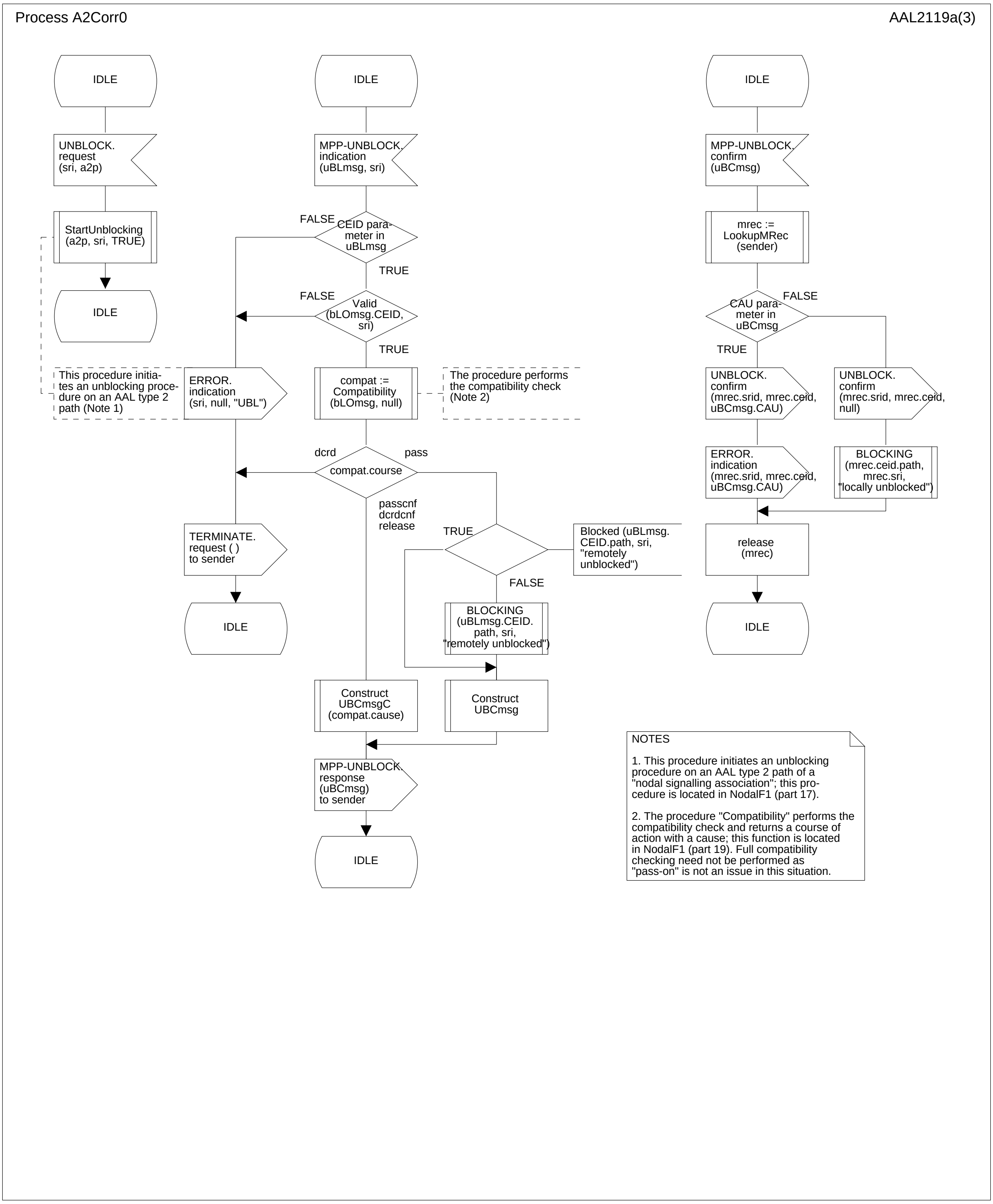


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 *(part 20 of 37)***

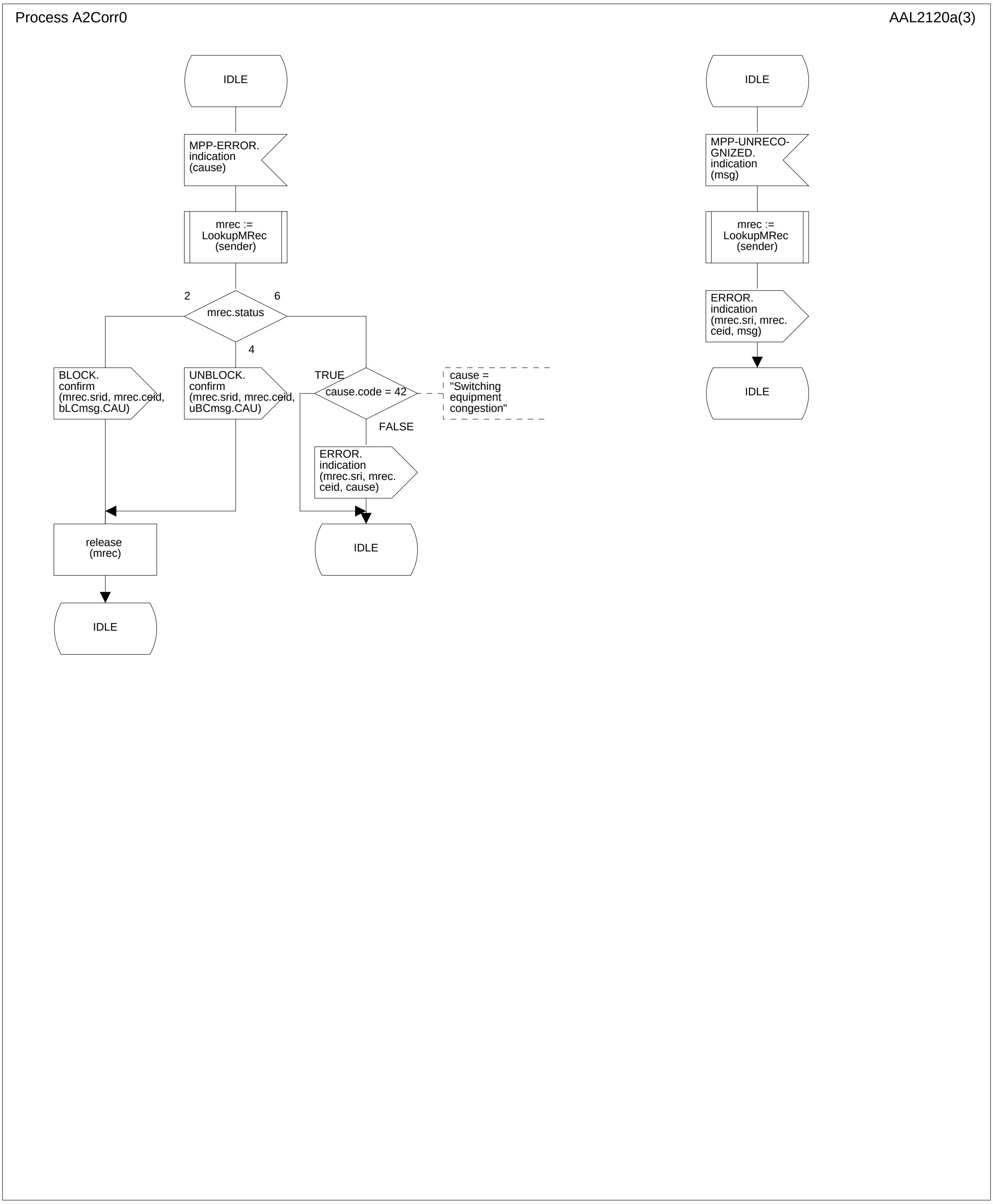
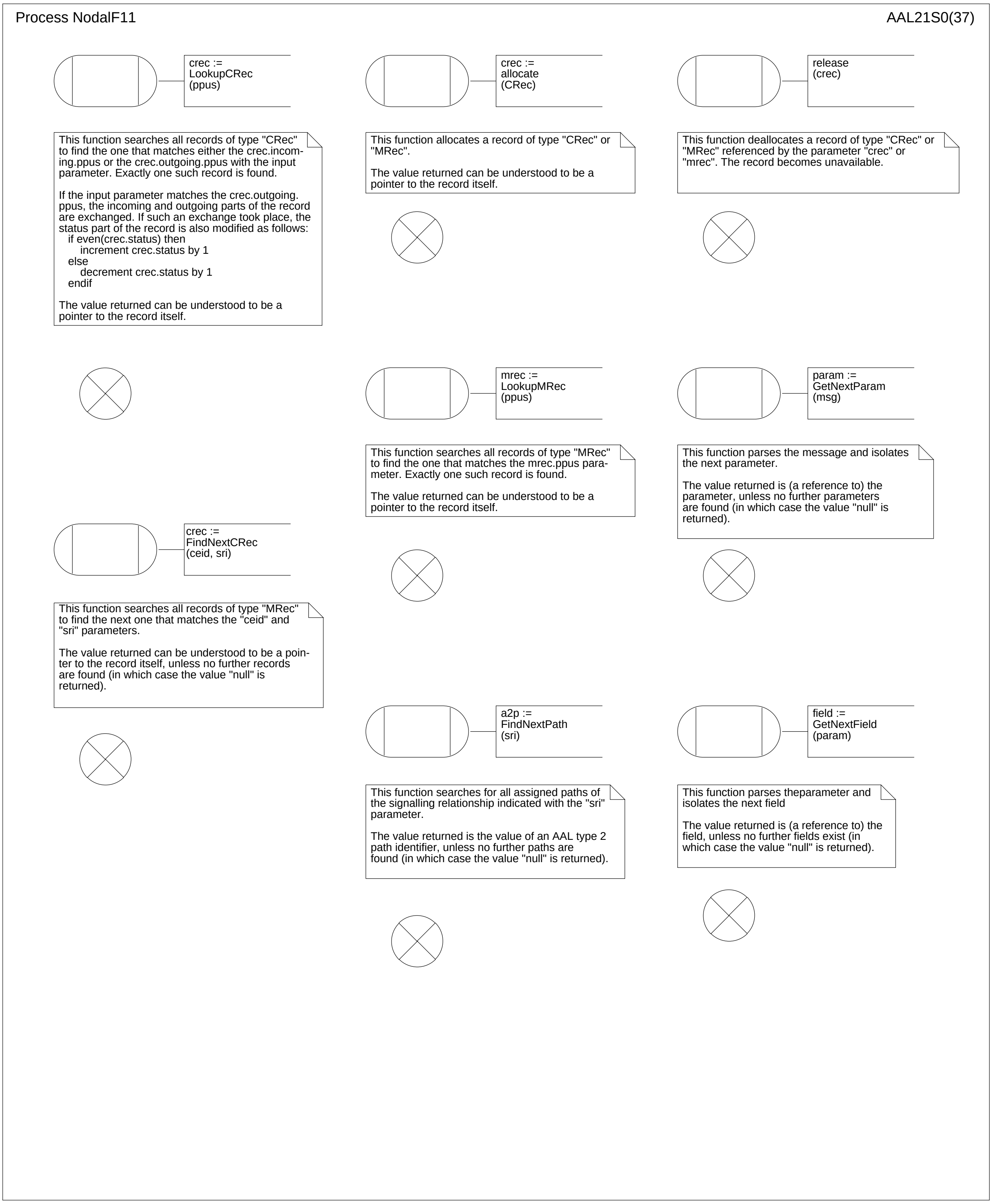


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1** ***(part 21 of 37)***


Process NodalF11
AAL21S0(37)
crec := LookupCRec (ppus)
crec := allocate (CRec)
release (crec)
mrec := LookupMRec (ppus)
param := GetNextParam (msg)
crec := FindNextCRec (ceid, sri)
a2p := FindNextPath (sri)
field := GetNextField (param)

This function searches all records of type "CRec" to find the one that matches either the crec.incoming.ppus or the crec.outgoing.ppus with the input parameter. Exactly one such record is found.

If the input parameter matches the crec.outgoing. ppus, the incoming and outgoing parts of the record are exchanged. If such an exchange took place, the status part of the record is also modified as follows:

```
if even(crec.status) then
    increment crec.status by 1
else
    decrement crec.status by 1
endif
```

The value returned can be understood to be a pointer to the record itself.

This function allocates a record of type "CRec" or "MRec".

The value returned can be understood to be a pointer to the record itself.

This function deallocates a record of type "CRec" or "MRec" referenced by the parameter "crec" or "mrec". The record becomes unavailable.

This function searches all records of type "MRec" to find the one that matches the mrec.ppus parameter. Exactly one such record is found.

The value returned can be understood to be a pointer to the record itself.

This function parses the message and isolates the next parameter.

The value returned is (a reference to) the parameter, unless no further parameters are found (in which case the value "null" is returned).

This function searches all records of type "MRec" to find the next one that matches the "ceid" and "sri" parameters.

The value returned can be understood to be a pointer to the record itself, unless no further records are found (in which case the value "null" is returned).

This function searches for all assigned paths of the signalling relationship indicated with the "sri" parameter.

The value returned is the value of an AAL type 2 path identifier, unless no further paths are found (in which case the value "null" is returned).

This function parses theparameter and isolates the next field

The value returned is (a reference to) the field, unless no further fields exist (in which case the value "null" is returned).

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (procedures)** ***(part 22 of 37)***

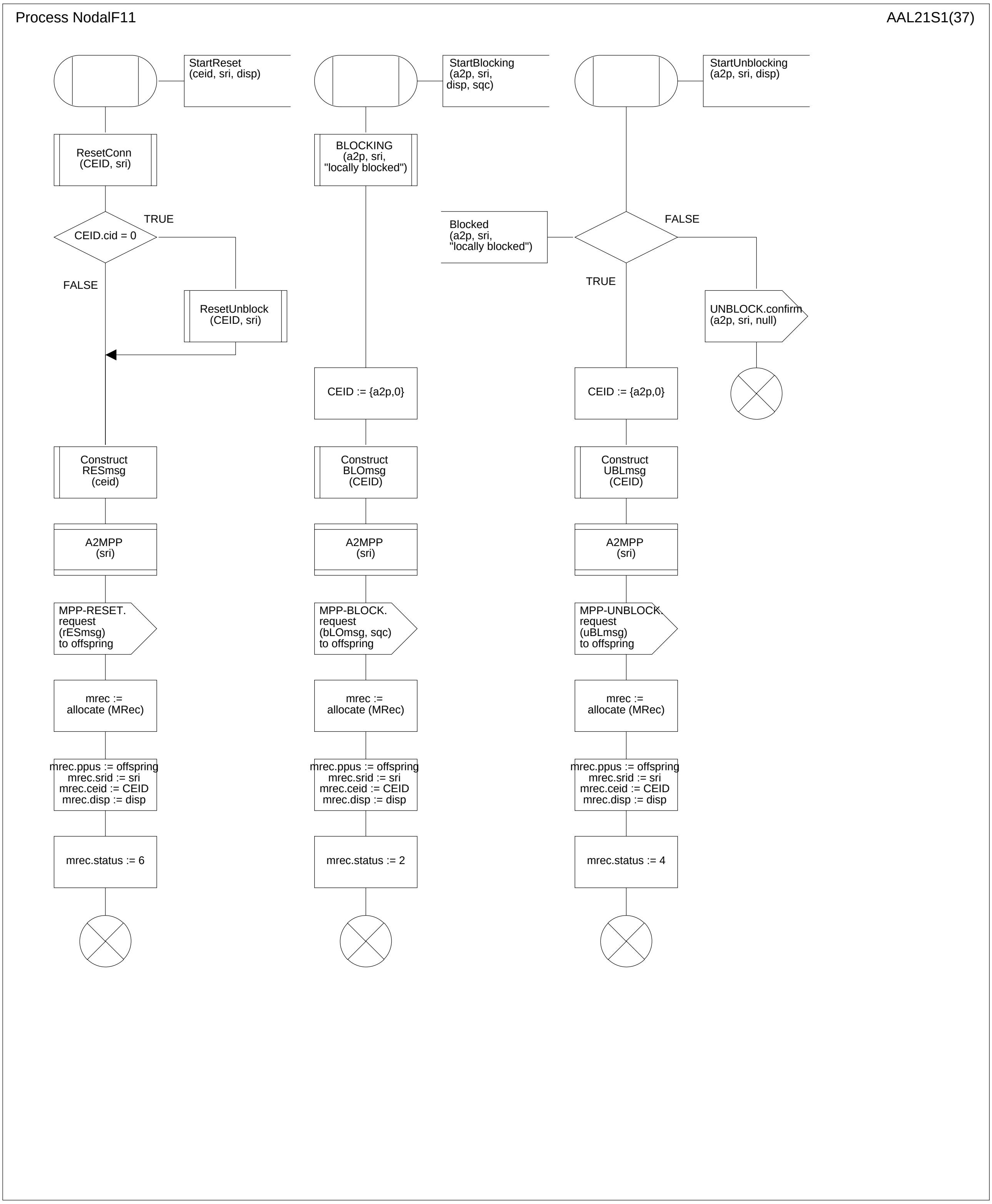


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (procedures)** ***(part 23 of 37)***

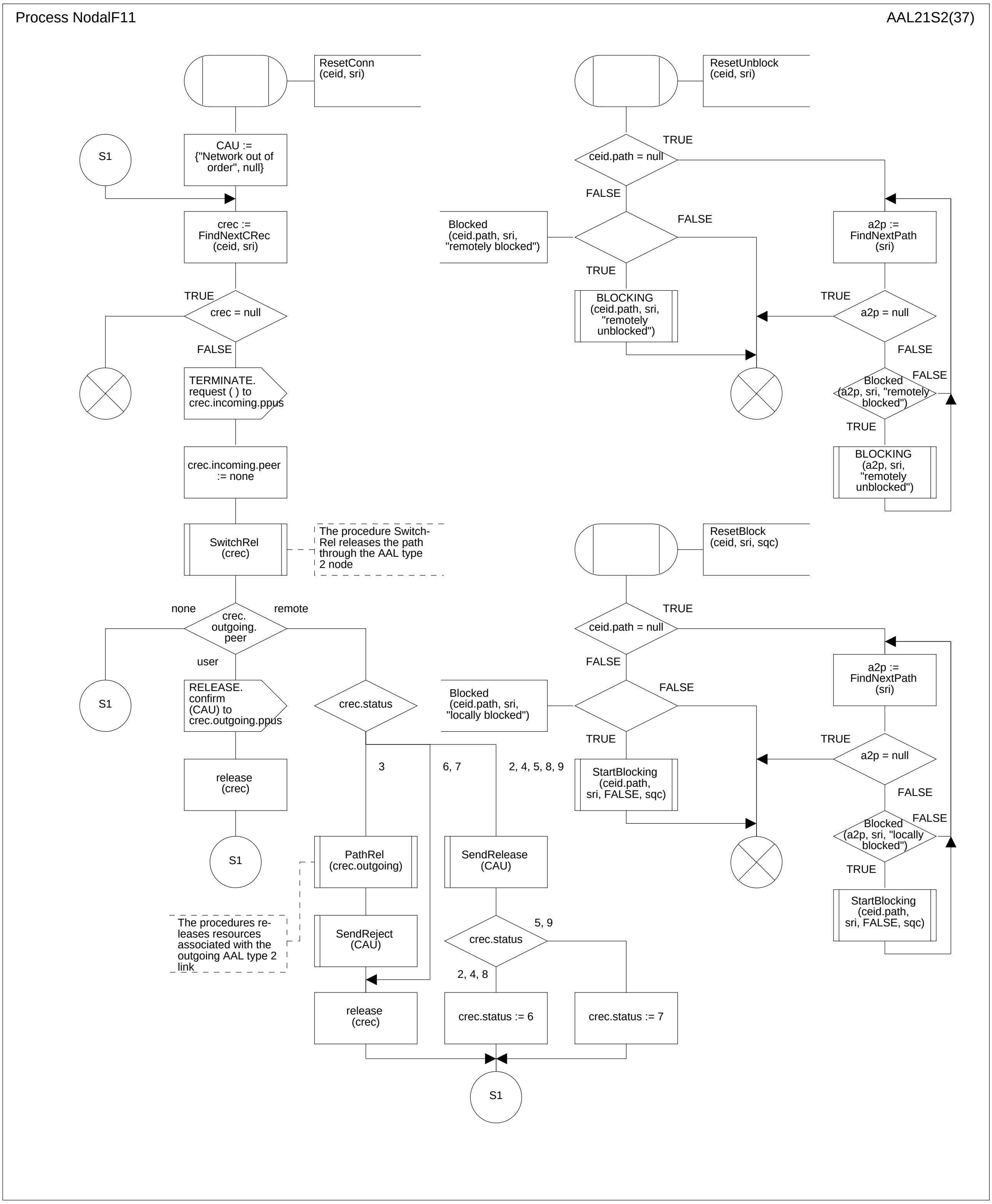


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (procedures)** ***(part 24 of 37)***

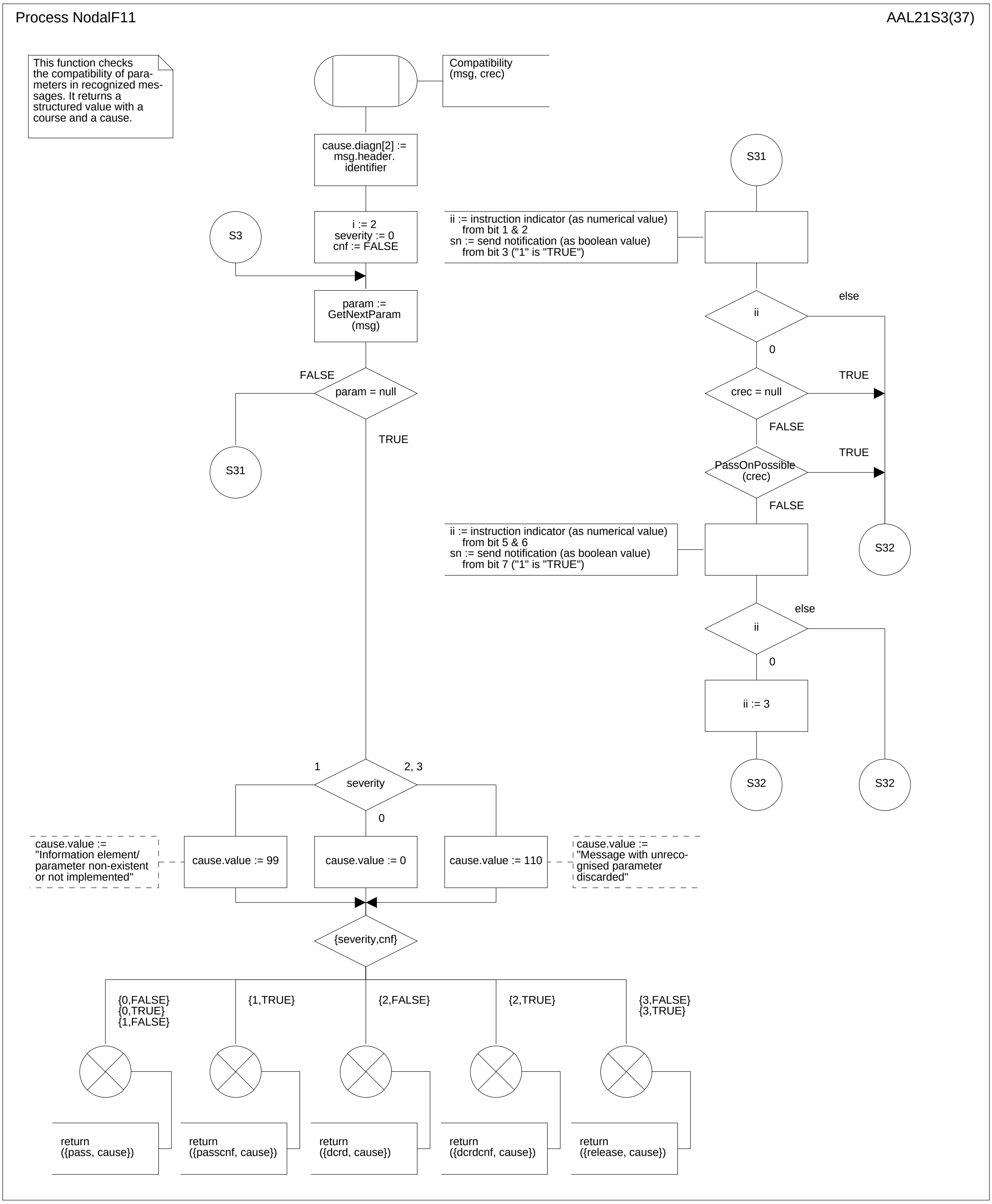

Process NodalF11
AAL21S3(37)
This function checks the compatibility of parameters in recognized messages. It returns a structured value with a course and a cause.
Compatibility (msg, crec)
cause.diagn[2] := msg.header. identifier
S3
i := 2
severity := 0
cnf := FALSE
param := GetNextParam (msg)
FALSE
param = null
TRUE
S31
S31
ii := instruction indicator (as numerical value) from bit 1 & 2
sn := send notification (as boolean value) from bit 3 ("1" is "TRUE")
ii
else
0
crec = null
TRUE
FALSE
PassOnPossible (crec)
TRUE
FALSE
S32
ii := instruction indicator (as numerical value) from bit 5 & 6
sn := send notification (as boolean value) from bit 7 ("1" is "TRUE")
ii
else
0
ii := 3
S32
S32
severity
1
2, 3
0
cause.value := "Information element/ parameter non-existent or not implemented"
cause.value := 99
cause.value := 0
cause.value := 110
cause.value := "Message with unrecognised parameter discarded"
{severity,cnf}
{0,FALSE}
{0,TRUE}
{1,FALSE}
{1,TRUE}
{2,FALSE}
{2,TRUE}
{3,FALSE}
{3,TRUE}
return ({pass, cause})
return ({passcnf, cause})
return ({dcrd, cause})
return ({dcrdcnf, cause})
return ({release, cause})

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (procedures)** ***(part 25 of 37)***

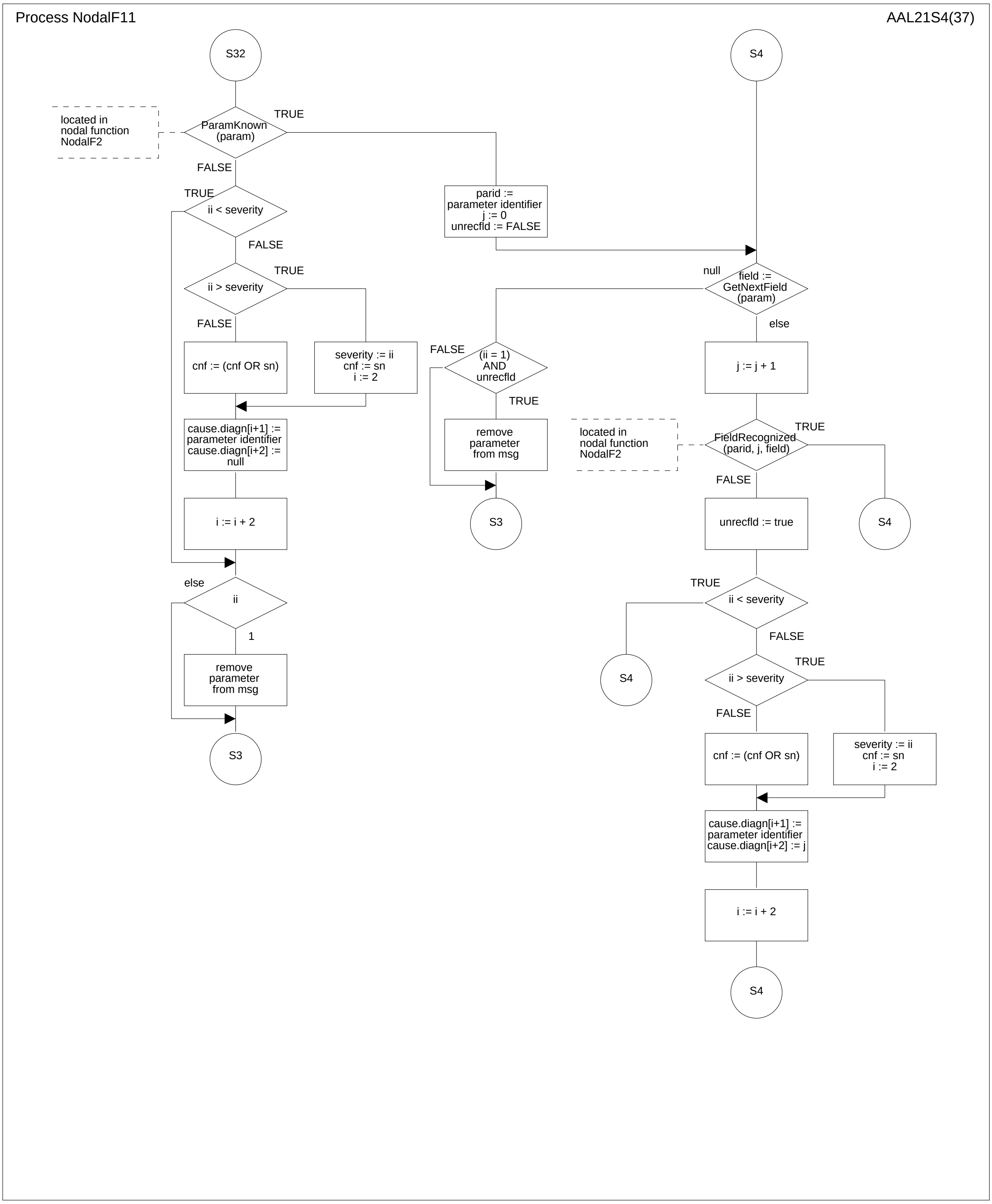


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (procedures)** ***(part 26 of 37)***

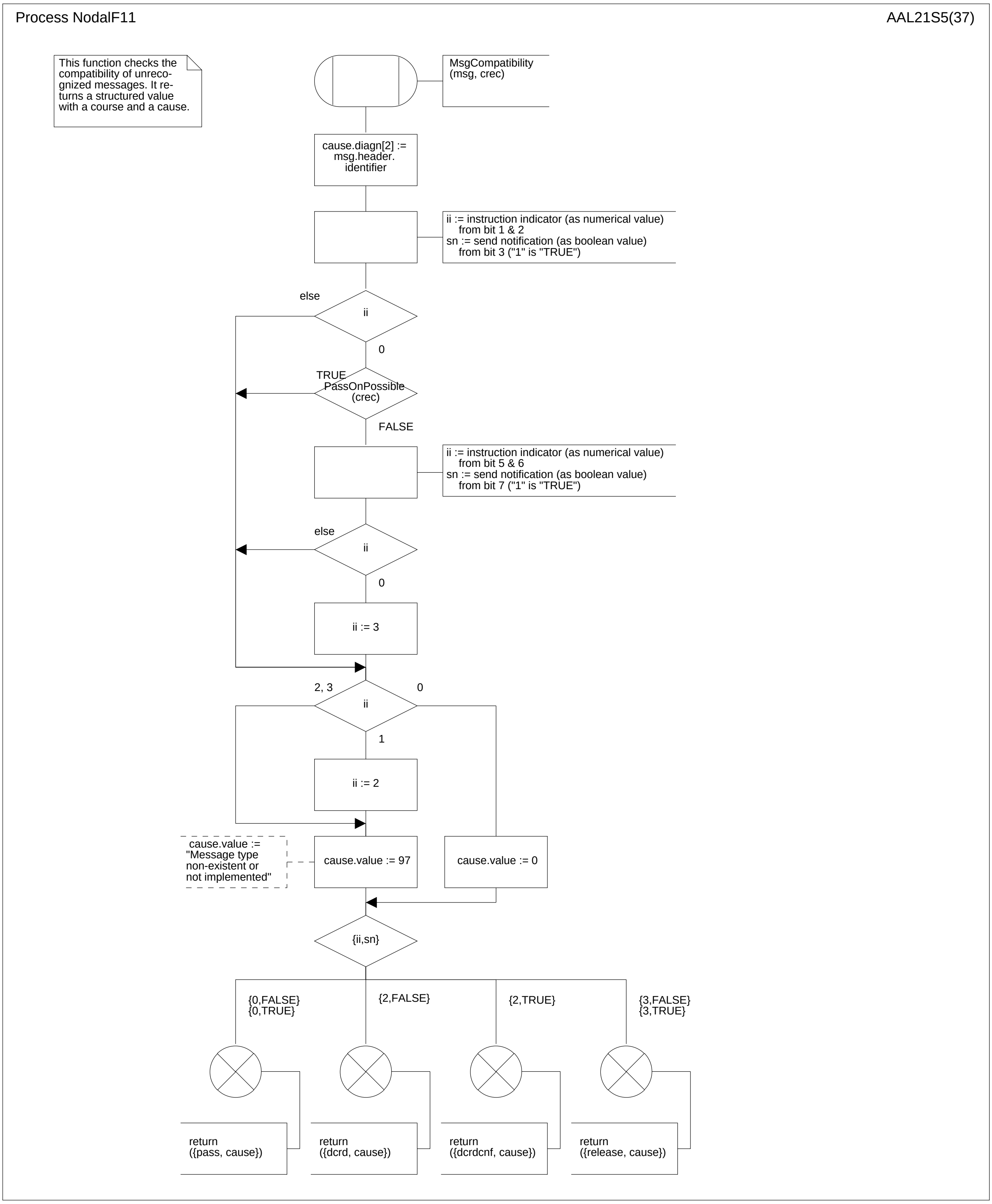

Process NodalF11
AAL21S5(37)
This function checks the compatibility of unreco-gnized messages. It re-turns a structured value with a course and a cause.
MsgCompatibility (msg, crec)
cause.diagn[2] := msg.header. identifier
ii := instruction indicator (as numerical value) from bit 1 & 2
sn := send notification (as boolean value) from bit 3 ("1" is "TRUE")
else
ii
0
TRUE
PassOnPossible (crec)
FALSE
ii := instruction indicator (as numerical value) from bit 5 & 6
sn := send notification (as boolean value) from bit 7 ("1" is "TRUE")
else
ii
0
ii := 3
2, 3
0
ii
1
ii := 2
cause.value := "Message type non-existent or not implemented"
cause.value := 97
cause.value := 0
{ii,sn}
{0,FALSE} {0,TRUE}
{2,FALSE}
{2,TRUE}
{3,FALSE} {3,TRUE}
return ({pass, cause})
return ({dcrd, cause})
return ({dcrdcnf, cause})
return ({release, cause})

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (procedures)** ***(part 27 of 37)***

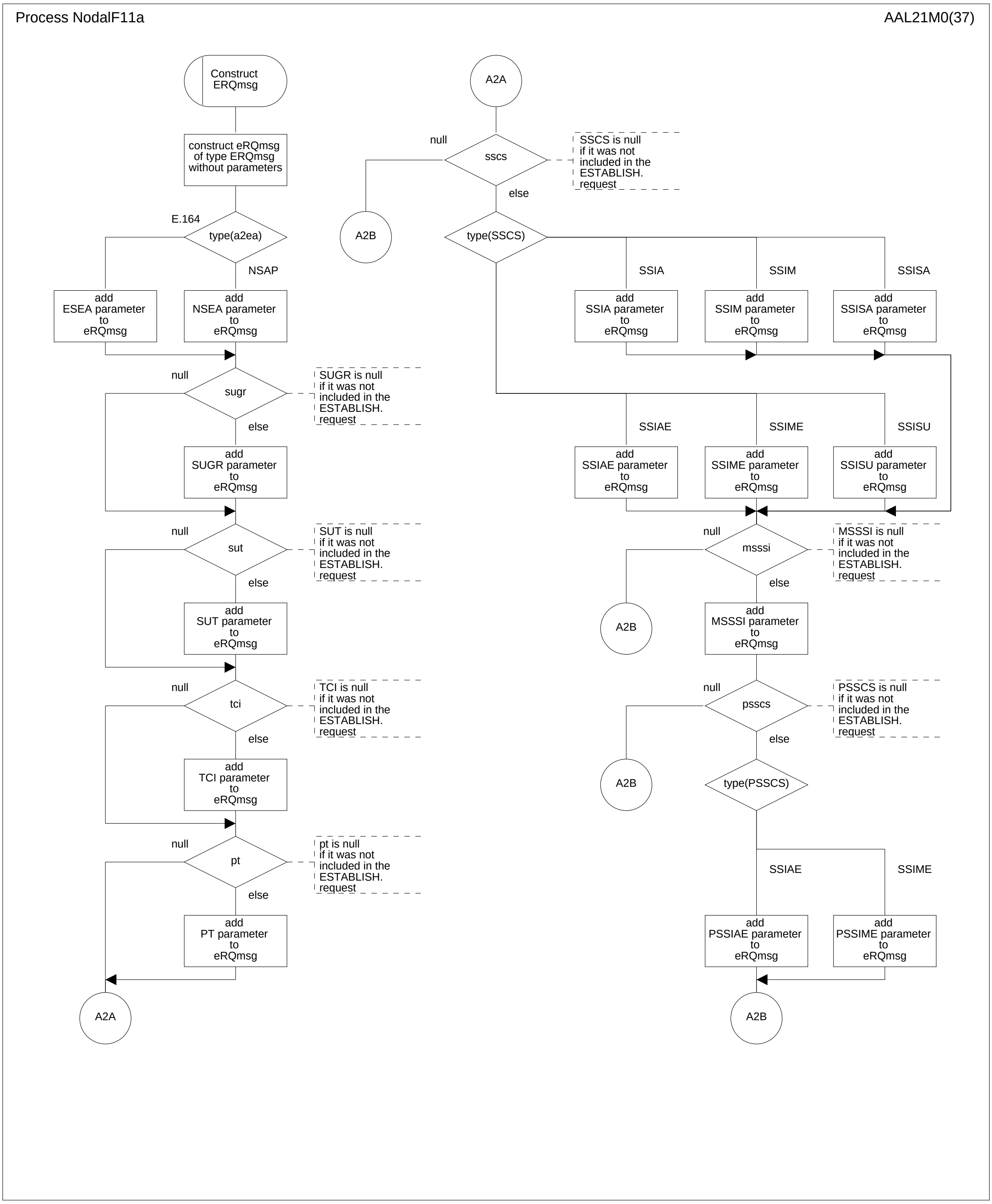

Process NodalF11a
AAL21M0(37)
Construct ERQmsg
construct eRQmsg of type ERQmsg without parameters
type(a2ea)
E.164
NSAP
add ESEA parameter to eRQmsg
add NSEA parameter to eRQmsg
sugr
null
else
SUGR is null if it was not included in the ESTABLISH. request
add SUGR parameter to eRQmsg
sut
SUT is null if it was not included in the ESTABLISH. request
add SUT parameter to eRQmsg
tci
TCI is null if it was not included in the ESTABLISH. request
add TCI parameter to eRQmsg
pt
pt is null if it was not included in the ESTABLISH. request
add PT parameter to eRQmsg
A2A
sscs
SSCS is null if it was not included in the ESTABLISH. request
A2B
type(SSCS)
SSIA
SSIM
SSISA
add SSIA parameter to eRQmsg
add SSIM parameter to eRQmsg
add SSISA parameter to eRQmsg
SSIAE
SSIME
SSISU
add SSIAE parameter to eRQmsg
add SSIME parameter to eRQmsg
add SSISU parameter to eRQmsg
msssi
MSSSI is null if it was not included in the ESTABLISH. request
add MSSSI parameter to eRQmsg
psscs
PSSCS is null if it was not included in the ESTABLISH. request
type(PSSCS)
SSIAE
SSIME
add PSSIAE parameter to eRQmsg
add PSSIME parameter to eRQmsg

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 28 of 37)***

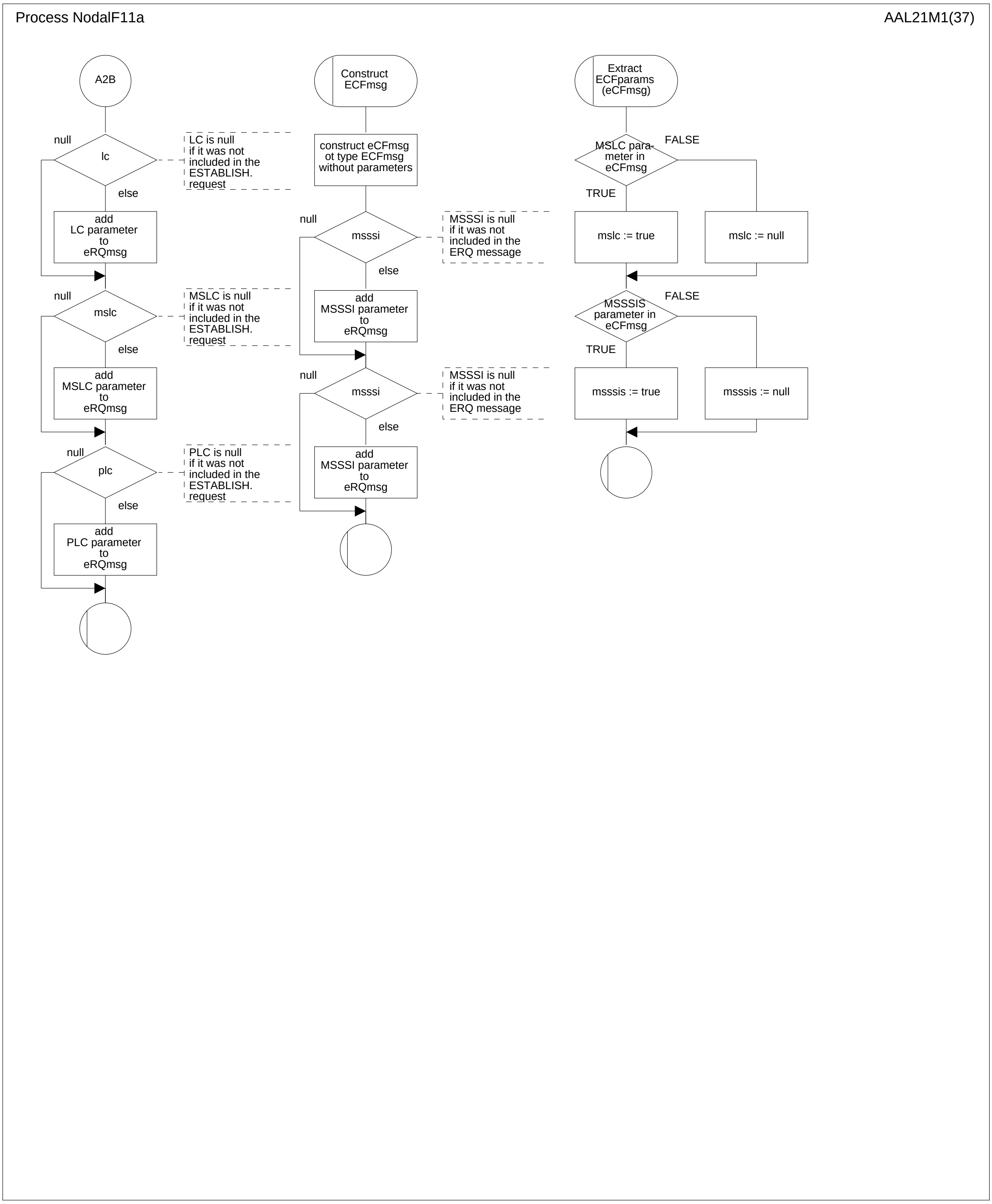


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 29 of 37)***

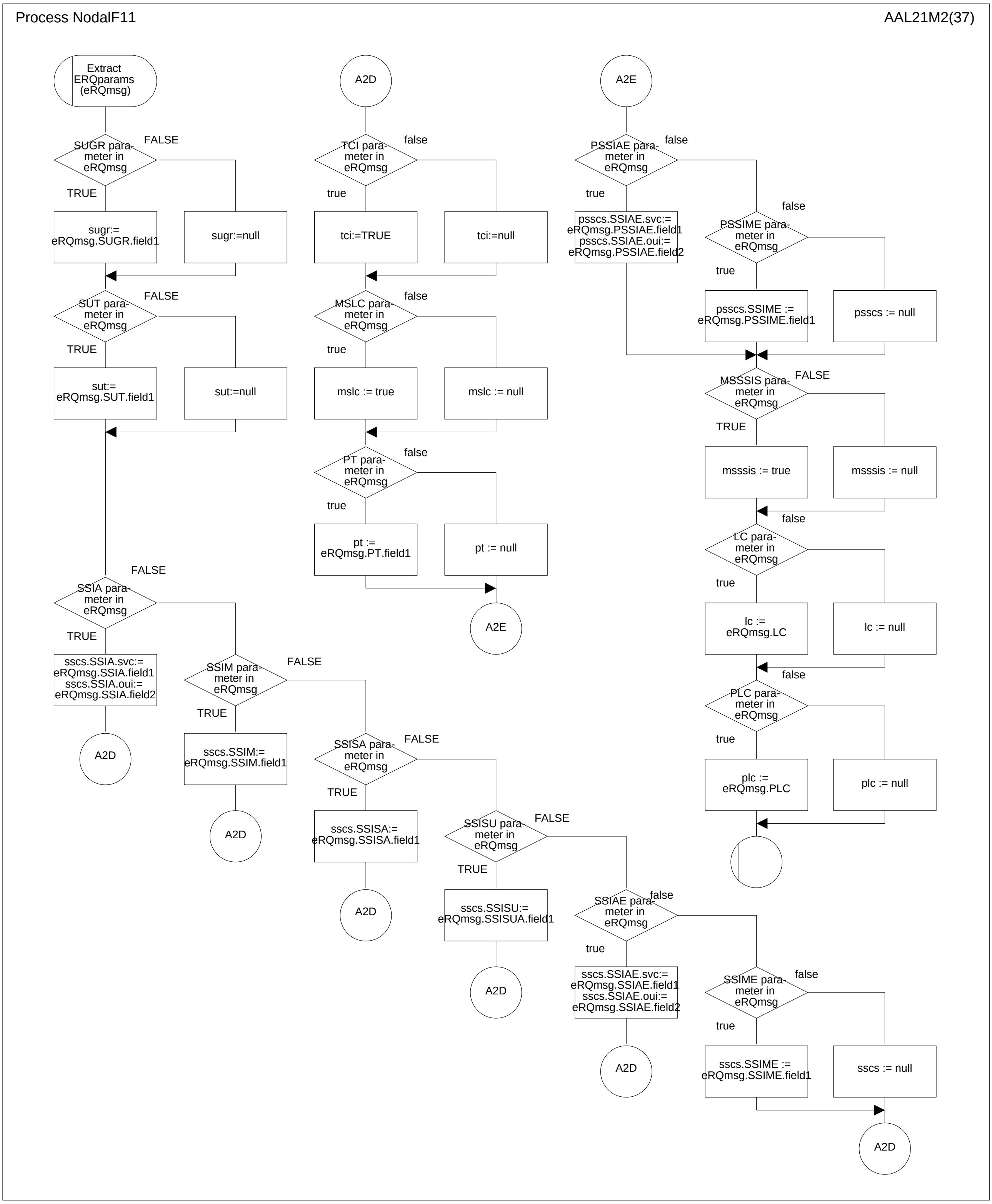
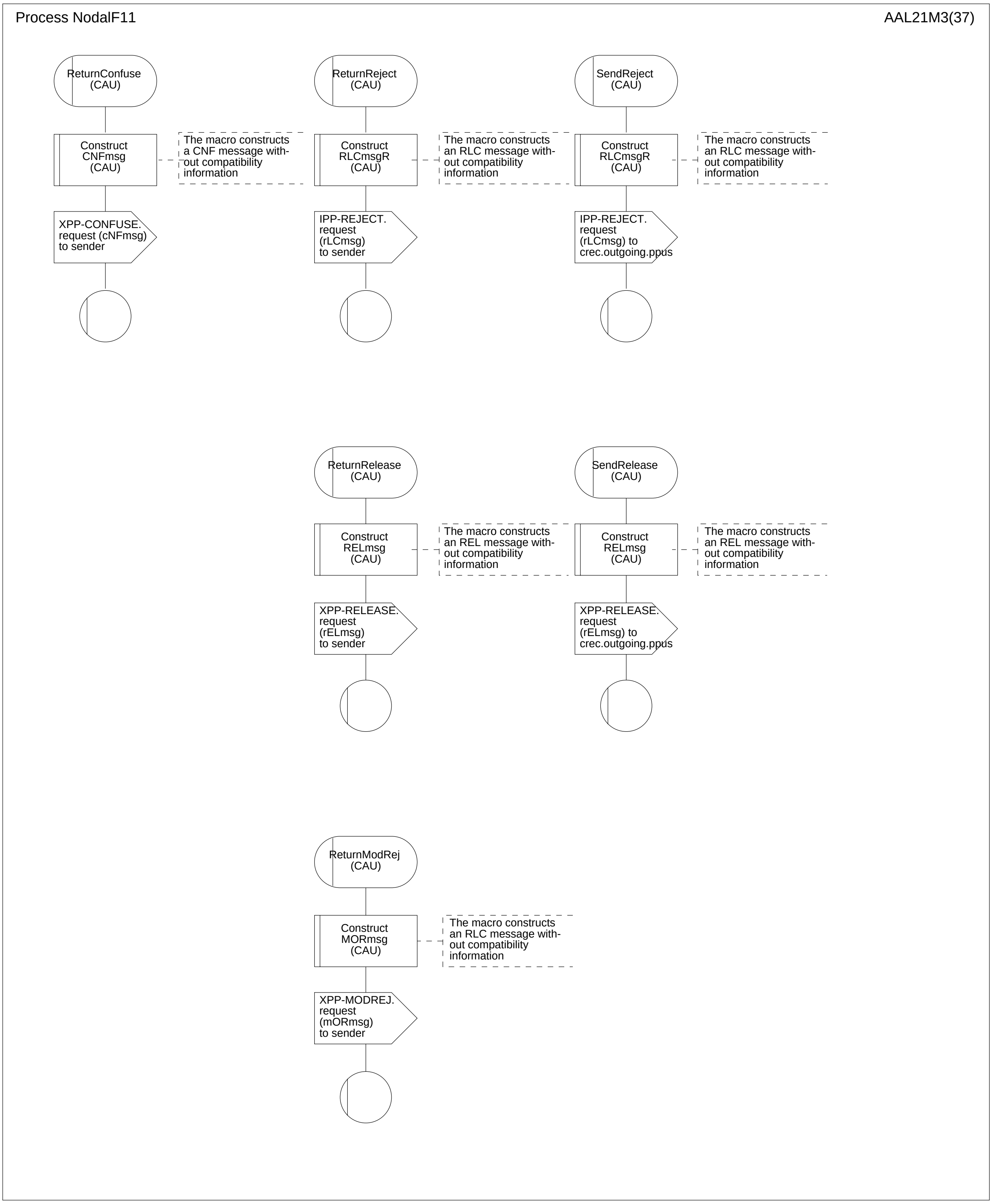
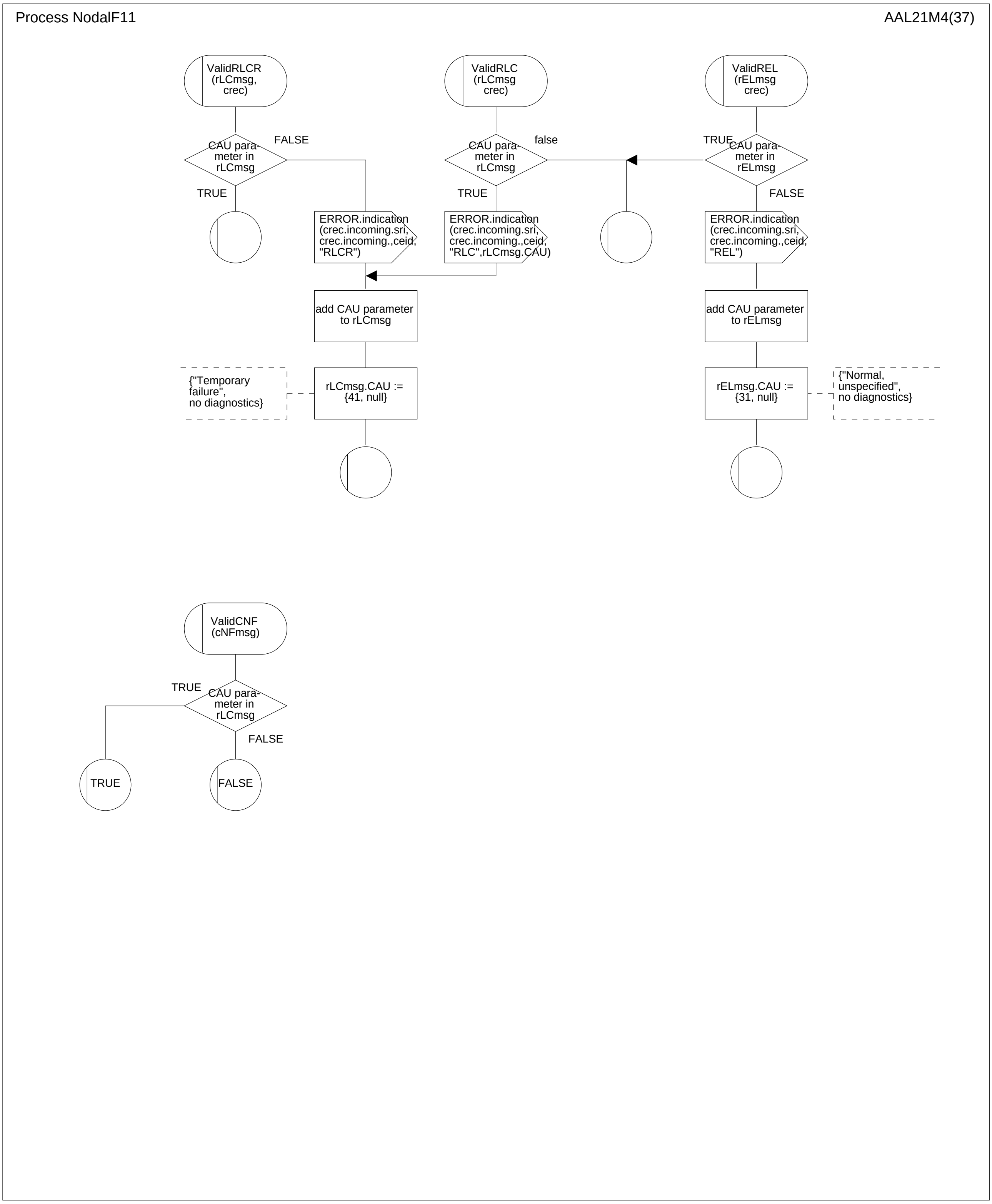


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 30 of 37)***



Process NodalF11
AAL21M3(37)
ReturnConfuse (CAU)
Construct CNFmsg (CAU)
The macro constructs a CNF message with- out compatibility information
XPP-CONFUSE. request (cNFmsg) to sender
ReturnReject (CAU)
Construct RLCmsgR (CAU)
The macro constructs an RLC message with- out compatibility information
IPP-REJECT. request (rLCmsg) to sender
SendReject (CAU)
Construct RLCmsgR (CAU)
The macro constructs an RLC message with- out compatibility information
IPP-REJECT. request (rLCmsg) to crec.outgoing.ppus
ReturnRelease (CAU)
Construct RELmsg (CAU)
The macro constructs an REL message with- out compatibility information
XPP-RELEASE. request (rELmsg) to sender
SendRelease (CAU)
Construct RELmsg (CAU)
The macro constructs an REL message with- out compatibility information
XPP-RELEASE. request (rELmsg) to crec.outgoing.ppus
ReturnModRej (CAU)
Construct MORmsg (CAU)
The macro constructs an RLC message with- out compatibility information
XPP-MODREJ. request (mORmsg) to sender

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 31 of 37)***



Process NodalF11
AAL21M4(37)
ValidRLCR (rLCmsg, crec)
CAU para- meter in rLCmsg
FALSE
TRUE
ERROR.indication (crec.incoming.sri, crec.incoming.,ceid, "RLCR")
add CAU parameter to rLCmsg
{"Temporary failure", no diagnostics}
rLCmsg.CAU := {41, null}
ValidRLC (rLCmsg crec)
CAU para- meter in rLCmsg
false
TRUE
ERROR.indication (crec.incoming.sri, crec.incoming.,ceid, "RLC",rLCmsg.CAU)
ValidREL (rELmsg crec)
TRUE
CAU para- meter in rELmsg
FALSE
ERROR.indication (crec.incoming.sri, crec.incoming.,ceid, "REL")
add CAU parameter to rELmsg
rELmsg.CAU := {31, null}
{"Normal, unspecified", no diagnostics}
ValidCNF (cNFmsg)
TRUE
CAU para- meter in rLCmsg
FALSE
TRUE
FALSE

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 32 of 37)***

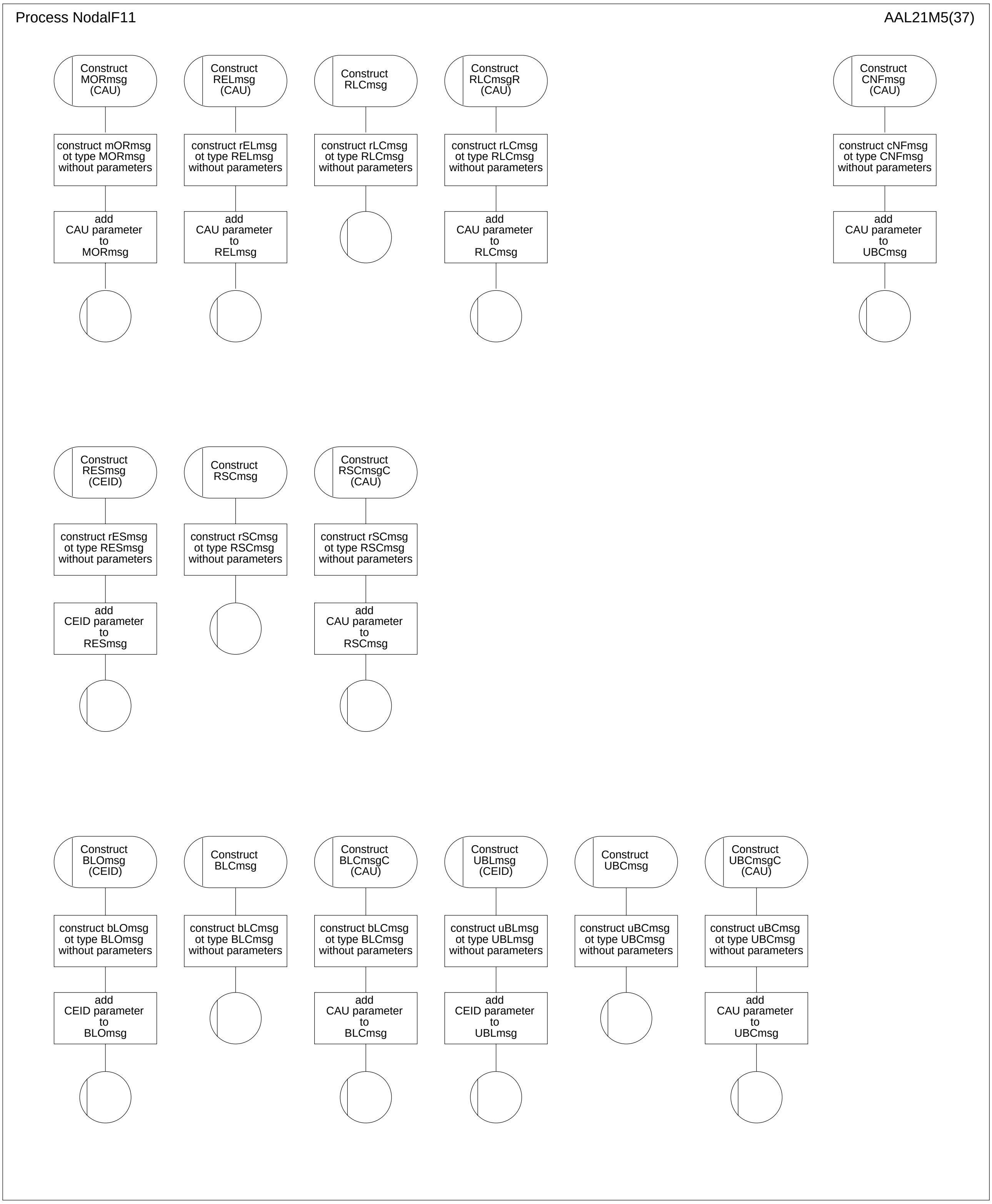

Process NodalF11
AAL21M5(37)
Construct MORmsg (CAU)
construct mORmsg ot type MORmsg without parameters
add CAU parameter to MORmsg
Construct RELmsg (CAU)
construct rELmsg ot type RELmsg without parameters
add CAU parameter to RELmsg
Construct RLCmsg
construct rLCmsg ot type RLCmsg without parameters
Construct RLCmsgR (CAU)
construct rLCmsg ot type RLCmsg without parameters
add CAU parameter to RLCmsg
Construct CNFmsg (CAU)
construct cNFmsg ot type CNFmsg without parameters
add CAU parameter to UBCmsg
Construct RESmsg (CEID)
construct rESmsg ot type RESmsg without parameters
add CEID parameter to RESmsg
Construct RSCmsg
construct rSCmsg ot type RSCmsg without parameters
Construct RSCmsgC (CAU)
construct rSCmsg ot type RSCmsg without parameters
add CAU parameter to RSCmsg
Construct BLOmsg (CEID)
construct bLOmsg ot type BLOmsg without parameters
add CEID parameter to BLOmsg
Construct BLCmsg
construct bLCmsg ot type BLCmsg without parameters
Construct BLCmsgC (CAU)
construct bLCmsg ot type BLCmsg without parameters
add CAU parameter to BLCmsg
Construct UBLmsg (CEID)
construct uBLmsg ot type UBLmsg without parameters
add CEID parameter to UBLmsg
Construct UBCmsg
construct uBCmsg ot type UBCmsg without parameters
Construct UBCmsgC (CAU)
construct uBCmsg ot type UBCmsg without parameters
add CAU parameter to UBCmsg

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 33 of 37)***

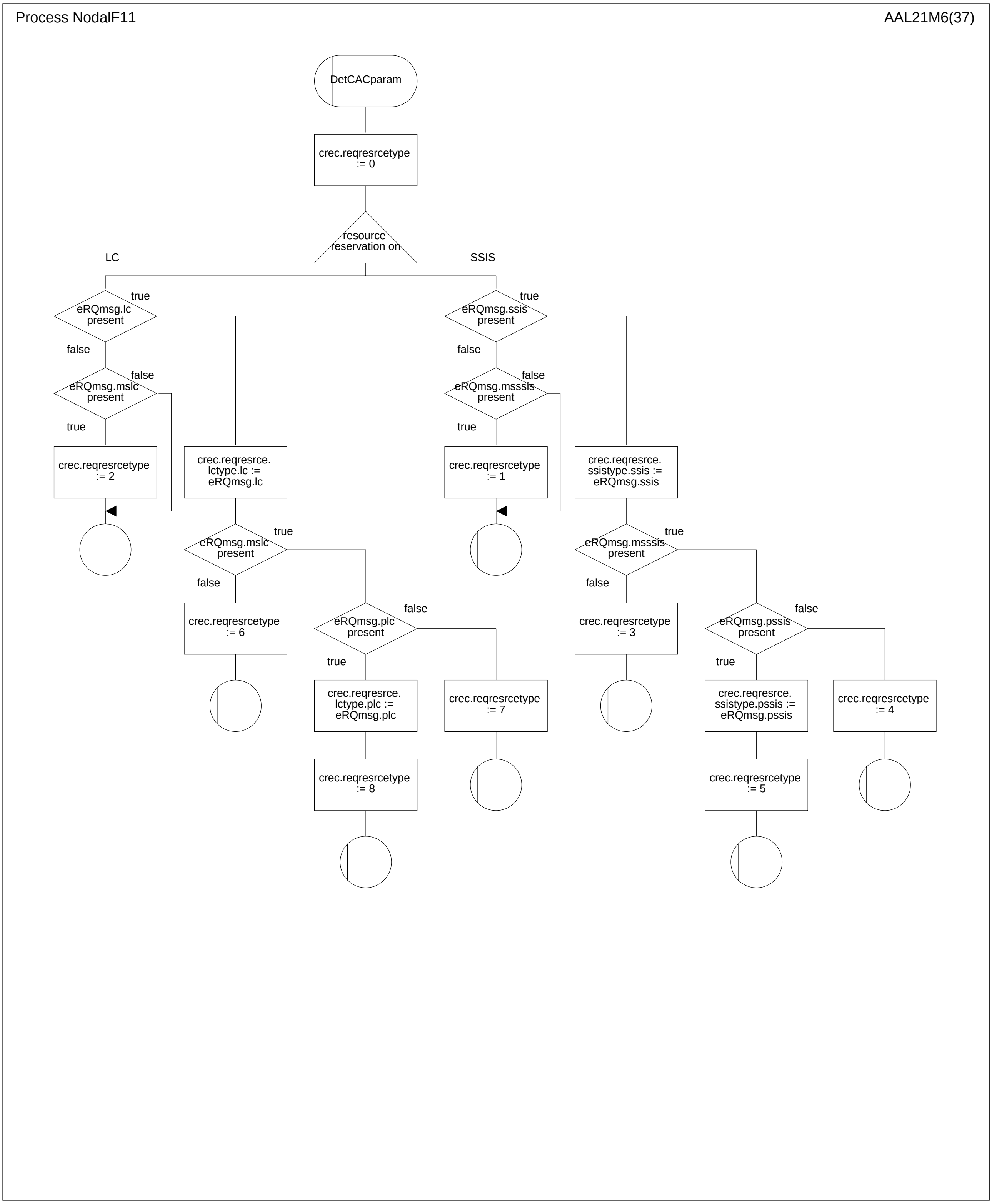
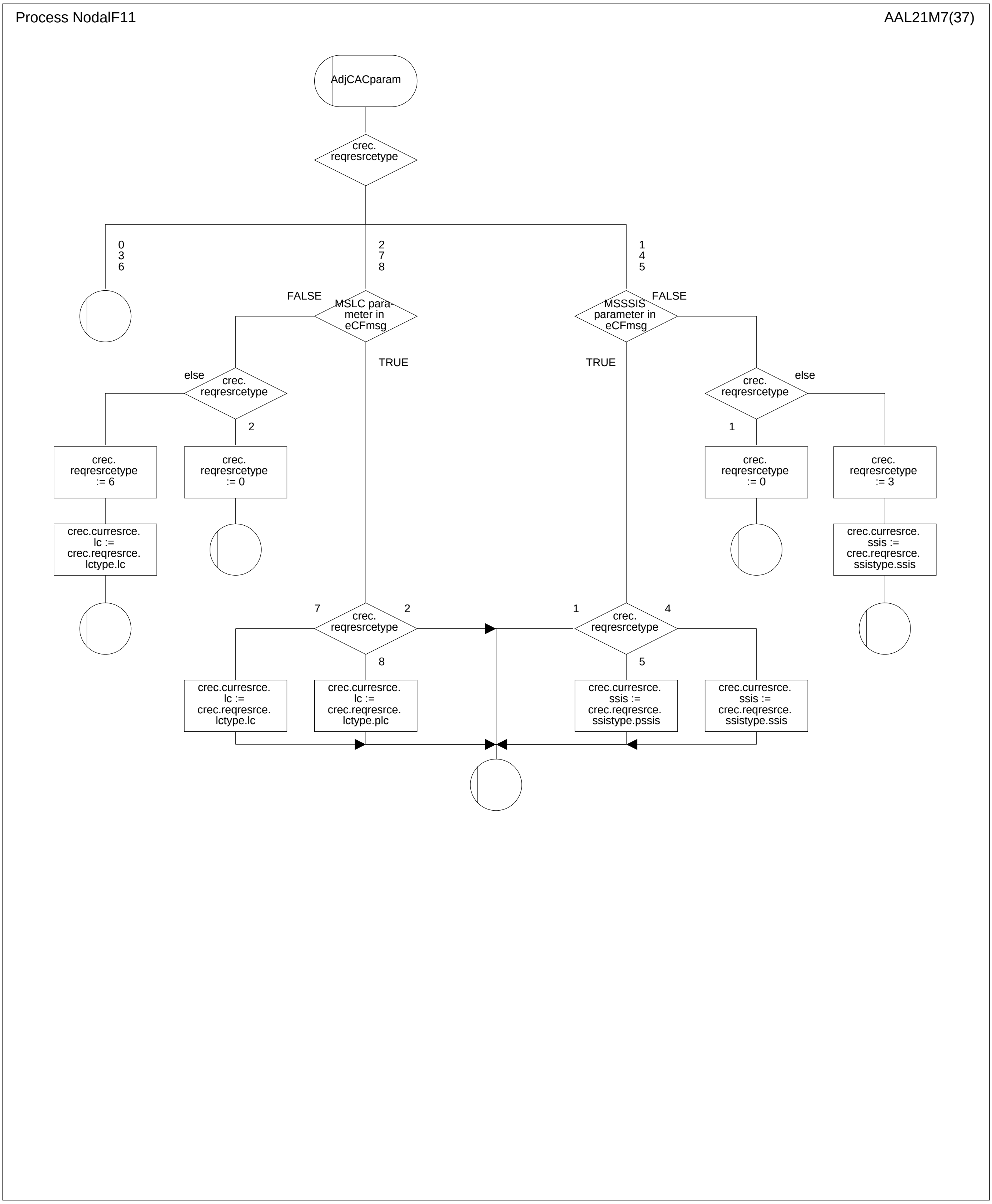


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 34 of 37)***



Process NodalF11
AAL21M7(37)
AdjCACparam
crec. reqresrcetype
0 3 6
2 7 8
1 4 5
FALSE
MSLC parameter in eCFmsg
TRUE
MSSSIS parameter in eCFmsg
FALSE
TRUE
else
crec. reqresrcetype
2
crec. reqresrcetype
else
1
crec. reqresrcetype := 6
crec. reqresrcetype := 0
crec. reqresrcetype := 0
crec. reqresrcetype := 3
crec.curresrce. lc := crec.reqresrce. lctype.lc
crec.curresrce. ssis := crec.reqresrce. ssistype.ssis
7
2
crec. reqresrcetype
8
1
4
crec. reqresrcetype
5
crec.curresrce. lc := crec.reqresrce. lctype.lc
crec.curresrce. lc := crec.reqresrce. lctype.plc
crec.curresrce. ssis := crec.reqresrce. ssistype.pssis
crec.curresrce. ssis := crec.reqresrce. ssistype.ssis

**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 35 of 37)***

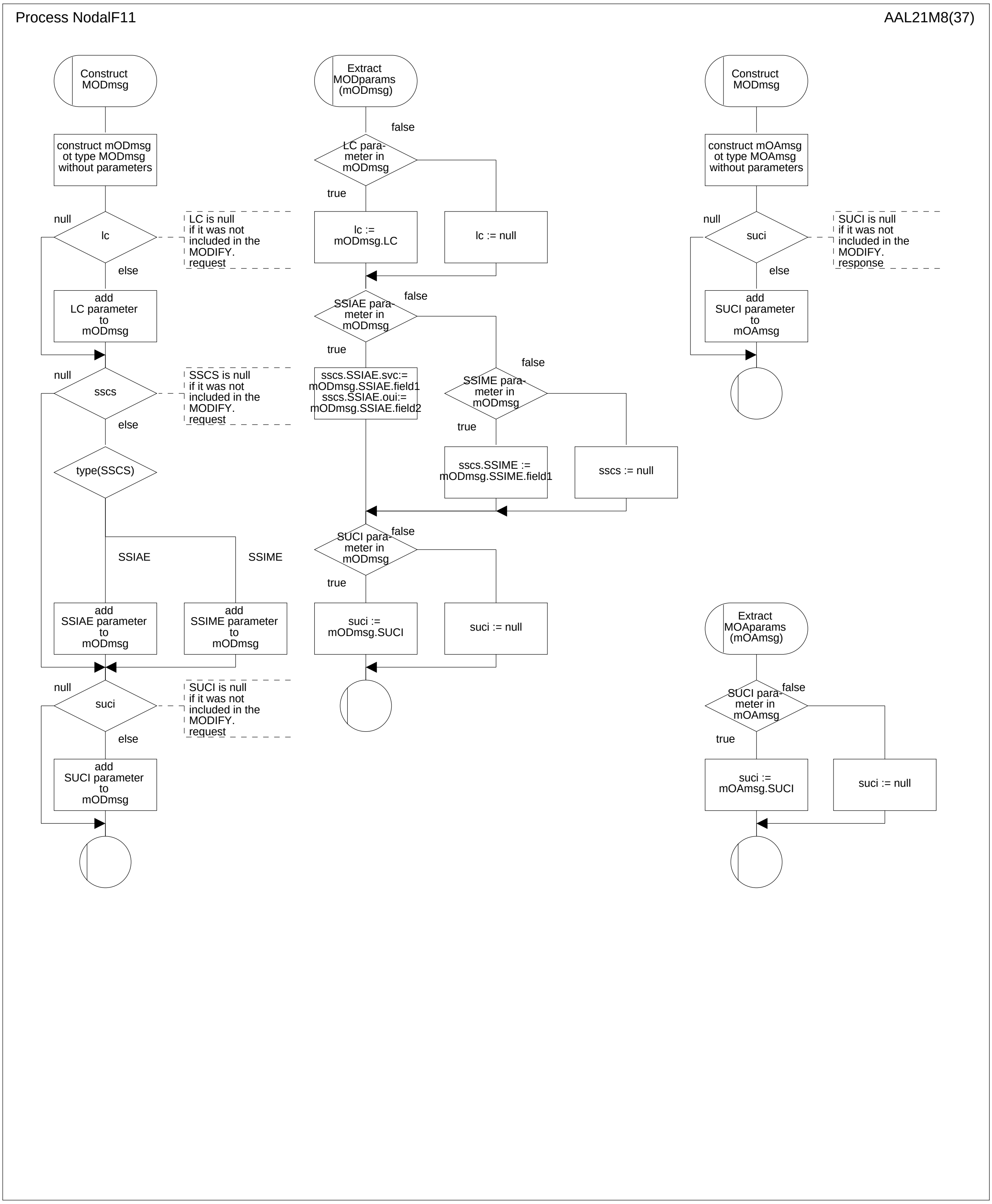

Process NodalF11
AAL21M8(37)
Construct MODmsg
construct mODmsg ot type MODmsg without parameters
null
lc
else
LC is null if it was not included in the MODIFY. request
add LC parameter to mODmsg
null
sscs
else
SSCS is null if it was not included in the MODIFY. request
type(SSCS)
SSIAE
SSIME
add SSIAE parameter to mODmsg
add SSIME parameter to mODmsg
null
suci
else
SUCI is null if it was not included in the MODIFY. request
add SUCI parameter to mODmsg
Extract MODparams (mODmsg)
false
LC para-meter in mODmsg
true
lc := mODmsg.LC
lc := null
SSIAE para-meter in mODmsg
false
true
sscs.SSIAE.svc:= mODmsg.SSIAE.field1 sscs.SSIAE.oui:= mODmsg.SSIAE.field2
false
SSIME para-meter in mODmsg
true
sscs.SSIME := mODmsg.SSIME.field1
sscs := null
SUCI para-meter in mODmsg
false
true
suci := mODmsg.SUCI
suci := null
Construct MODmsg
construct mOAmsg ot type MOAmsg without parameters
null
suci
else
SUCI is null if it was not included in the MODIFY. response
add SUCI parameter to mOAmsg
Extract MOAparams (mOAmsg)
SUCI para-meter in mOAmsg
false
true
suci := mOAmsg.SUCI
suci := null

Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros) *(part 36 of 37)*

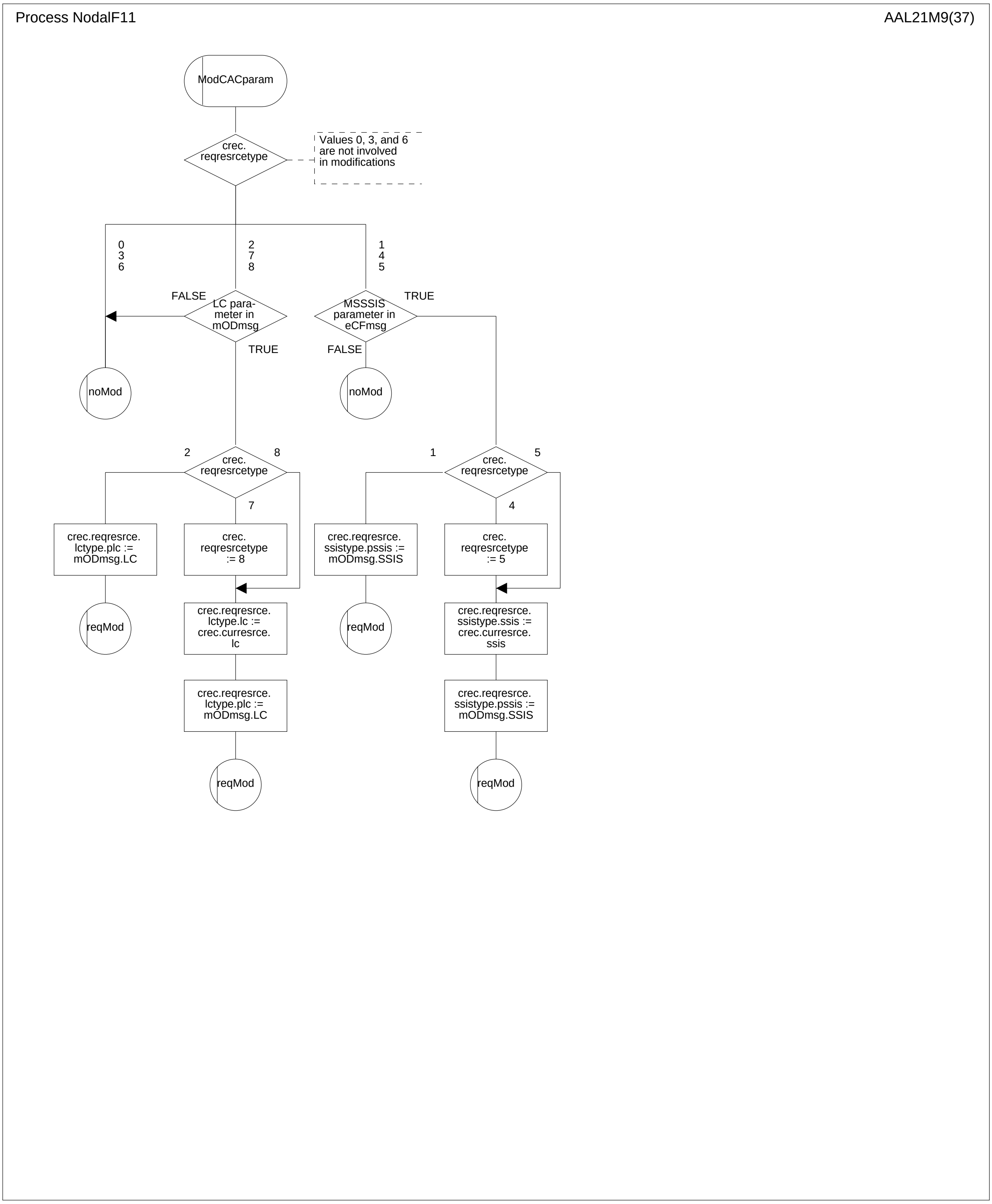


**Figure D.3/Q.2630.2 – SDL diagram of the nodal function 1 (macros)** ***(part 37 of 37)***

## D.5 SDL diagrams for the protocol entities

### D.5.1 Introduction

B.5.1/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D.

### D.5.2 SDL diagrams for the outgoing, incoming, and maintenance protocol procedures

#### D.5.2.1 Data structures

B.5.2.1/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D.

#### D.5.2.2 SDL diagrams for the outgoing protocol procedures

B.5.2.2/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modification is required:

The SDL diagram for the outgoing protocol procedure is described in parts 1 to 8 in Figure D.4/Q.2630.2.

#### D.5.2.3 SDL diagrams for the incoming protocol procedures

B.5.2.3/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. Further, the following modifications are required:

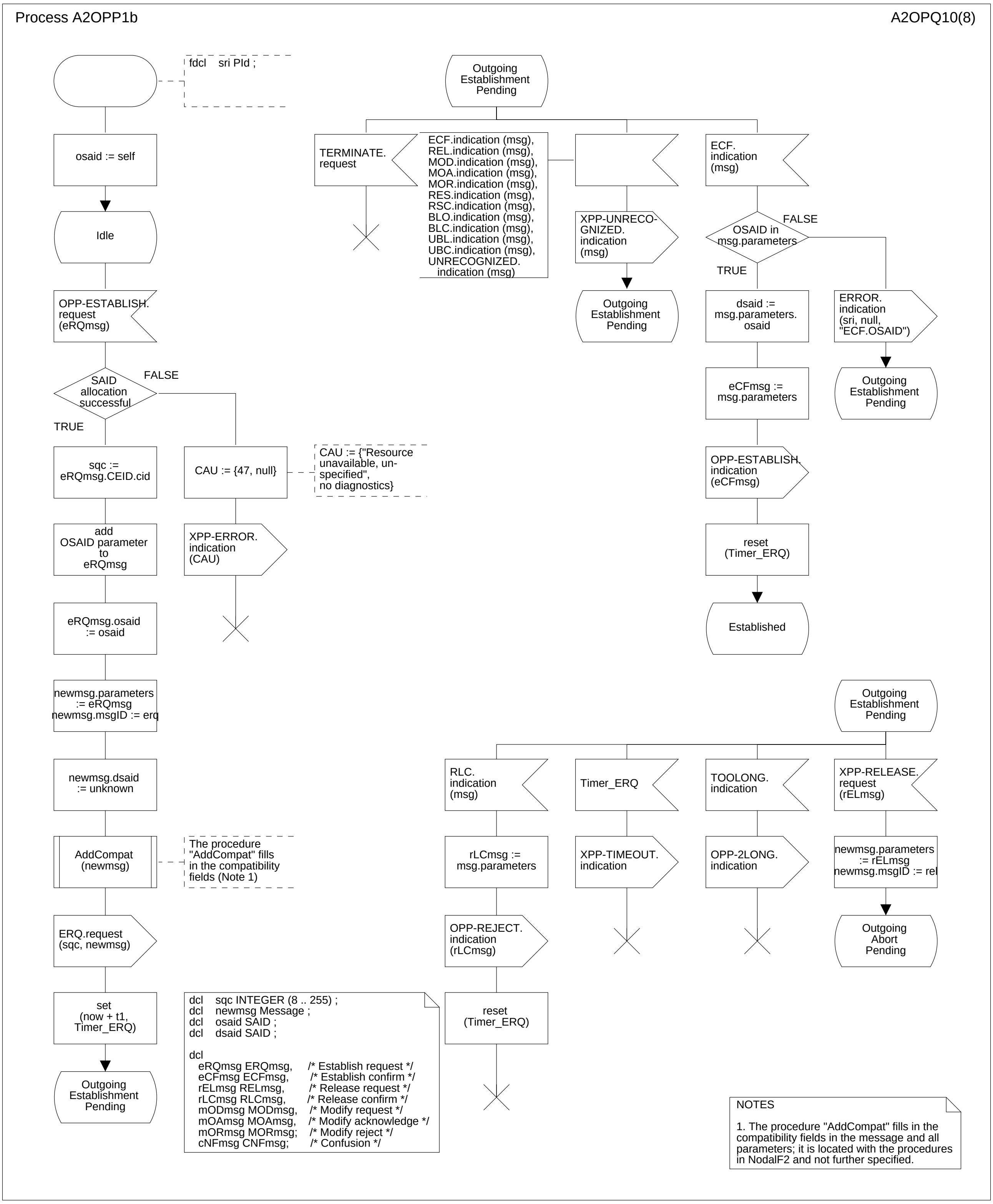
The SDL diagrams for the incoming protocol procedure is described in parts 1 to 8 in Figure D.5/Q.2630.2.

#### D.5.2.4 SDL diagrams for the maintenance protocol procedures

B.5.2.4/Q.2630.1 applies, replacing any reference to subclauses, figures, or tables to Annex B by references to Annex D. The SDL diagrams for the maintenance protocol procedure are described in parts 1 to 5 in Figure D.6/Q.2630.2.

#### D.5.2.5 SDL diagrams for the signalling transport interface

The SDL diagrams for the signalling transport interface is described in parts 1 to 3 in Figure D.7/Q.2630.2.


Process A2OPP1b
A2OPQ10(8)
fdcl sri PId ;
osaid := self
Idle
OPP-ESTABLISH. request (eRQmsg)
SAID allocation successful
FALSE
TRUE
sqc := eRQmsg.CEID.cid
add OSAID parameter to eRQmsg
eRQmsg.osaid := osaid
newmsg.parameters := eRQmsg newmsg.msgID := erq
newmsg.dsaid := unknown
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
ERQ.request (sqc, newmsg)
set (now + t1, Timer_ERQ)
Outgoing Establishment Pending
CAU := {47, null}
CAU := {"Resource unavailable, un-specified", no diagnostics}
XPP-ERROR. indication (CAU)
Outgoing Establishment Pending
TERMINATE. request
ECF.indication (msg), REL.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECO-GNIZED. indication (msg)
Outgoing Establishment Pending
ECF. indication (msg)
OSAID in msg.parameters
FALSE
TRUE
dsaid := msg.parameters. osaid
eCFmsg := msg.parameters
OPP-ESTABLISH. indication (eCFmsg)
reset (Timer_ERQ)
Established
ERROR. indication (sri, null, "ECF.OSAID")
Outgoing Establishment Pending
Outgoing Establishment Pending
RLC. indication (msg)
rLCmsg := msg.parameters
OPP-REJECT. indication (rLCmsg)
reset (Timer_ERQ)
Timer_ERQ
XPP-TIMEOUT. indication
TOOLONG. indication
OPP-2LONG. indication
XPP-RELEASE. request (rELmsg)
newmsg.parameters := rELmsg newmsg.msgID := rel
Outgoing Abort Pending

```
dcl    sqc INTEGER (8 .. 255) ;
dcl    newmsg Message ;
dcl    osaid SAID ;
dcl    dsaid SAID ;

dcl
   eRQmsg ERQmsg,      /* Establish request */
   eCFmsg ECFmsg,      /* Establish confirm */
   rELmsg RELmsg,      /* Release request */
   rLCmsg RLCmsg,      /* Release confirm */
   mODmsg MODmsg,      /* Modify request */
   mOAmsg MOAmsg,      /* Modify acknowledge */
   mORmsg MORmsg;      /* Modify reject */
   cNFmsg CNFmsg;      /* Confusion */
```

NOTES

1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 1 of 8)***

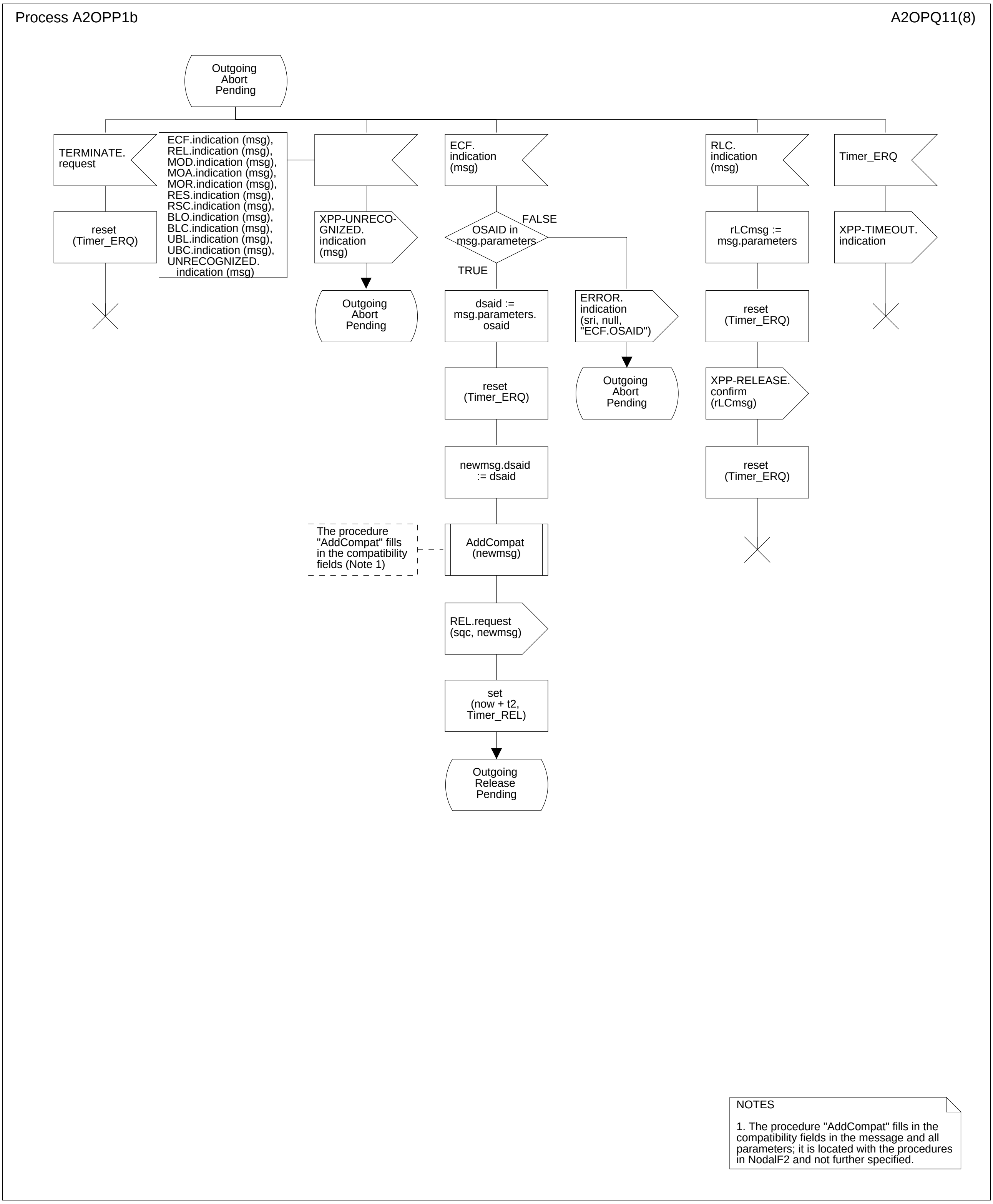
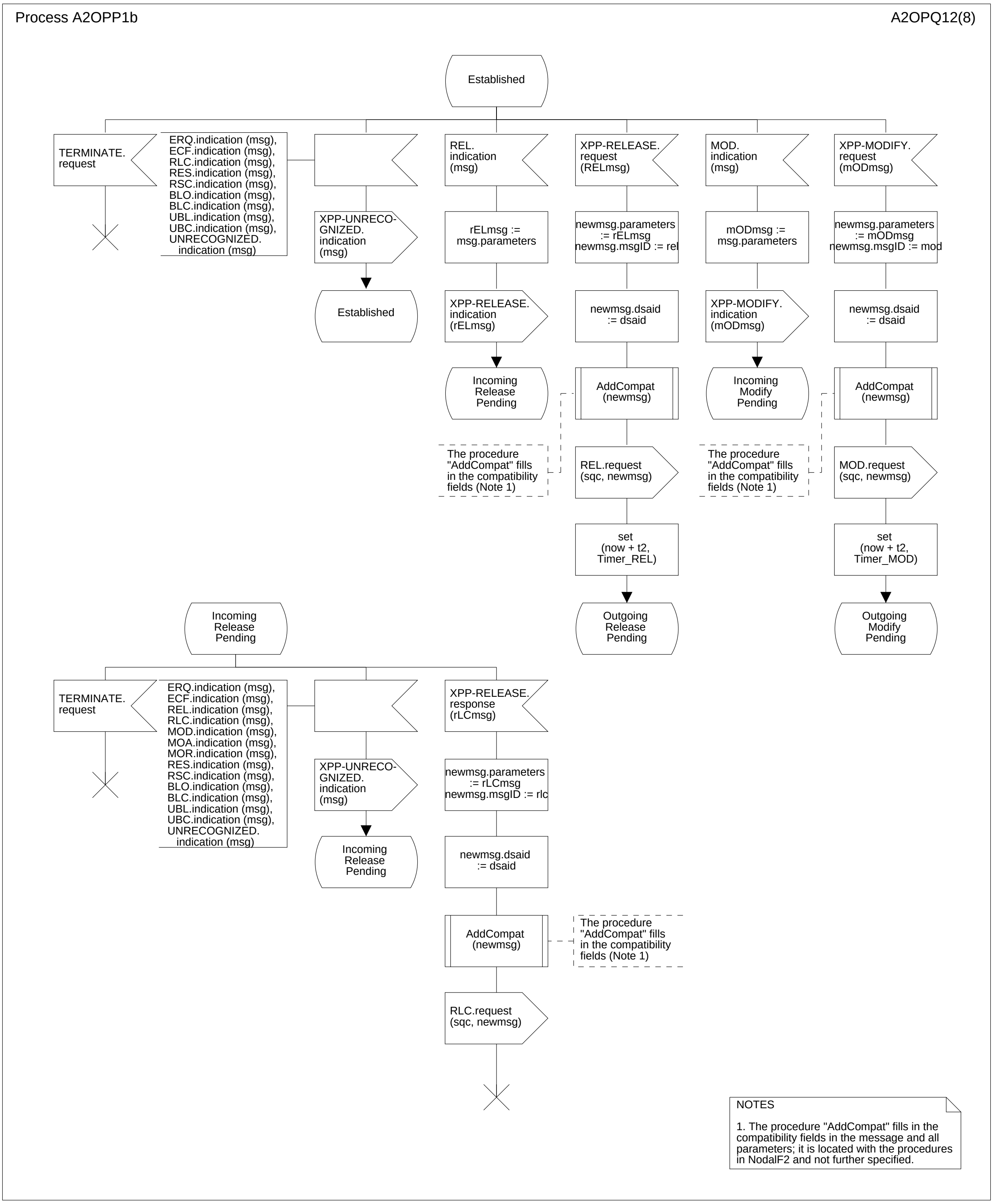

Process A2OPP1b
A2OPQ11(8)
Outgoing Abort Pending
TERMINATE. request
reset (Timer_ERQ)
ECF.indication (msg), REL.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECO-GNIZED. indication (msg)
Outgoing Abort Pending
ECF. indication (msg)
OSAID in msg.parameters
FALSE
TRUE
dsaid := msg.parameters. osaid
reset (Timer_ERQ)
newmsg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
REL.request (sqc, newmsg)
set (now + t2, Timer_REL)
Outgoing Release Pending
ERROR. indication (sri, null, "ECF.OSAID")
Outgoing Abort Pending
RLC. indication (msg)
rLCmsg := msg.parameters
reset (Timer_ERQ)
XPP-RELEASE. confirm (rLCmsg)
reset (Timer_ERQ)
Timer_ERQ
XPP-TIMEOUT. indication
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 2 of 8)***


Process A2OPP1b
A2OPQ12(8)
Established
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECOGNIZED. indication (msg)
Established
REL. indication (msg)
rELmsg := msg.parameters
XPP-RELEASE. indication (rELmsg)
Incoming Release Pending
XPP-RELEASE. request (RELmsg)
newmsg.parameters := rELmsg newmsg.msgID := rel
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
REL.request (sqc, newmsg)
set (now + t2, Timer_REL)
Outgoing Release Pending
MOD. indication (msg)
mODmsg := msg.parameters
XPP-MODIFY. indication (mODmsg)
Incoming Modify Pending
The procedure "AddCompat" fills in the compatibility fields (Note 1)
XPP-MODIFY. request (mODmsg)
newmsg.parameters := mODmsg newmsg.msgID := mod
newmsg.dsaid := dsaid
AddCompat (newmsg)
MOD.request (sqc, newmsg)
set (now + t2, Timer_MOD)
Outgoing Modify Pending
Incoming Release Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECOGNIZED. indication (msg)
Incoming Release Pending
XPP-RELEASE. response (rLCmsg)
newmsg.parameters := rLCmsg newmsg.msgID := rlc
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
RLC.request (sqc, newmsg)

NOTES

1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 3 of 8)***

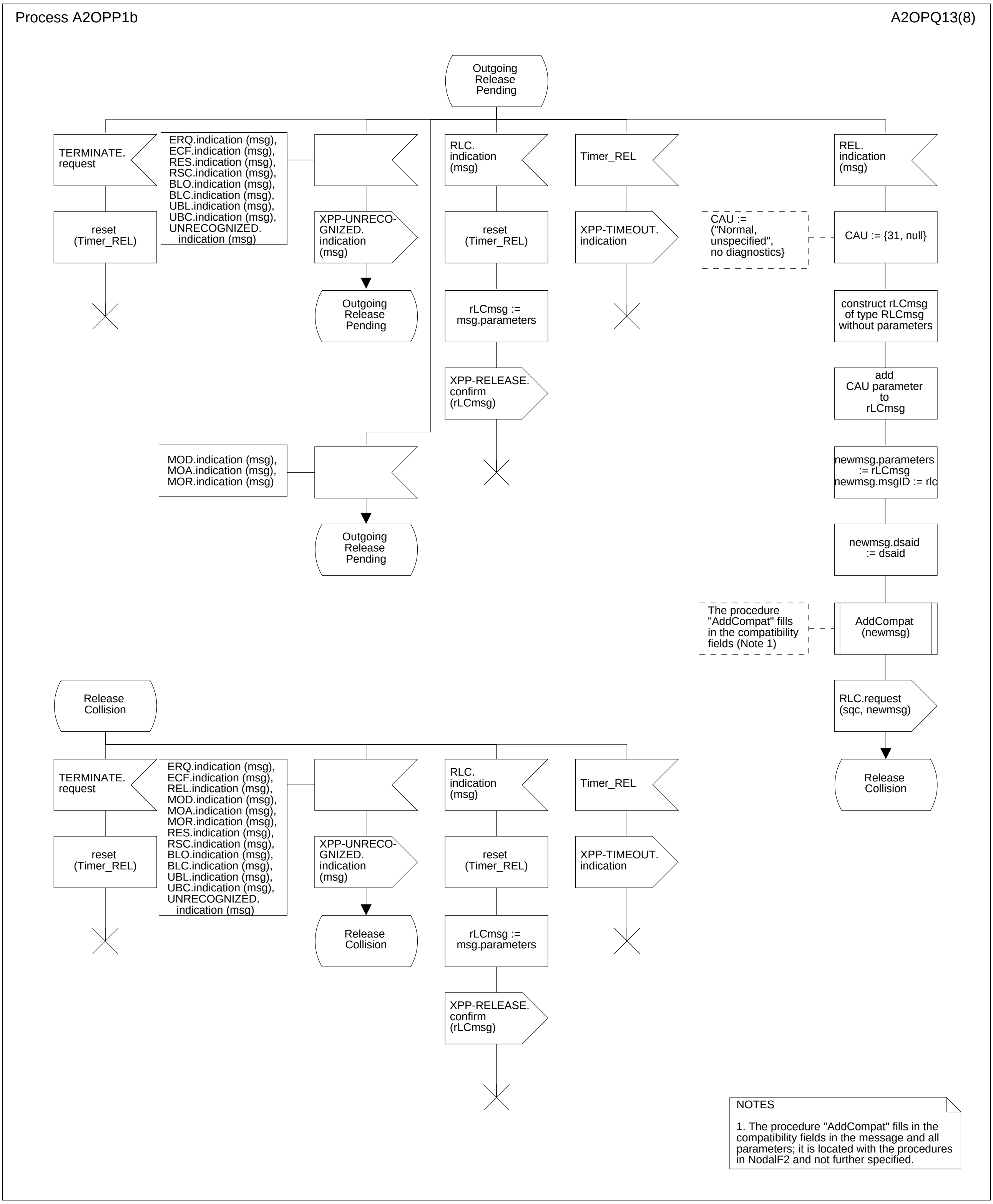
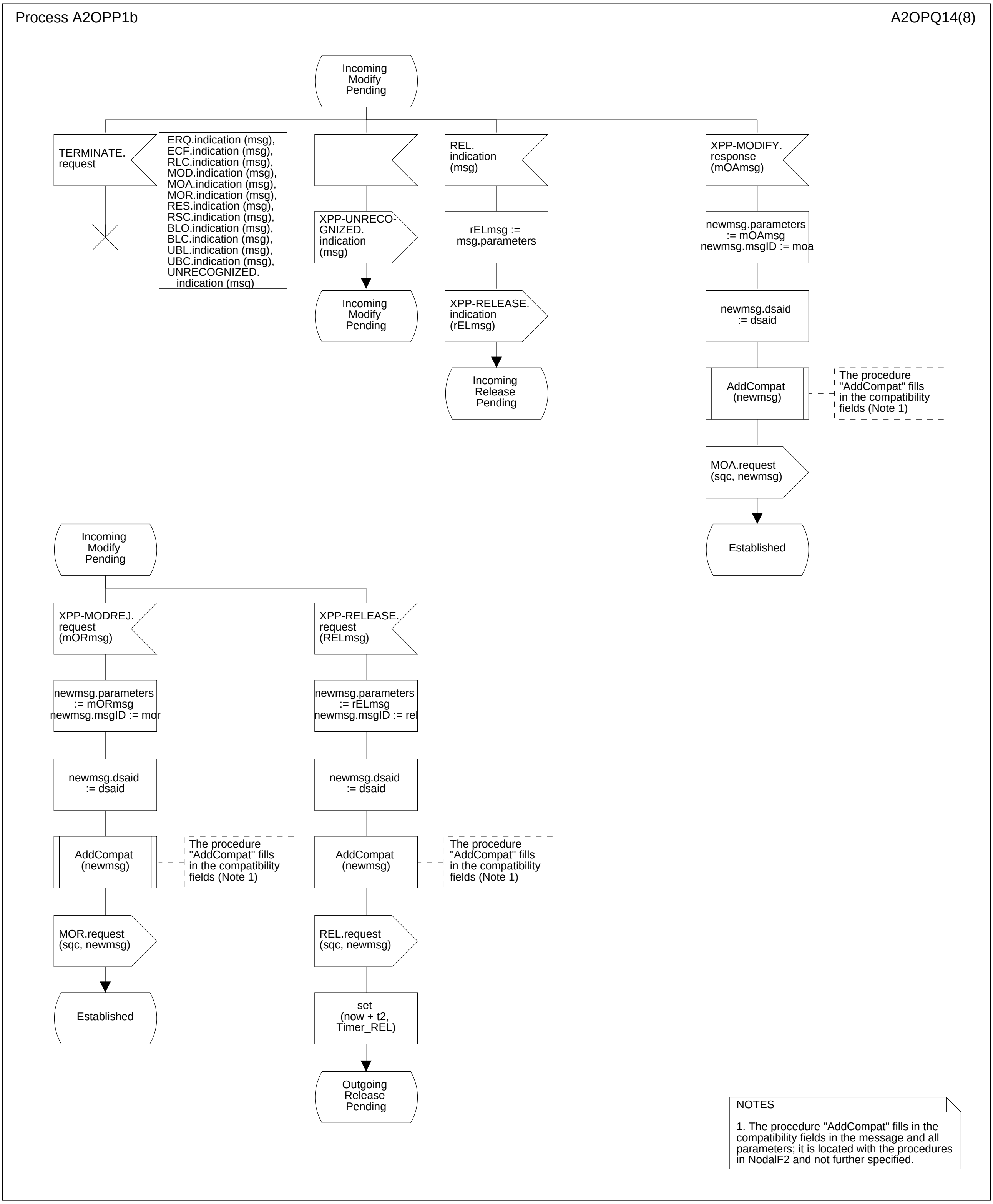

Process A2OPP1b
A2OPQ13(8)
Outgoing Release Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
RLC. indication (msg)
Timer_REL
REL. indication (msg)
reset (Timer_REL)
XPP-UNRECO-GNIZED. indication (msg)
reset (Timer_REL)
XPP-TIMEOUT. indication
CAU := ("Normal, unspecified", no diagnostics}
CAU := {31, null}
Outgoing Release Pending
rLCmsg := msg.parameters
construct rLCmsg of type RLCmsg without parameters
XPP-RELEASE. confirm (rLCmsg)
add CAU parameter to rLCmsg
MOD.indication (msg), MOA.indication (msg), MOR.indication (msg)
newmsg.parameters := rLCmsg newmsg.msgID := rlc
Outgoing Release Pending
newmsg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
Release Collision
RLC.request (sqc, newmsg)
Release Collision
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
RLC. indication (msg)
Timer_REL
reset (Timer_REL)
XPP-UNRECO-GNIZED. indication (msg)
reset (Timer_REL)
XPP-TIMEOUT. indication
Release Collision
rLCmsg := msg.parameters
XPP-RELEASE. confirm (rLCmsg)
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

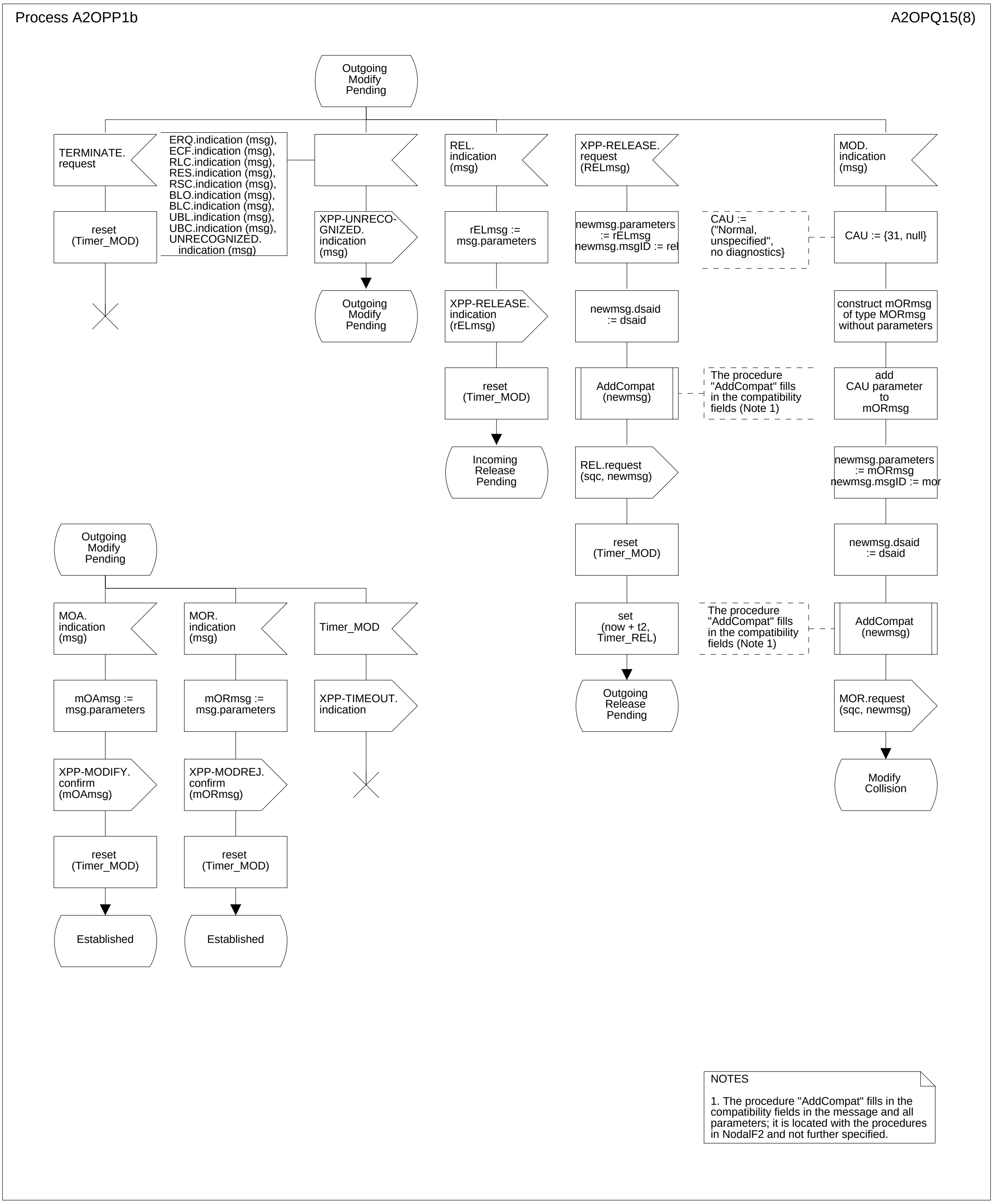
**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 4 of 8)***


Process A2OPP1b
A2OPQ14(8)
Incoming Modify Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECO- GNIZED. indication (msg)
Incoming Modify Pending
REL. indication (msg)
rELmsg := msg.parameters
XPP-RELEASE. indication (rELmsg)
Incoming Release Pending
XPP-MODIFY. response (mOAmsg)
newmsg.parameters := mOAmsg newmsg.msgID := moa
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
MOA.request (sqc, newmsg)
Established
Incoming Modify Pending
XPP-MODREJ. request (mORmsg)
newmsg.parameters := mORmsg newmsg.msgID := mor
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
MOR.request (sqc, newmsg)
Established
XPP-RELEASE. request (RELmsg)
newmsg.parameters := rELmsg newmsg.msgID := rel
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
REL.request (sqc, newmsg)
set (now + t2, Timer_REL)
Outgoing Release Pending

NOTES

1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 5 of 8)***


Process A2OPP1b
A2OPQ15(8)
Outgoing Modify Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
REL. indication (msg)
XPP-RELEASE. request (RELmsg)
MOD. indication (msg)
reset (Timer_MOD)
XPP-UNRECO- GNIZED. indication (msg)
rELmsg := msg.parameters
newmsg.parameters := rELmsg newmsg.msgID := rel
CAU := ("Normal, unspecified", no diagnostics}
CAU := {31, null}
Outgoing Modify Pending
XPP-RELEASE. indication (rELmsg)
newmsg.dsaid := dsaid
construct mORmsg of type MORmsg without parameters
reset (Timer_MOD)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
add CAU parameter to mORmsg
Incoming Release Pending
REL.request (sqc, newmsg)
newmsg.parameters := mORmsg newmsg.msgID := mor
Outgoing Modify Pending
reset (Timer_MOD)
newmsg.dsaid := dsaid
MOA. indication (msg)
MOR. indication (msg)
Timer_MOD
set (now + t2, Timer_REL)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
mOAmsg := msg.parameters
mORmsg := msg.parameters
XPP-TIMEOUT. indication
Outgoing Release Pending
MOR.request (sqc, newmsg)
XPP-MODIFY. confirm (mOAmsg)
XPP-MODREJ. confirm (mORmsg)
Modify Collision
reset (Timer_MOD)
reset (Timer_MOD)
Established
Established

NOTES

1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 6 of 8)***

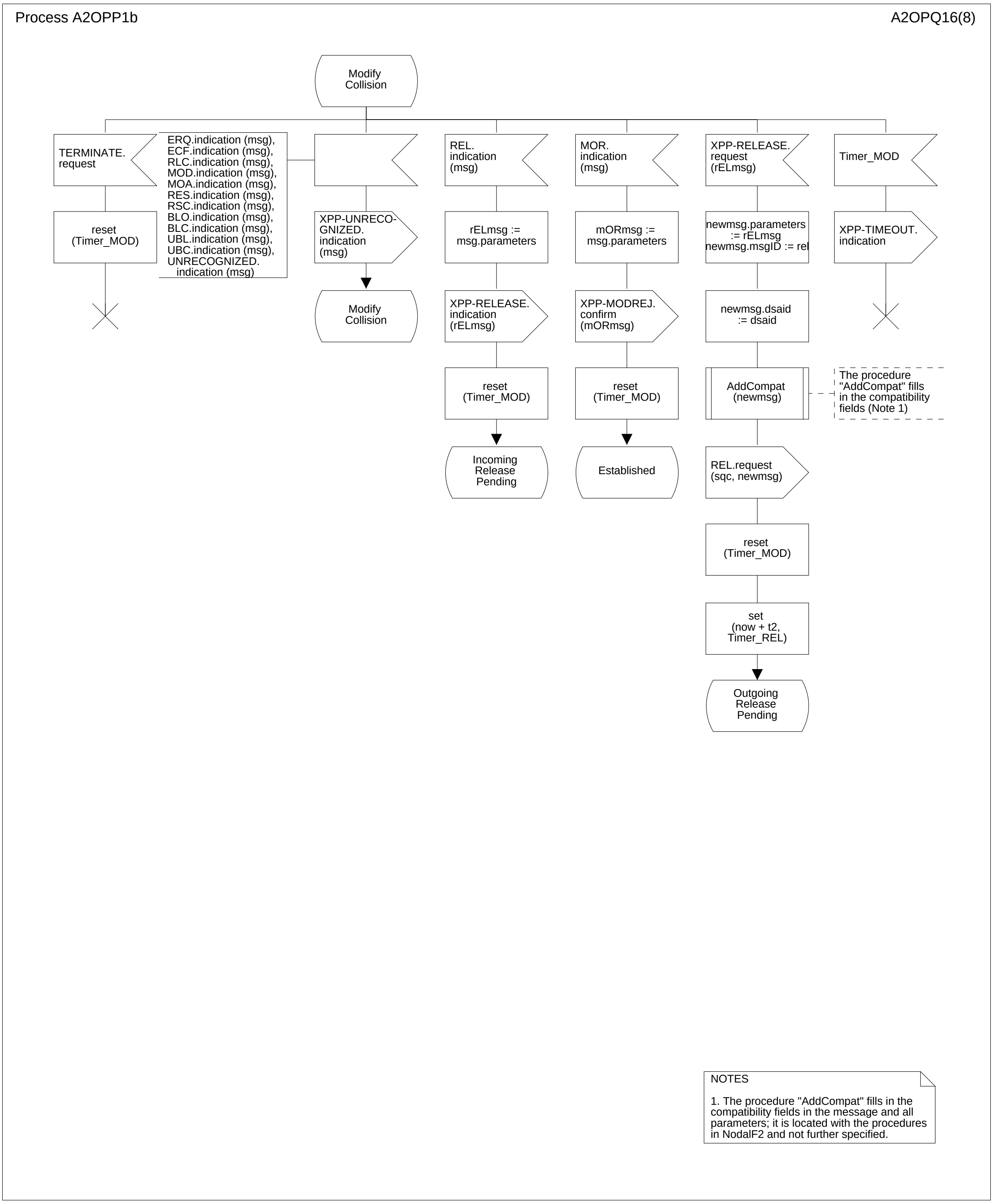

Process A2OPP1b
A2OPQ16(8)
Modify Collision
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
REL. indication (msg)
MOR. indication (msg)
XPP-RELEASE. request (rELmsg)
Timer_MOD
reset (Timer_MOD)
XPP-UNRECOGNIZED. indication (msg)
rELmsg := msg.parameters
mORmsg := msg.parameters
newmsg.parameters := rELmsg newmsg.msgID := rel
XPP-TIMEOUT. indication
Modify Collision
XPP-RELEASE. indication (rELmsg)
XPP-MODREJ. confirm (mORmsg)
newmsg.dsaid := dsaid
reset (Timer_MOD)
reset (Timer_MOD)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
Incoming Release Pending
Established
REL.request (sqc, newmsg)
reset (Timer_MOD)
set (now + t2, Timer_REL)
Outgoing Release Pending
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 7 of 8)***

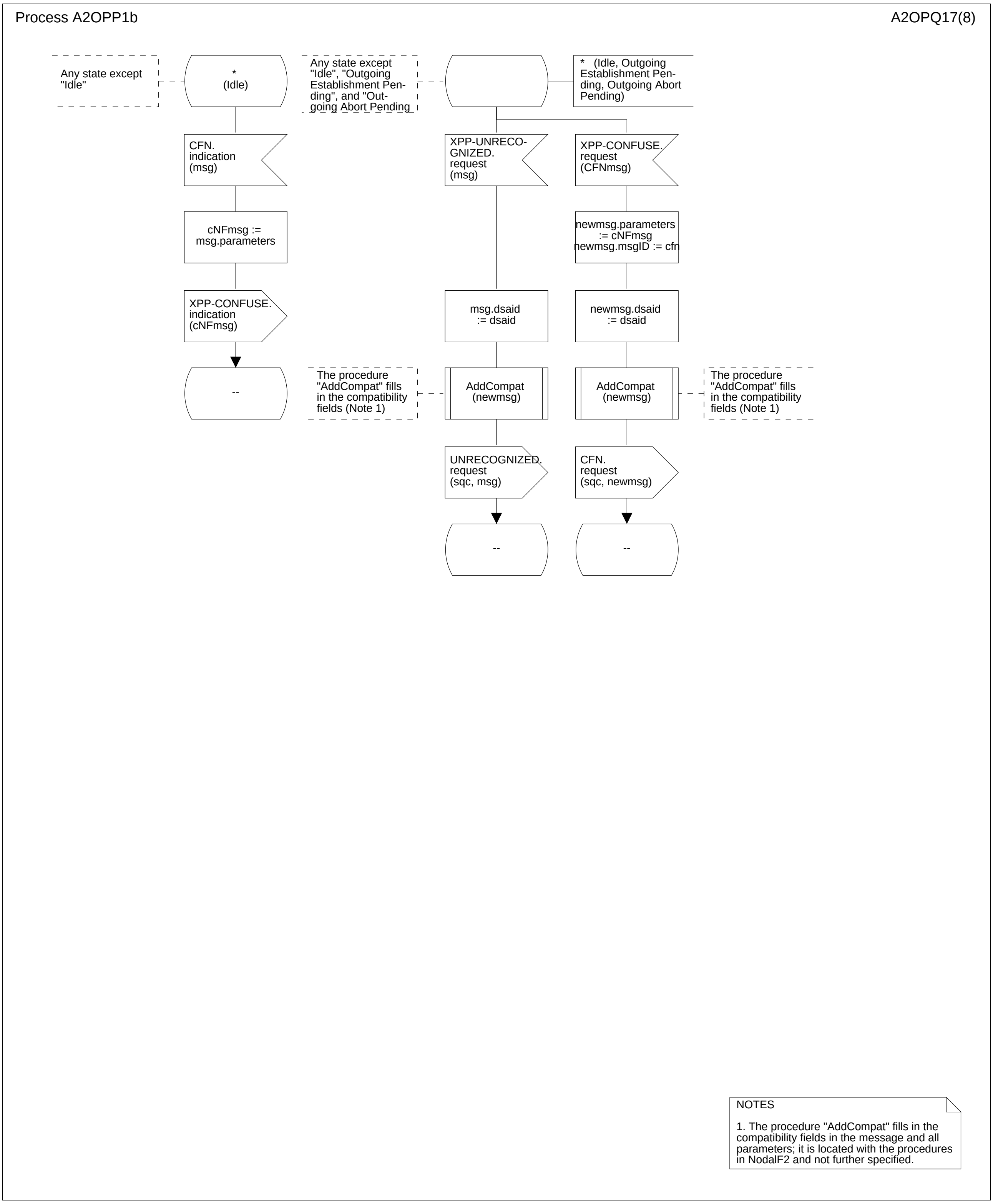
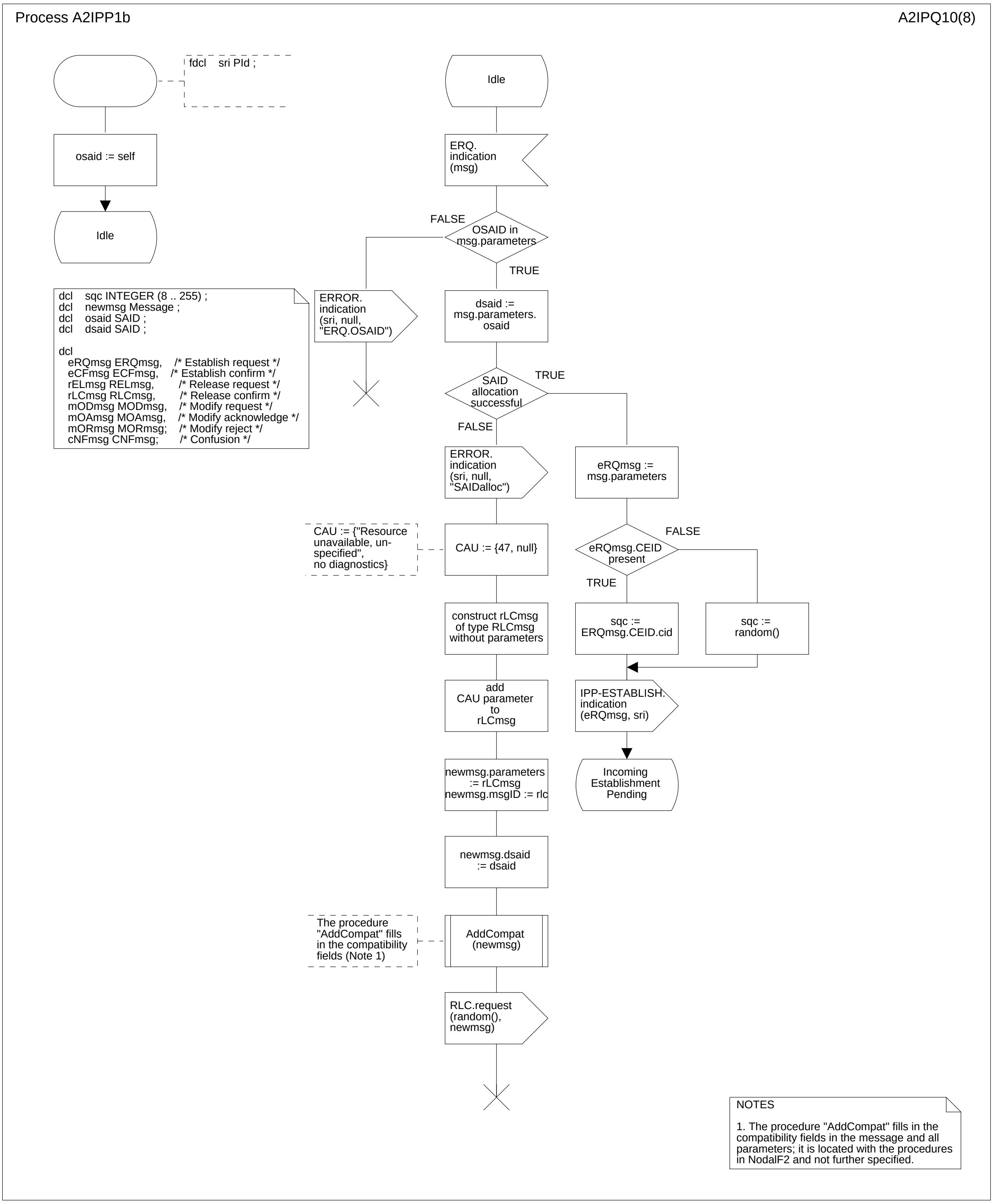

Process A2OPP1b
A2OPQ17(8)
Any state except "Idle"
* (Idle)
Any state except "Idle", "Outgoing Establishment Pending", and "Outgoing Abort Pending
* (Idle, Outgoing Establishment Pending, Outgoing Abort Pending)
CFN. indication (msg)
cNFmsg := msg.parameters
XPP-CONFUSE. indication (cNFmsg)
--
XPP-UNRECOGNIZED. request (msg)
msg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
UNRECOGNIZED. request (sqc, msg)
--
XPP-CONFUSE. request (CFNmsg)
newmsg.parameters := cNFmsg newmsg.msgID := cfn
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
CFN. request (sqc, newmsg)
--
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.4/Q.2630.2 – SDL diagram of the outgoing protocol procedure *(part 8 of 8)***


Process A2IPP1b
A2IPQ10(8)
fdcl sri PId ;
osaid := self
Idle
Idle
ERQ. indication (msg)
FALSE
OSAID in msg.parameters
TRUE
ERROR. indication (sri, null, "ERQ.OSAID")
dsaid := msg.parameters. osaid
SAID allocation successful
TRUE
FALSE
ERROR. indication (sri, null, "SAIDalloc")
eRQmsg := msg.parameters
CAU := {"Resource unavailable, un-specified", no diagnostics}
CAU := {47, null}
eRQmsg.CEID present
FALSE
TRUE
construct rLCmsg of type RLCmsg without parameters
sqc := ERQmsg.CEID.cid
sqc := random()
add CAU parameter to rLCmsg
IPP-ESTABLISH. indication (eRQmsg, sri)
newmsg.parameters := rLCmsg newmsg.msgID := rlc
Incoming Establishment Pending
newmsg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
RLC.request (random(), newmsg)

```
dcl    sqc INTEGER (8 .. 255) ;
dcl    newmsg Message ;
dcl    osaid SAID ;
dcl    dsaid SAID ;

dcl
   eRQmsg ERQmsg,    /* Establish request */
   eCFmsg ECFmsg,    /* Establish confirm */
   rELmsg RELmsg,        /* Release request */
   rLCmsg RLCmsg,        /* Release confirm */
   mODmsg MODmsg,    /* Modify request */
   mOAmsg MOAmsg,    /* Modify acknowledge */
   mORmsg MORmsg;    /* Modify reject */
   cNFmsg CNFmsg;      /* Confusion */
```

NOTES

1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 1 of 8)***

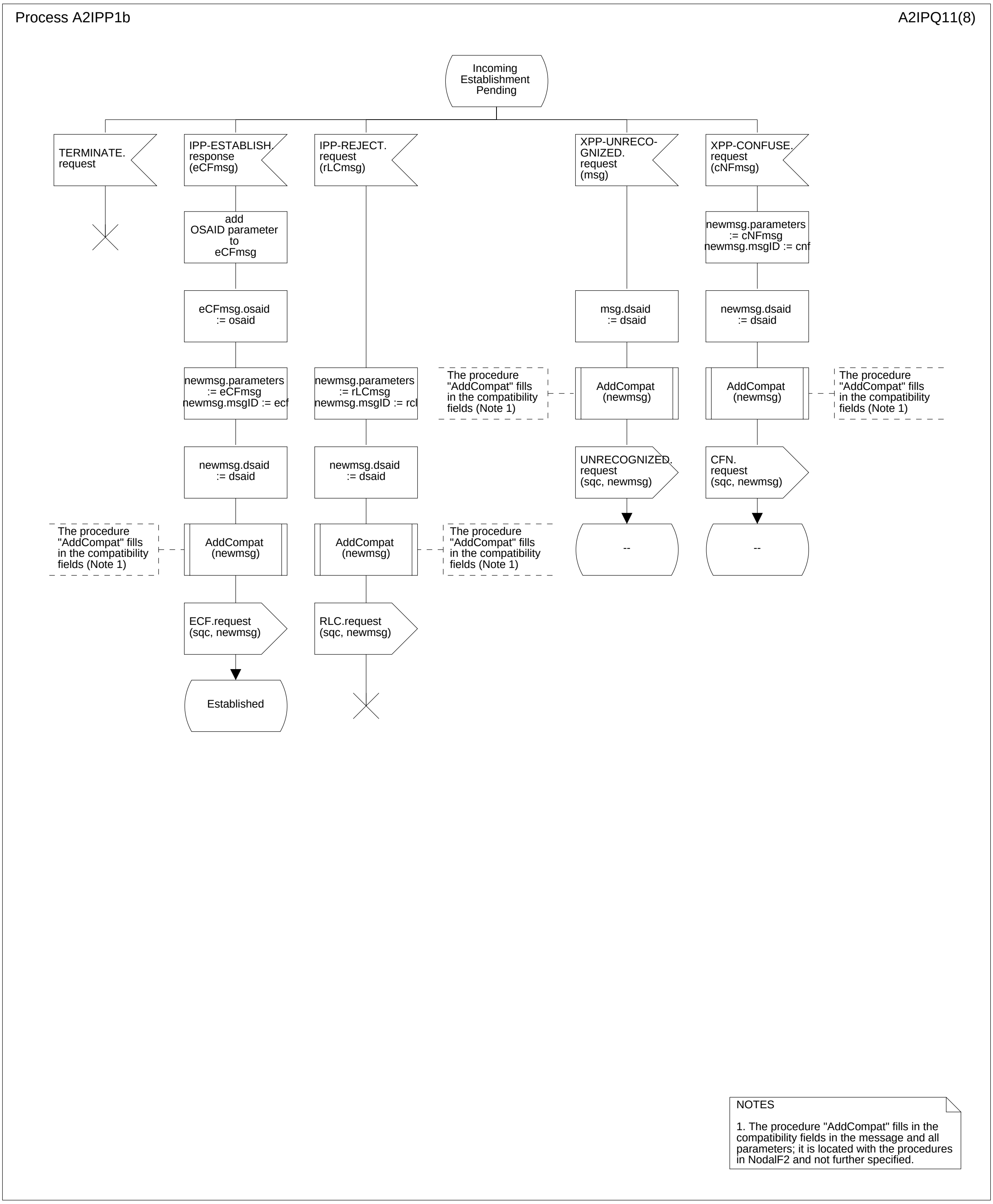

Process A2IPP1b
A2IPQ11(8)
Incoming Establishment Pending
TERMINATE. request
IPP-ESTABLISH. response (eCFmsg)
IPP-REJECT. request (rLCmsg)
XPP-UNRECO-GNIZED. request (msg)
XPP-CONFUSE. request (cNFmsg)
add OSAID parameter to eCFmsg
newmsg.parameters := cNFmsg newmsg.msgID := cnf
eCFmsg.osaid := osaid
msg.dsaid := dsaid
newmsg.dsaid := dsaid
newmsg.parameters := eCFmsg newmsg.msgID := ecf
newmsg.parameters := rLCmsg newmsg.msgID := rcl
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
newmsg.dsaid := dsaid
newmsg.dsaid := dsaid
UNRECOGNIZED. request (sqc, newmsg)
CFN. request (sqc, newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
--
--
ECF.request (sqc, newmsg)
RLC.request (sqc, newmsg)
Established
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 2 of 8)***

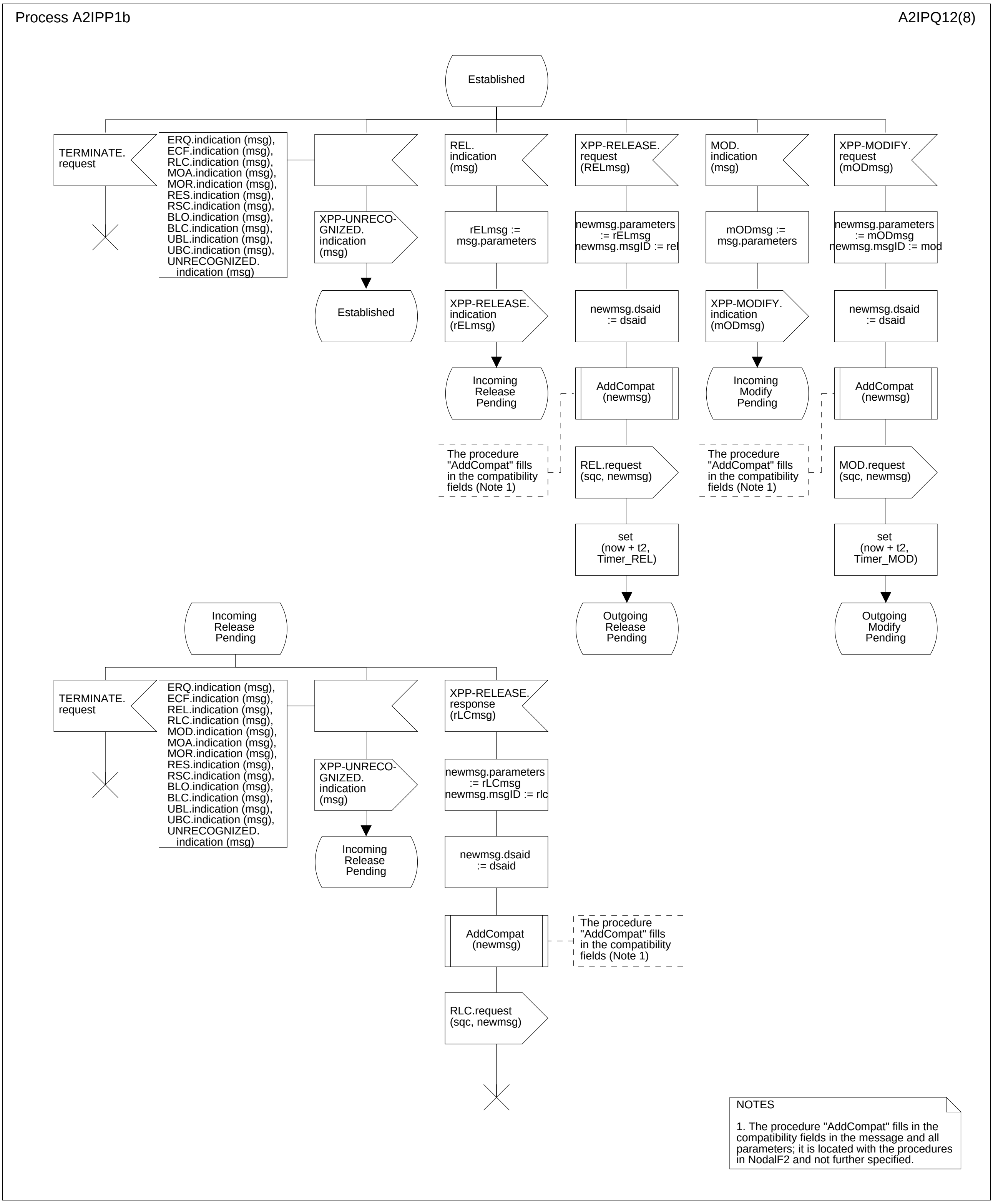
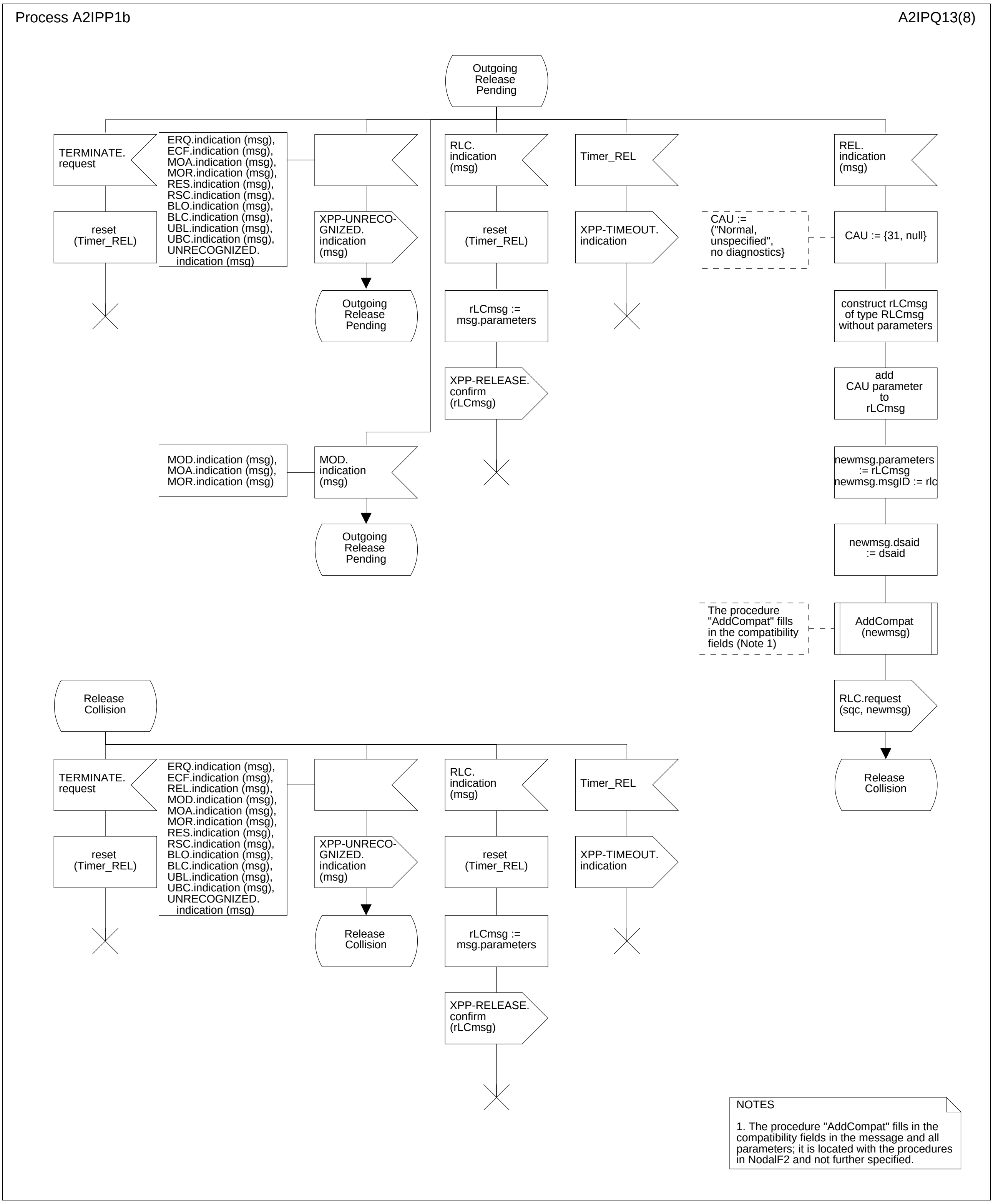

Process A2IPP1b
A2IPQ12(8)
Established
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECO-GNIZED. indication (msg)
Established
REL. indication (msg)
rELmsg := msg.parameters
XPP-RELEASE. indication (rELmsg)
Incoming Release Pending
The procedure "AddCompat" fills in the compatibility fields (Note 1)
XPP-RELEASE. request (RELmsg)
newmsg.parameters := rELmsg newmsg.msgID := rel
newmsg.dsaid := dsaid
AddCompat (newmsg)
REL.request (sqc, newmsg)
set (now + t2, Timer_REL)
Outgoing Release Pending
MOD. indication (msg)
mODmsg := msg.parameters
XPP-MODIFY. indication (mODmsg)
Incoming Modify Pending
The procedure "AddCompat" fills in the compatibility fields (Note 1)
XPP-MODIFY. request (mODmsg)
newmsg.parameters := mODmsg newmsg.msgID := mod
newmsg.dsaid := dsaid
AddCompat (newmsg)
MOD.request (sqc, newmsg)
set (now + t2, Timer_MOD)
Outgoing Modify Pending
Incoming Release Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECO-GNIZED. indication (msg)
Incoming Release Pending
XPP-RELEASE. response (rLCmsg)
newmsg.parameters := rLCmsg newmsg.msgID := rlc
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
RLC.request (sqc, newmsg)
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 3 of 8)***


Process A2IPP1b
A2IPQ13(8)
Outgoing Release Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
RLC. indication (msg)
Timer_REL
REL. indication (msg)
reset (Timer_REL)
XPP-UNRECOGNIZED. indication (msg)
reset (Timer_REL)
XPP-TIMEOUT. indication
CAU := ("Normal, unspecified", no diagnostics}
CAU := {31, null}
Outgoing Release Pending
rLCmsg := msg.parameters
construct rLCmsg of type RLCmsg without parameters
XPP-RELEASE. confirm (rLCmsg)
add CAU parameter to rLCmsg
MOD.indication (msg), MOA.indication (msg), MOR.indication (msg)
MOD. indication (msg)
newmsg.parameters := rLCmsg newmsg.msgID := rlc
Outgoing Release Pending
newmsg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
Release Collision
RLC.request (sqc, newmsg)
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
RLC. indication (msg)
Timer_REL
Release Collision
reset (Timer_REL)
XPP-UNRECOGNIZED. indication (msg)
reset (Timer_REL)
XPP-TIMEOUT. indication
Release Collision
rLCmsg := msg.parameters
XPP-RELEASE. confirm (rLCmsg)

NOTES

1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 4 of 8)***

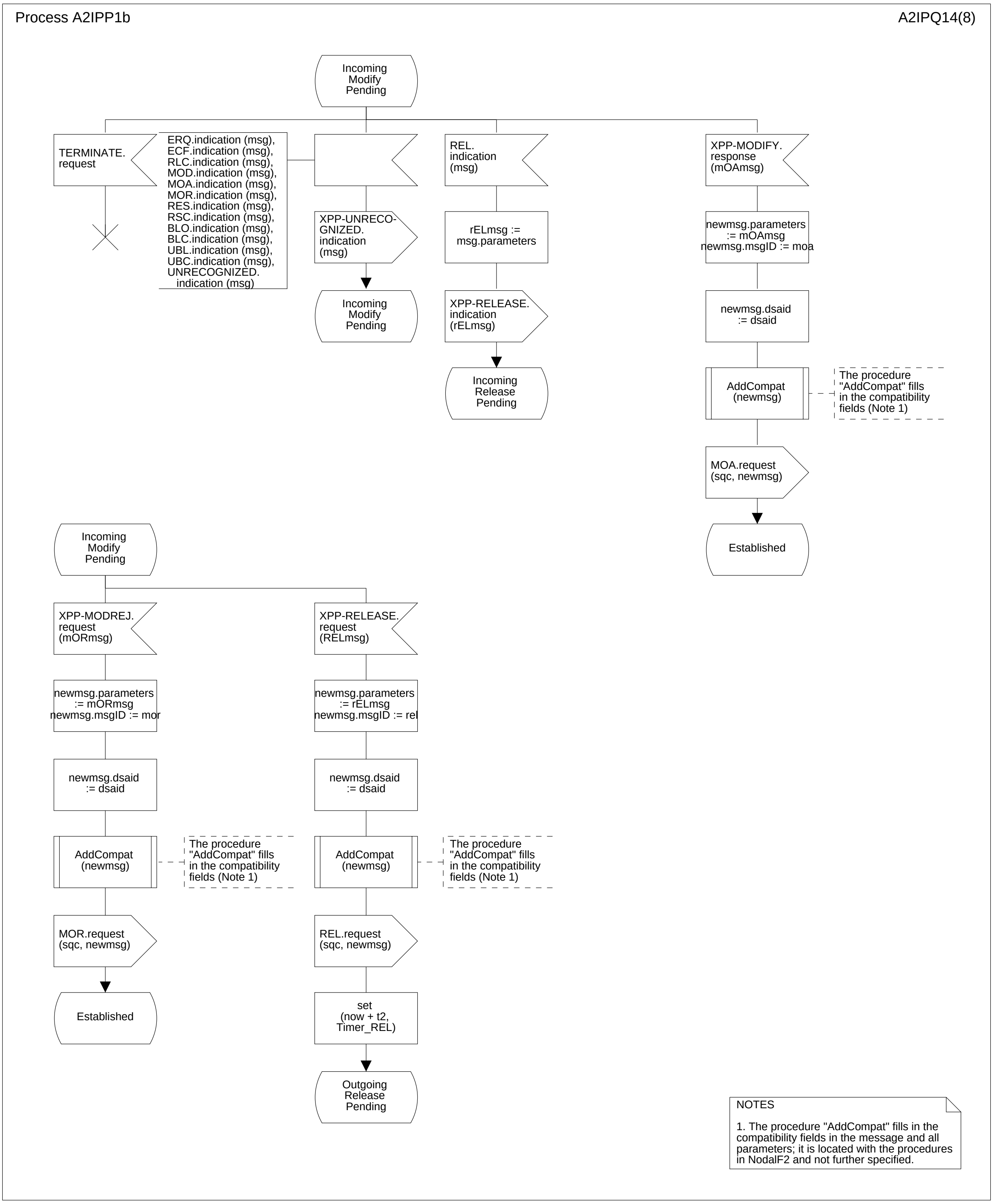

Process A2IPP1b
A2IPQ14(8)
Incoming Modify Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECOGNIZED. indication (msg)
Incoming Modify Pending
REL. indication (msg)
rELmsg := msg.parameters
XPP-RELEASE. indication (rELmsg)
Incoming Release Pending
XPP-MODIFY. response (mOAmsg)
newmsg.parameters := mOAmsg newmsg.msgID := moa
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
MOA.request (sqc, newmsg)
Established
Incoming Modify Pending
XPP-MODREJ. request (mORmsg)
newmsg.parameters := mORmsg newmsg.msgID := mor
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
MOR.request (sqc, newmsg)
Established
XPP-RELEASE. request (RELmsg)
newmsg.parameters := rELmsg newmsg.msgID := rel
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
REL.request (sqc, newmsg)
set (now + t2, Timer_REL)
Outgoing Release Pending
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 5 of 8)***

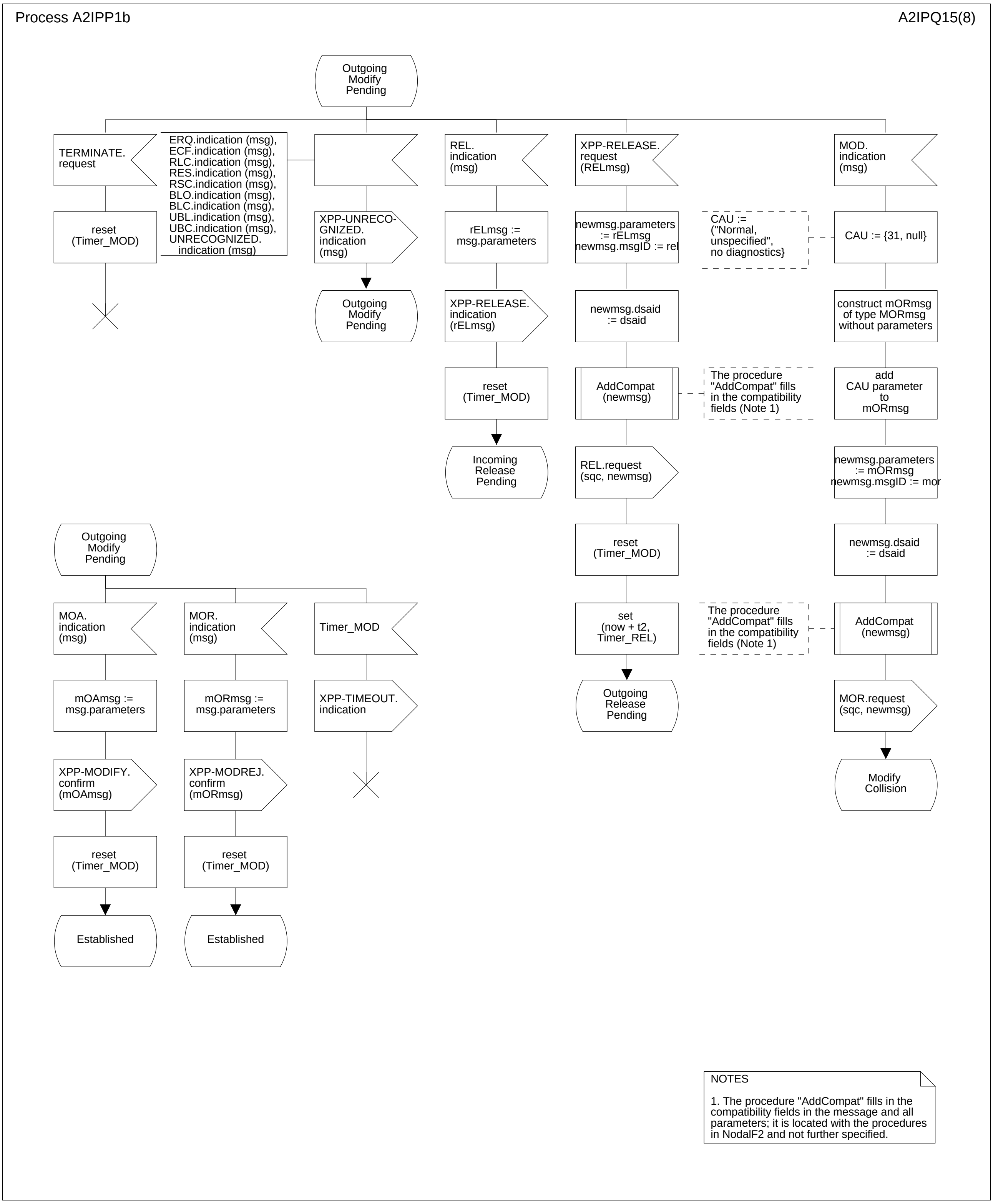

Process A2IPP1b
A2IPQ15(8)
Outgoing Modify Pending
TERMINATE. request
reset (Timer_MOD)
ERQ.indication (msg), ECF.indication (msg), RLC.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), UNRECOGNIZED. indication (msg)
XPP-UNRECOGNIZED. indication (msg)
Outgoing Modify Pending
REL. indication (msg)
rELmsg := msg.parameters
XPP-RELEASE. indication (rELmsg)
reset (Timer_MOD)
Incoming Release Pending
XPP-RELEASE. request (RELmsg)
newmsg.parameters := rELmsg newmsg.msgID := rel
CAU := ("Normal, unspecified", no diagnostics}
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
REL.request (sqc, newmsg)
reset (Timer_MOD)
set (now + t2, Timer_REL)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
Outgoing Release Pending
MOD. indication (msg)
CAU := {31, null}
construct mORmsg of type MORmsg without parameters
add CAU parameter to mORmsg
newmsg.parameters := mORmsg newmsg.msgID := mor
newmsg.dsaid := dsaid
AddCompat (newmsg)
MOR.request (sqc, newmsg)
Modify Collision
Outgoing Modify Pending
MOA. indication (msg)
mOAmsg := msg.parameters
XPP-MODIFY. confirm (mOAmsg)
reset (Timer_MOD)
Established
MOR. indication (msg)
mORmsg := msg.parameters
XPP-MODREJ. confirm (mORmsg)
reset (Timer_MOD)
Established
Timer_MOD
XPP-TIMEOUT. indication
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 6 of 8)***

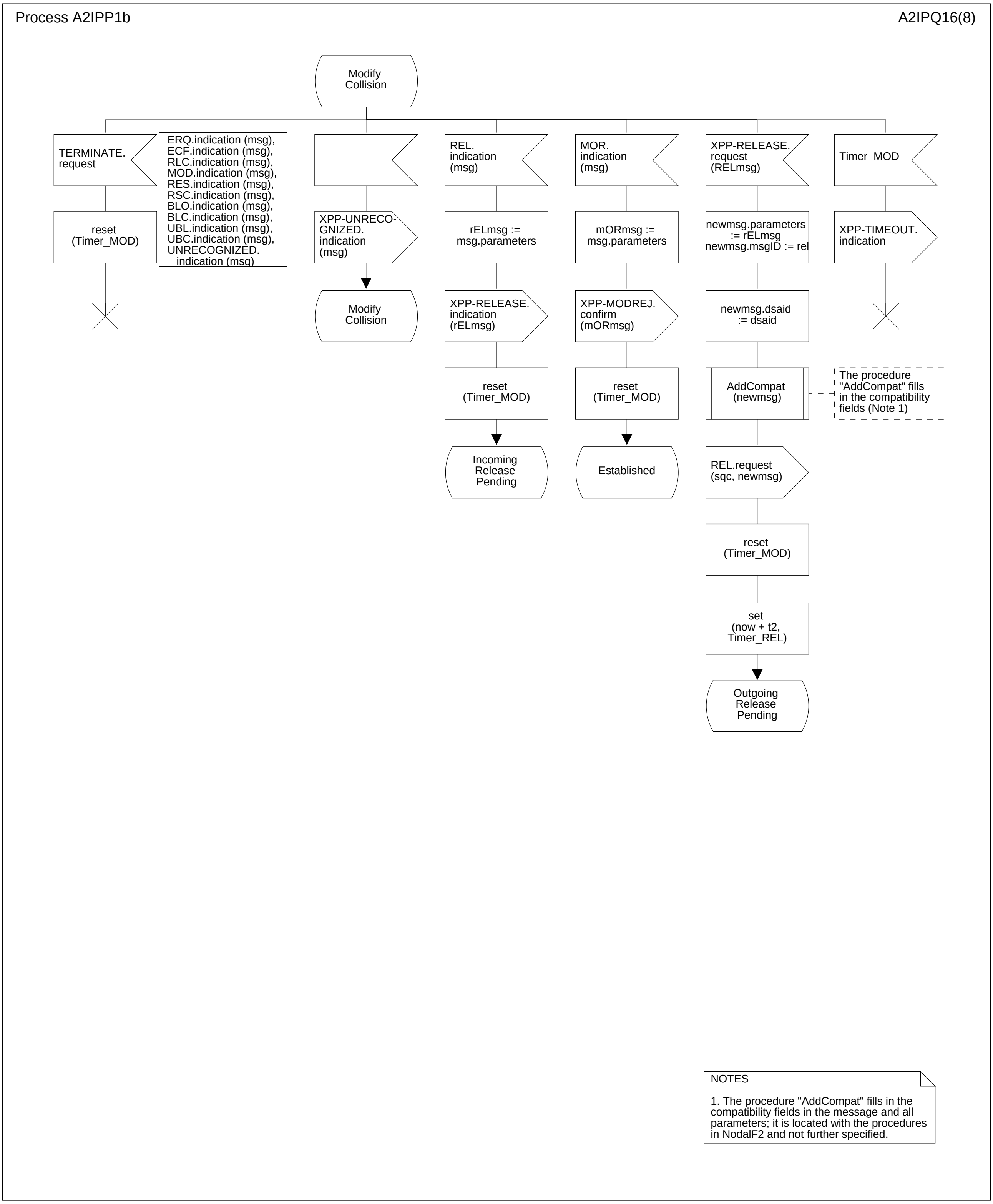


**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 7 of 8)***

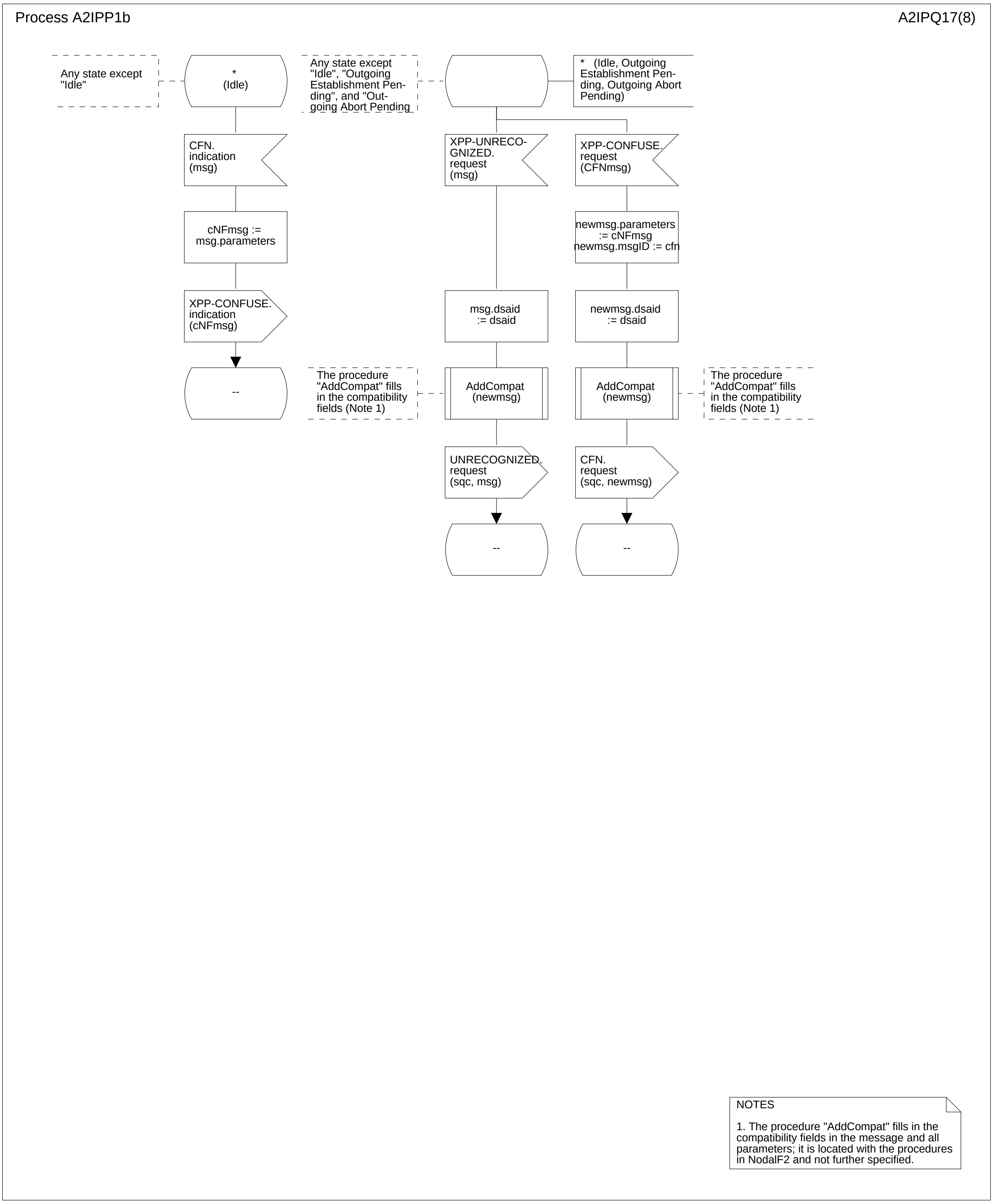

Process A2IPP1b
A2IPQ17(8)
Any state except "Idle"
* (Idle)
Any state except "Idle", "Outgoing Establishment Pending", and "Outgoing Abort Pending
* (Idle, Outgoing Establishment Pending, Outgoing Abort Pending)
CFN. indication (msg)
cNFmsg := msg.parameters
XPP-CONFUSE. indication (cNFmsg)
--
XPP-UNRECOGNIZED. request (msg)
msg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
UNRECOGNIZED. request (sqc, msg)
--
XPP-CONFUSE. request (CFNmsg)
newmsg.parameters := cNFmsg newmsg.msgID := cfn
newmsg.dsaid := dsaid
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
CFN. request (sqc, newmsg)
--
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.5/Q.2630.2 – SDL diagram of the incoming protocol procedure *(part 8 of 8)***

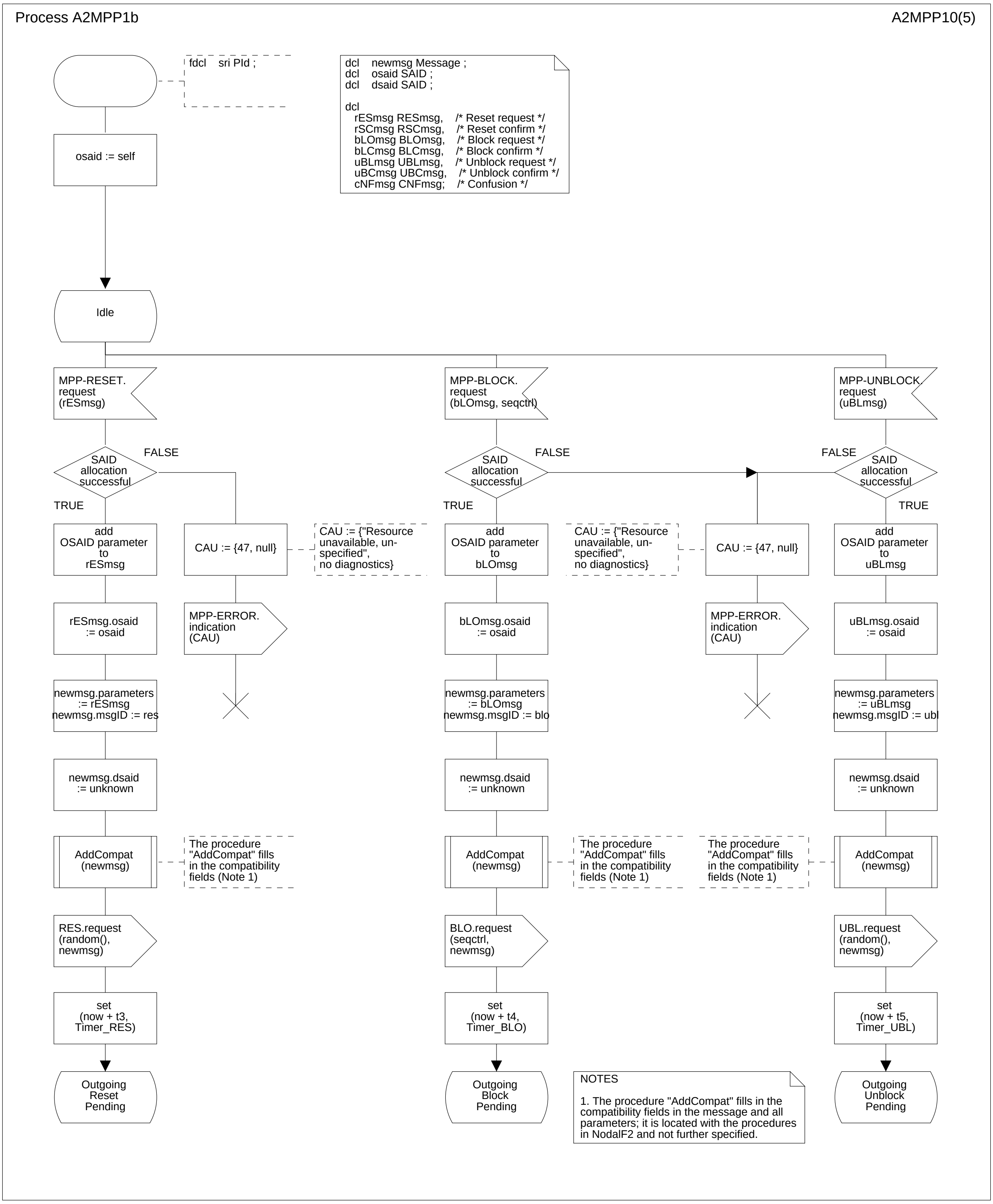

Process A2MPP1b
A2MPP10(5)
fdcl sri PId ;
dcl newmsg Message ;
dcl osaid SAID ;
dcl dsaid SAID ;
dcl
rESmsg RESmsg, /* Reset request */
rSCmsg RSCmsg, /* Reset confirm */
bLOmsg BLOmsg, /* Block request */
bLCmsg BLCmsg, /* Block confirm */
uBLmsg UBLmsg, /* Unblock request */
uBCmsg UBCmsg, /* Unblock confirm */
cNFmsg CNFmsg; /* Confusion */
osaid := self
Idle
MPP-RESET. request (rESmsg)
MPP-BLOCK. request (bLOmsg, seqctrl)
MPP-UNBLOCK. request (uBLmsg)
SAID allocation successful
FALSE
TRUE
SAID allocation successful
FALSE
TRUE
FALSE
SAID allocation successful
TRUE
add OSAID parameter to rESmsg
CAU := {47, null}
CAU := {"Resource unavailable, un-specified", no diagnostics}
add OSAID parameter to bLOmsg
CAU := {"Resource unavailable, un-specified", no diagnostics}
CAU := {47, null}
add OSAID parameter to uBLmsg
rESmsg.osaid := osaid
MPP-ERROR. indication (CAU)
bLOmsg.osaid := osaid
MPP-ERROR. indication (CAU)
uBLmsg.osaid := osaid
newmsg.parameters := rESmsg newmsg.msgID := res
newmsg.parameters := bLOmsg newmsg.msgID := blo
newmsg.parameters := uBLmsg newmsg.msgID := ubl
newmsg.dsaid := unknown
newmsg.dsaid := unknown
newmsg.dsaid := unknown
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
RES.request (random(), newmsg)
BLO.request (seqctrl, newmsg)
UBL.request (random(), newmsg)
set (now + t3, Timer_RES)
set (now + t4, Timer_BLO)
set (now + t5, Timer_UBL)
Outgoing Reset Pending
Outgoing Block Pending
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.
Outgoing Unblock Pending

**Figure D.6/Q.2630.2 – SDL diagram of the maintenance protocol procedure *(part 1 of 5)***

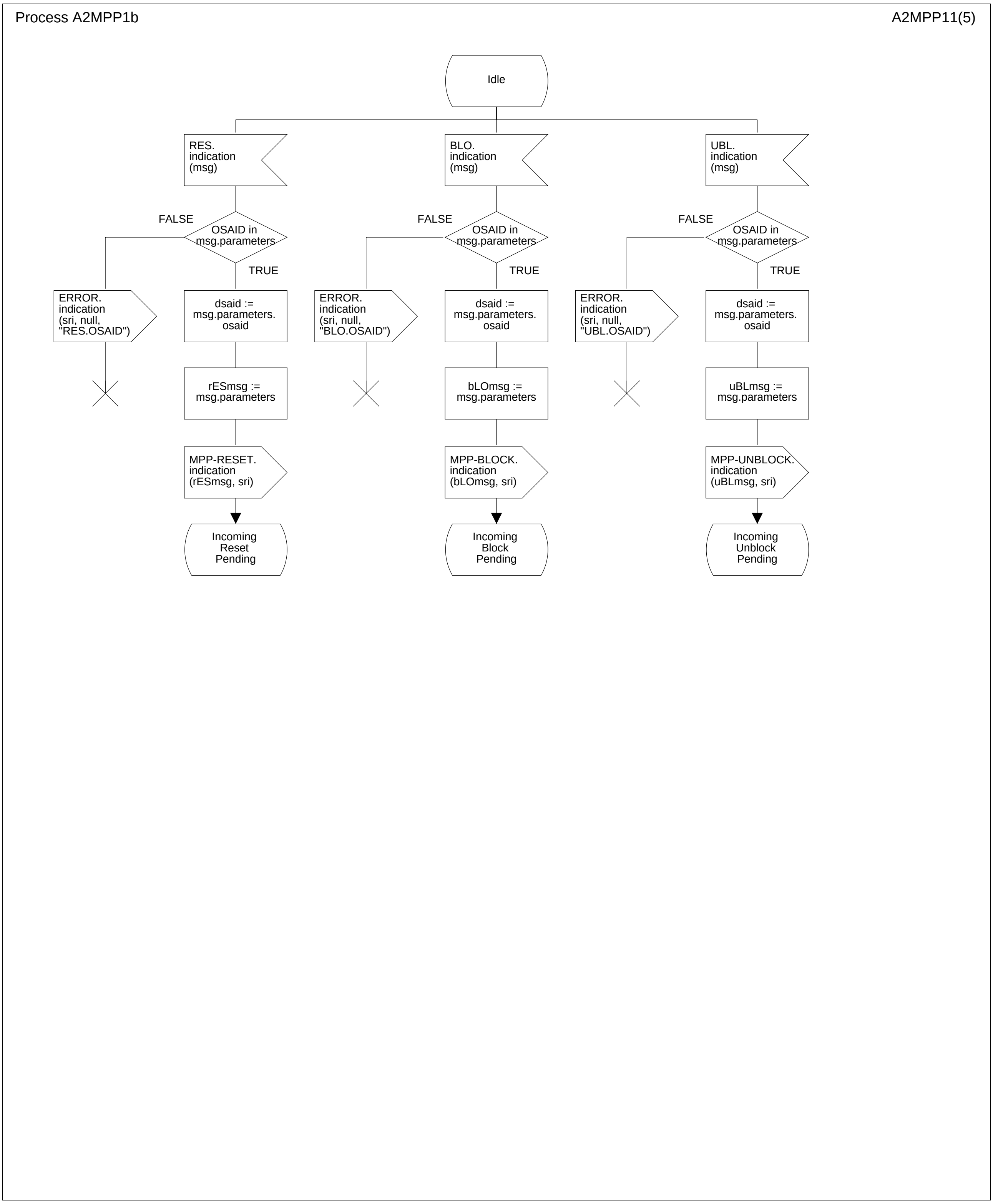


**Figure D.6/Q.2630.2 – SDL diagram of the maintenance protocol procedure *(part 2 of 5)***

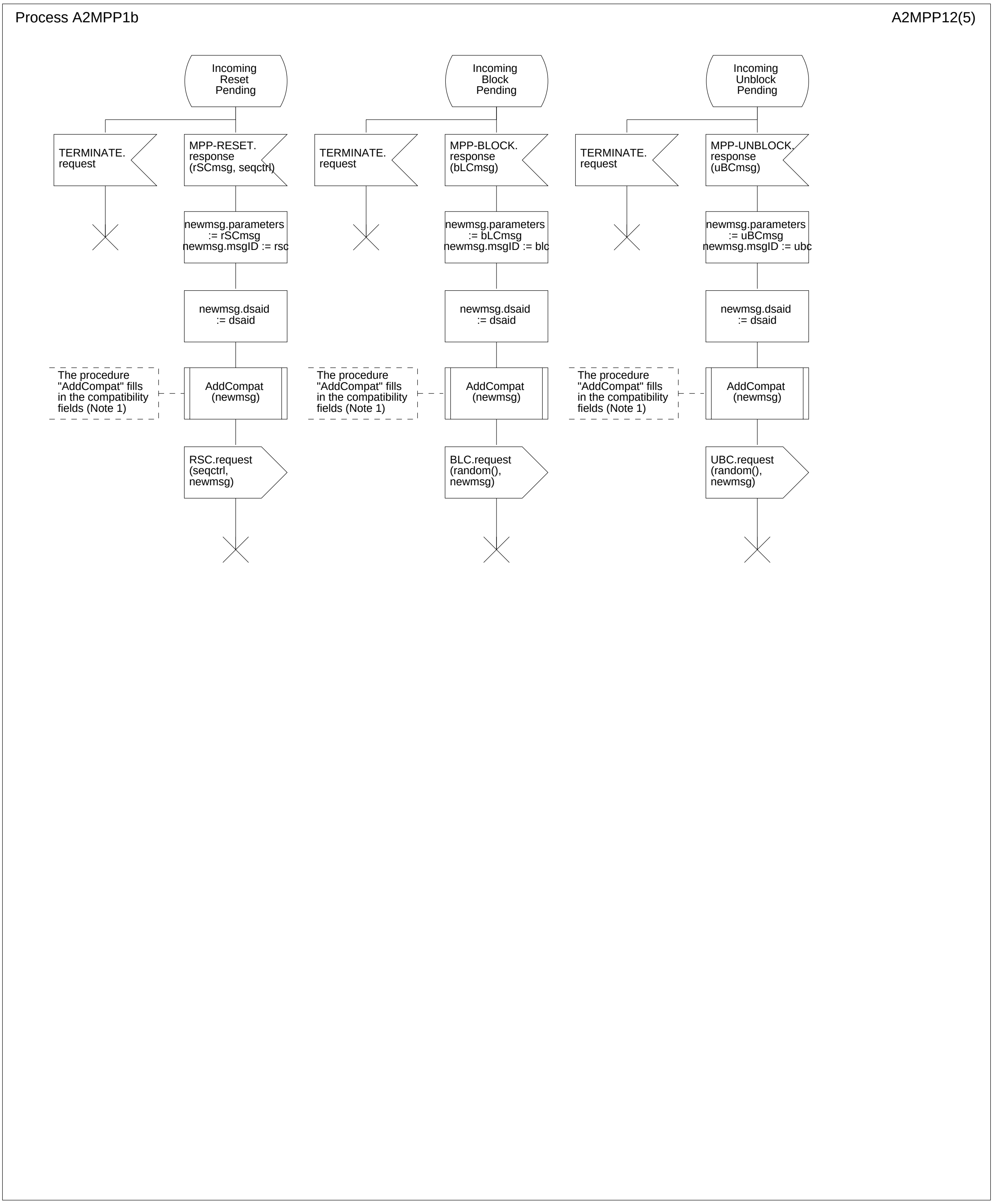

Process A2MPP1b
A2MPP12(5)
Incoming Reset Pending
Incoming Block Pending
Incoming Unblock Pending
TERMINATE. request
MPP-RESET. response (rSCmsg, seqctrl)
TERMINATE. request
MPP-BLOCK. response (bLCmsg)
TERMINATE. request
MPP-UNBLOCK. response (uBCmsg)
newmsg.parameters := rSCmsg newmsg.msgID := rsc
newmsg.parameters := bLCmsg newmsg.msgID := blc
newmsg.parameters := uBCmsg newmsg.msgID := ubc
newmsg.dsaid := dsaid
newmsg.dsaid := dsaid
newmsg.dsaid := dsaid
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
The procedure "AddCompat" fills in the compatibility fields (Note 1)
AddCompat (newmsg)
RSC.request (seqctrl, newmsg)
BLC.request (random(), newmsg)
UBC.request (random(), newmsg)

**Figure D.6/Q.2630.2 – SDL diagram of the maintenance protocol procedure *(part 3 of 5)***

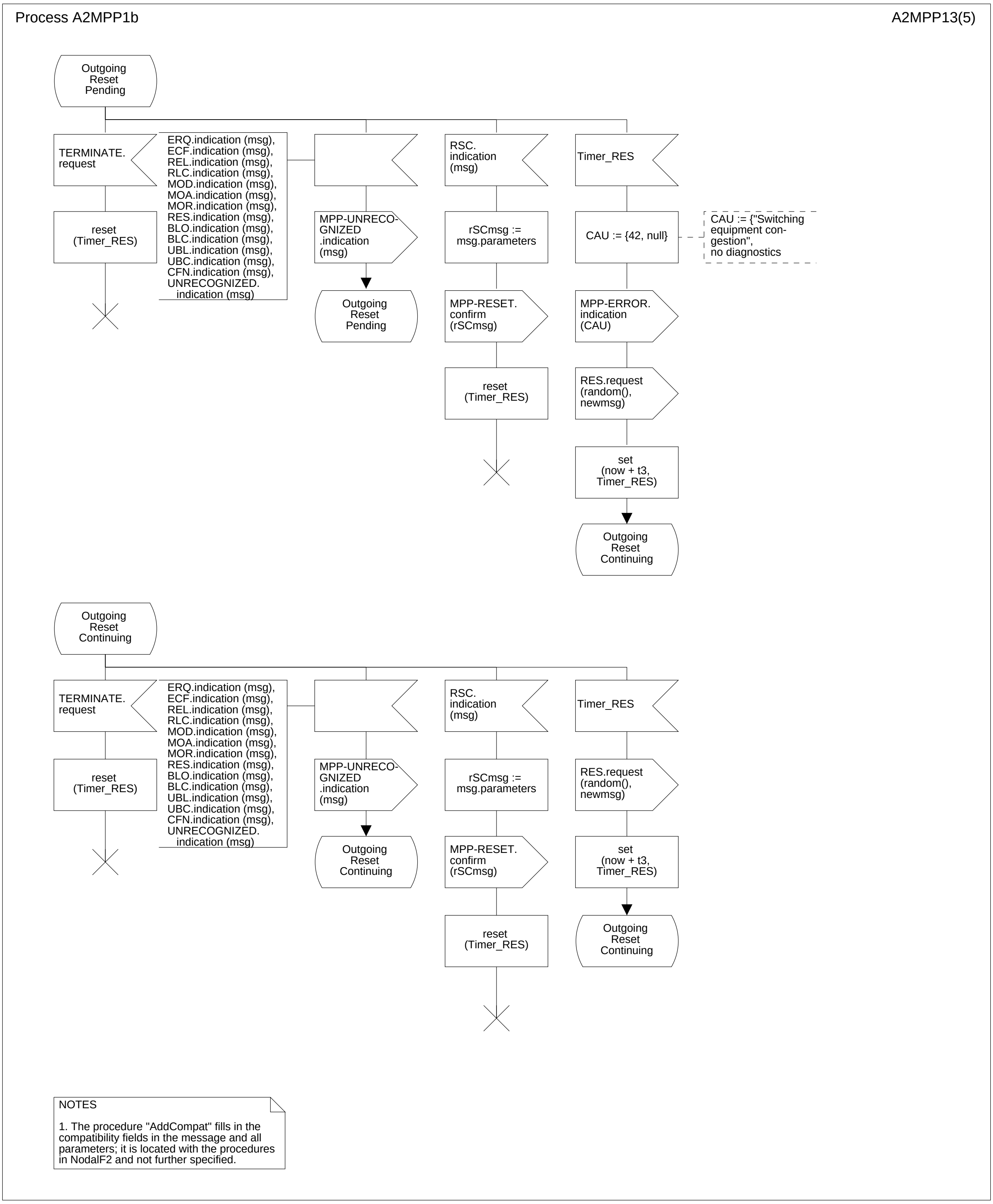

Process A2MPP1b
A2MPP13(5)
Outgoing Reset Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), CFN.indication (msg), UNRECOGNIZED. indication (msg)
RSC. indication (msg)
Timer_RES
reset (Timer_RES)
MPP-UNRECOGNIZED .indication (msg)
rSCmsg := msg.parameters
CAU := {42, null}
CAU := {"Switching equipment congestion", no diagnostics
Outgoing Reset Pending
MPP-RESET. confirm (rSCmsg)
MPP-ERROR. indication (CAU)
reset (Timer_RES)
RES.request (random(), newmsg)
set (now + t3, Timer_RES)
Outgoing Reset Continuing
Outgoing Reset Continuing
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), BLO.indication (msg), BLC.indication (msg), UBL.indication (msg), UBC.indication (msg), CFN.indication (msg), UNRECOGNIZED. indication (msg)
RSC. indication (msg)
Timer_RES
reset (Timer_RES)
MPP-UNRECOGNIZED .indication (msg)
rSCmsg := msg.parameters
RES.request (random(), newmsg)
Outgoing Reset Continuing
MPP-RESET. confirm (rSCmsg)
set (now + t3, Timer_RES)
reset (Timer_RES)
Outgoing Reset Continuing
NOTES
1. The procedure "AddCompat" fills in the compatibility fields in the message and all parameters; it is located with the procedures in NodalF2 and not further specified.

**Figure D.6/Q.2630.2 – SDL diagram of the maintenance protocol procedure *(part 4 of 5)***

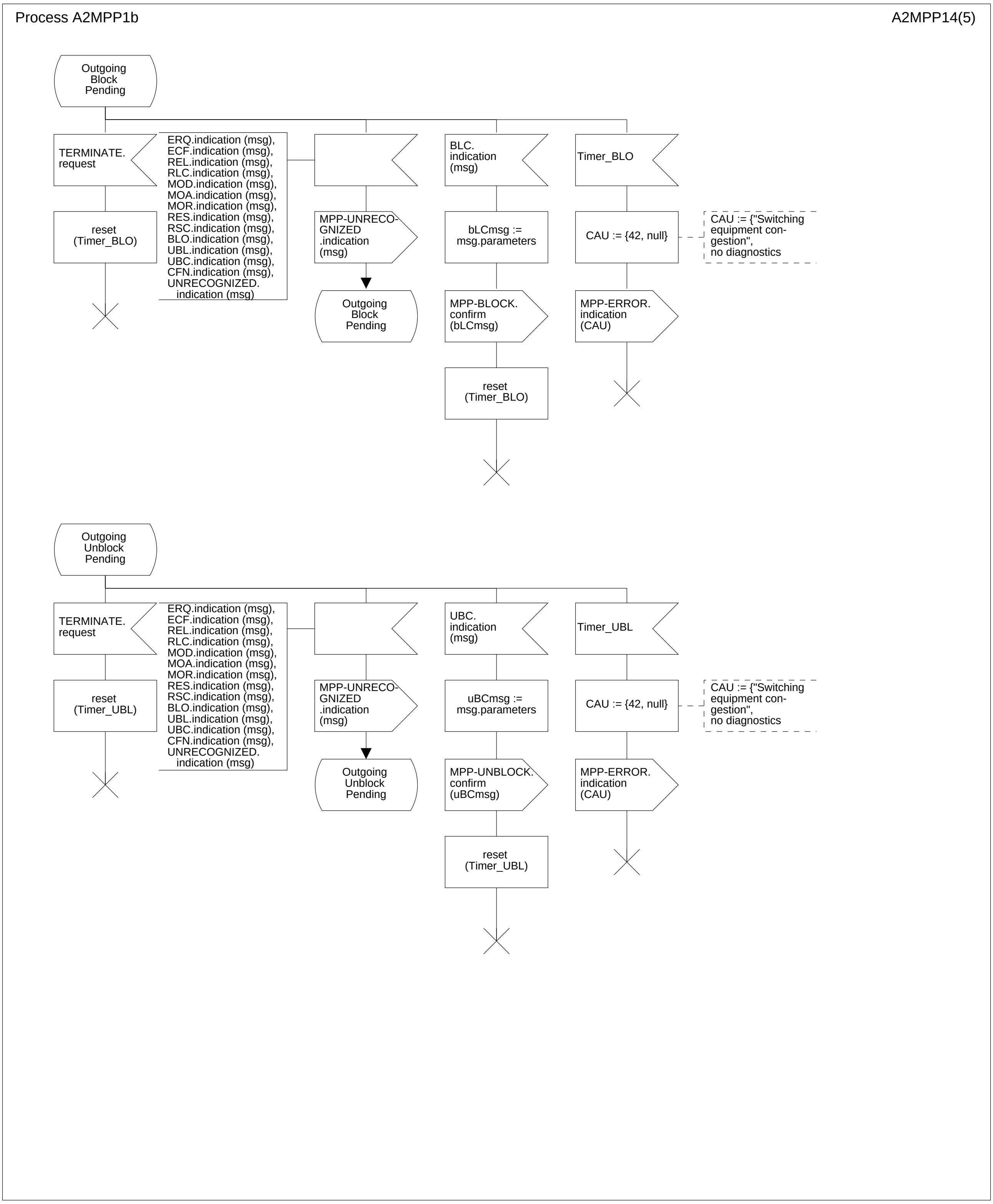

Process A2MPP1b
A2MPP14(5)
Outgoing Block Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), UBL.indication (msg), UBC.indication (msg), CFN.indication (msg), UNRECOGNIZED. indication (msg)
reset (Timer_BLO)
MPP-UNRECOGNIZED .indication (msg)
Outgoing Block Pending
BLC. indication (msg)
bLCmsg := msg.parameters
MPP-BLOCK. confirm (bLCmsg)
reset (Timer_BLO)
Timer_BLO
CAU := {42, null}
CAU := {"Switching equipment congestion", no diagnostics
MPP-ERROR. indication (CAU)
Outgoing Unblock Pending
TERMINATE. request
ERQ.indication (msg), ECF.indication (msg), REL.indication (msg), RLC.indication (msg), MOD.indication (msg), MOA.indication (msg), MOR.indication (msg), RES.indication (msg), RSC.indication (msg), BLO.indication (msg), UBL.indication (msg), UBC.indication (msg), CFN.indication (msg), UNRECOGNIZED. indication (msg)
reset (Timer_UBL)
MPP-UNRECOGNIZED .indication (msg)
Outgoing Unblock Pending
UBC. indication (msg)
uBCmsg := msg.parameters
MPP-UNBLOCK. confirm (uBCmsg)
reset (Timer_UBL)
Timer_UBL
CAU := {42, null}
CAU := {"Switching equipment congestion", no diagnostics
MPP-ERROR. indication (CAU)

**Figure D.6/Q.2630.2 – SDL diagram of the maintenance protocol procedure *(part 5 of 5)***

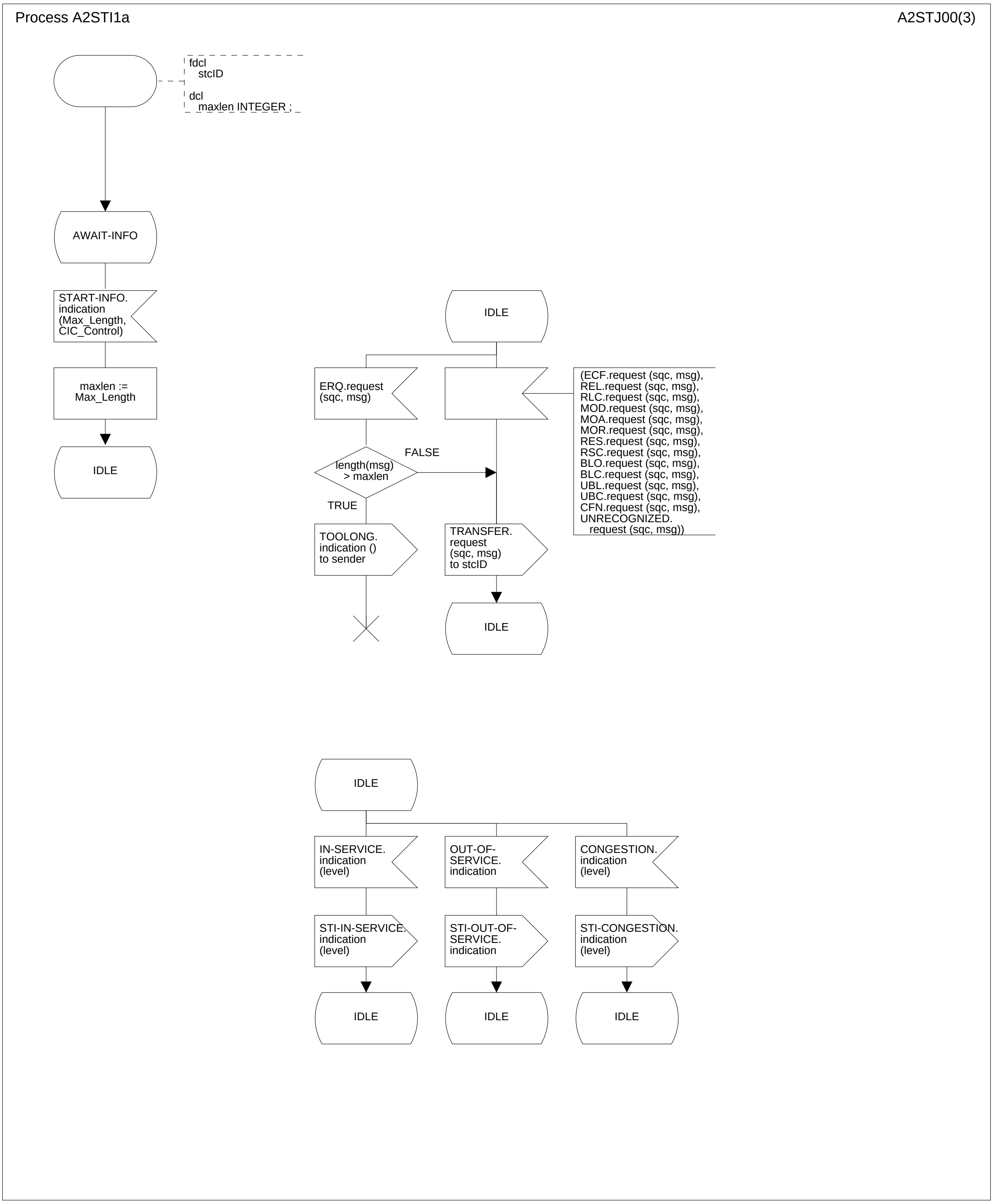


**Figure D.7/Q.2630.2 – SDL diagram of the signalling transport interface *(part 1 of 3)***

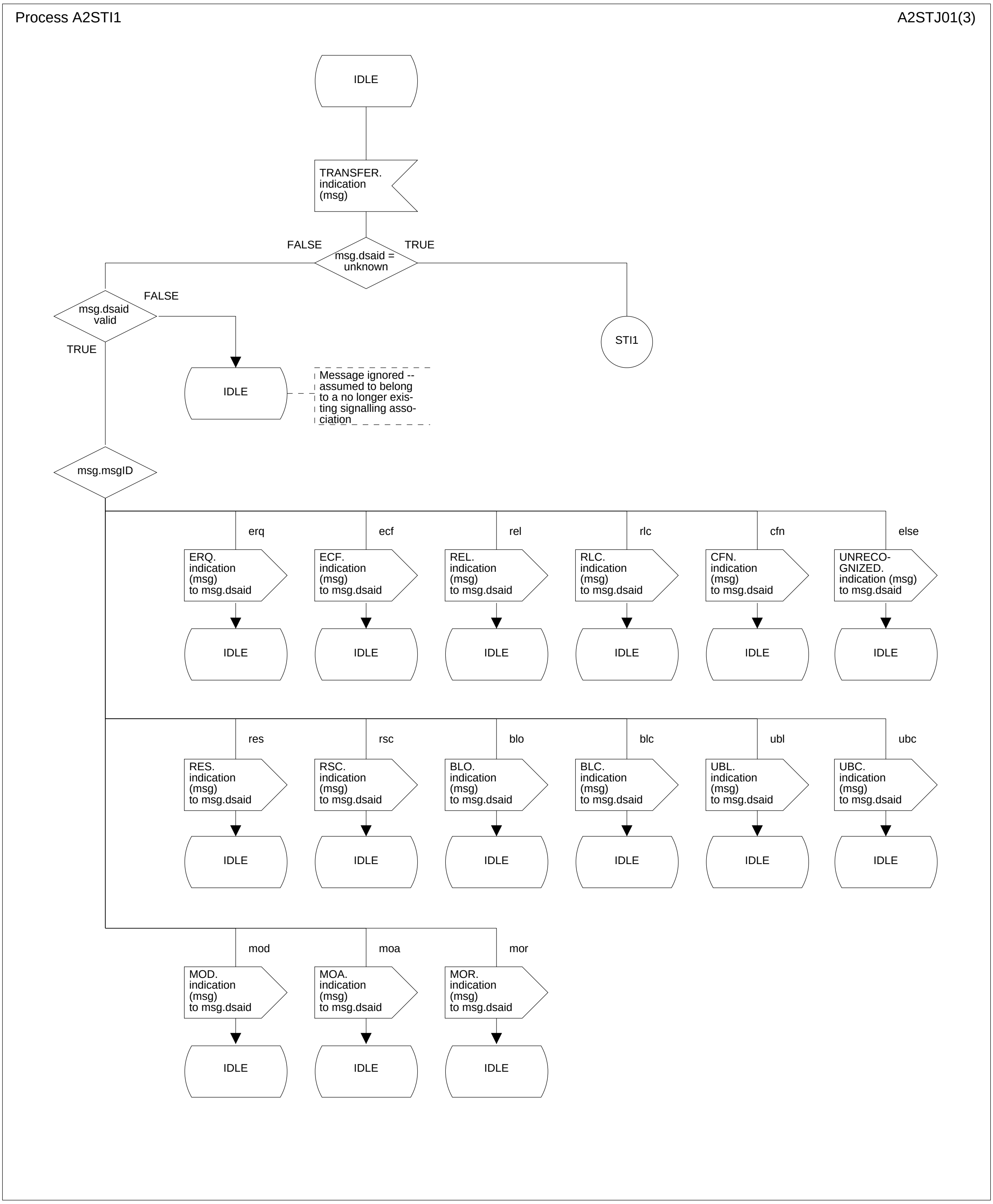
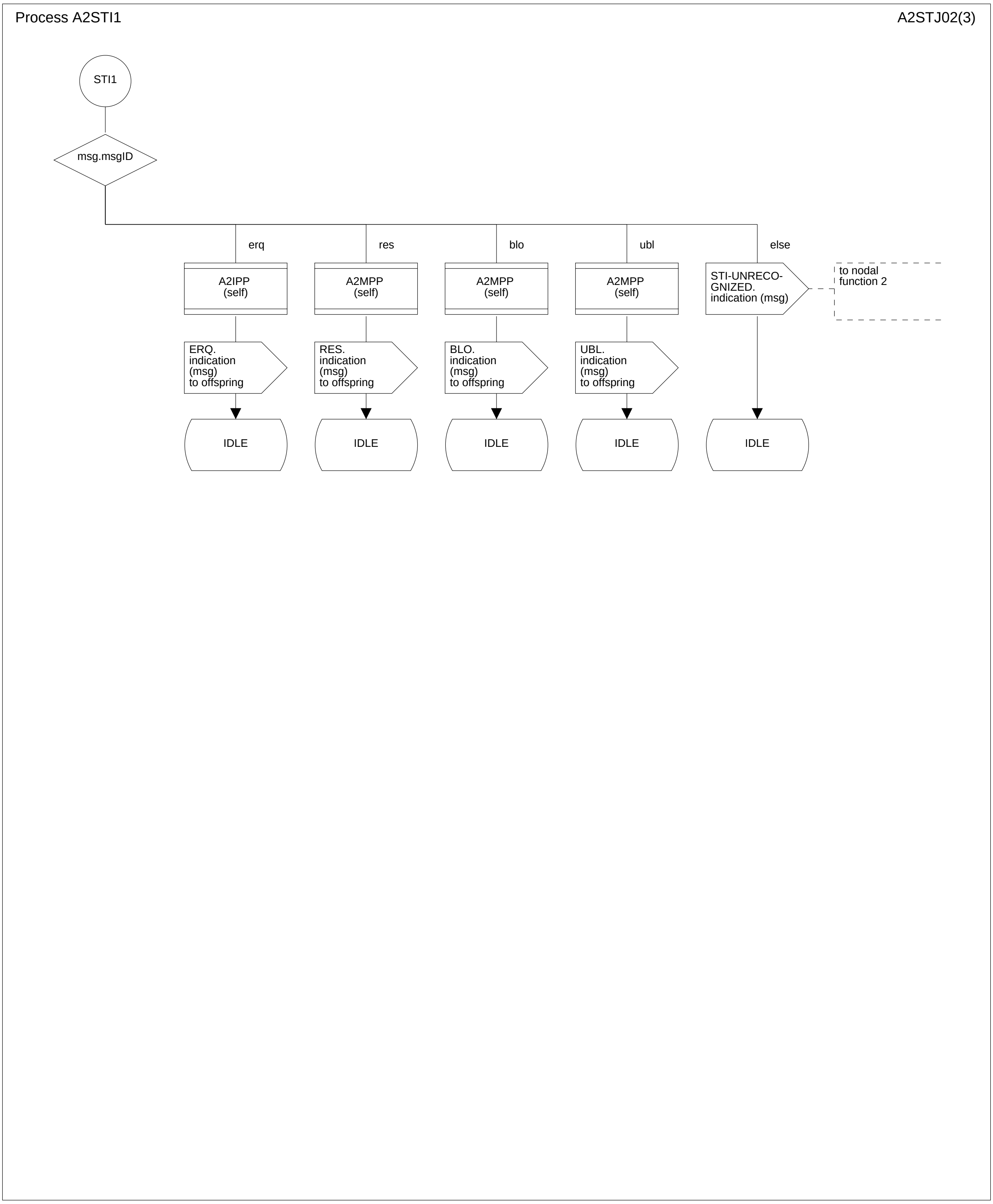

Process A2STI1
A2STJ01(3)
IDLE
TRANSFER. indication (msg)
FALSE
TRUE
msg.dsaid = unknown
msg.dsaid valid
FALSE
TRUE
STI1
IDLE
Message ignored -- assumed to belong to a no longer existing signalling association
msg.msgID
erq
ecf
rel
rlc
cfn
else
ERQ. indication (msg) to msg.dsaid
ECF. indication (msg) to msg.dsaid
REL. indication (msg) to msg.dsaid
RLC. indication (msg) to msg.dsaid
CFN. indication (msg) to msg.dsaid
UNRECOGNIZED. indication (msg) to msg.dsaid
IDLE
IDLE
IDLE
IDLE
IDLE
IDLE
res
rsc
blo
blc
ubl
ubc
RES. indication (msg) to msg.dsaid
RSC. indication (msg) to msg.dsaid
BLO. indication (msg) to msg.dsaid
BLC. indication (msg) to msg.dsaid
UBL. indication (msg) to msg.dsaid
UBC. indication (msg) to msg.dsaid
IDLE
IDLE
IDLE
IDLE
IDLE
IDLE
mod
moa
mor
MOD. indication (msg) to msg.dsaid
MOA. indication (msg) to msg.dsaid
MOR. indication (msg) to msg.dsaid
IDLE
IDLE
IDLE

**Figure D.7/Q.2630.2 – SDL diagram of the signalling transport interface *(part 2 of 3)***



Process A2STI1
A2STJ02(3)
STI1
msg.msgID
erq
res
blo
ubl
else
A2IPP (self)
A2MPP (self)
A2MPP (self)
A2MPP (self)
STI-UNRECO-GNIZED. indication (msg)
to nodal function 2
ERQ. indication (msg) to offspring
RES. indication (msg) to offspring
BLO. indication (msg) to offspring
UBL. indication (msg) to offspring
IDLE
IDLE
IDLE
IDLE
IDLE

**Figure D.7/Q.2630.2 – SDL diagram of the signalling transport interface *(part 3 of 3)***

## SERIES OF ITU-T RECOMMENDATIONS

| | |
|---|---|
| Series A | Organization of the work of ITU-T |
| Series B | Means of expression: definitions, symbols, classification |
| Series C | General telecommunication statistics |
| Series D | General tariff principles |
| Series E | Overall network operation, telephone service, service operation and human factors |
| Series F | Non-telephone telecommunication services |
| Series G | Transmission systems and media, digital systems and networks |
| Series H | Audiovisual and multimedia systems |
| Series I | Integrated services digital network |
| Series J | Cable networks and transmission of television, sound programme and other multimedia signals |
| Series K | Protection against interference |
| Series L | Construction, installation and protection of cables and other elements of outside plant |
| Series M | TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits |
| Series N | Maintenance: international sound programme and television transmission circuits |
| Series O | Specifications of measuring equipment |
| Series P | Telephone transmission quality, telephone installations, local line networks |
| **Series Q** | **Switching and signalling** |
| Series R | Telegraph transmission |
| Series S | Telegraph services terminal equipment |
| Series T | Terminals for telematic services |
| Series U | Telegraph switching |
| Series V | Data communication over the telephone network |
| Series X | Data networks and open system communications |
| Series Y | Global information infrastructure and Internet protocol aspects |
| Series Z | Languages and general software aspects for telecommunication systems |

Printed in Switzerland
Geneva, 2003