INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.2630.1

Annex B

(03/2001)

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 1

Annex B: SDL definition of the AAL type 2 signalling protocol CS-1

ITU-T Recommendation Q.2630.1 – Annex B

(Formerly CCITT Recommendation)

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 SYSTEMS No. 4 AND No. 5	Q.120–Q.249
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.2630.1

AAL type 2 signalling protocol – Capability set 1

ANNEX B

SDL definition of the AAL type 2 signalling protocol CS-1

Summary

This annex contains the SDL definition of AAL Type 2 Signalling Protocol CS-1 for ITU-T Recommendation Q.2630.1. SDL diagrams are in electronic form only.

Source

Annex B to ITU-T Recommendation Q.2630.1 was prepared by ITU-T Study Group 11 (2001-2004) and approved under the WTSA Resolution 1 procedure on 1 March 2001.

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.2630.1

AAL type 2 signalling protocol – Capability set 1

ANNEX B

SDL definition of the AAL type 2 signalling protocol CS-1

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

B.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.

B.2 The SDL system diagram

The SDL system diagram is depicted in Figure B.1.

B.3 The SDL block structure diagram

The SDL block structure diagrams are depicted in Figure B.2 (parts 1 to 4 of 4).

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.

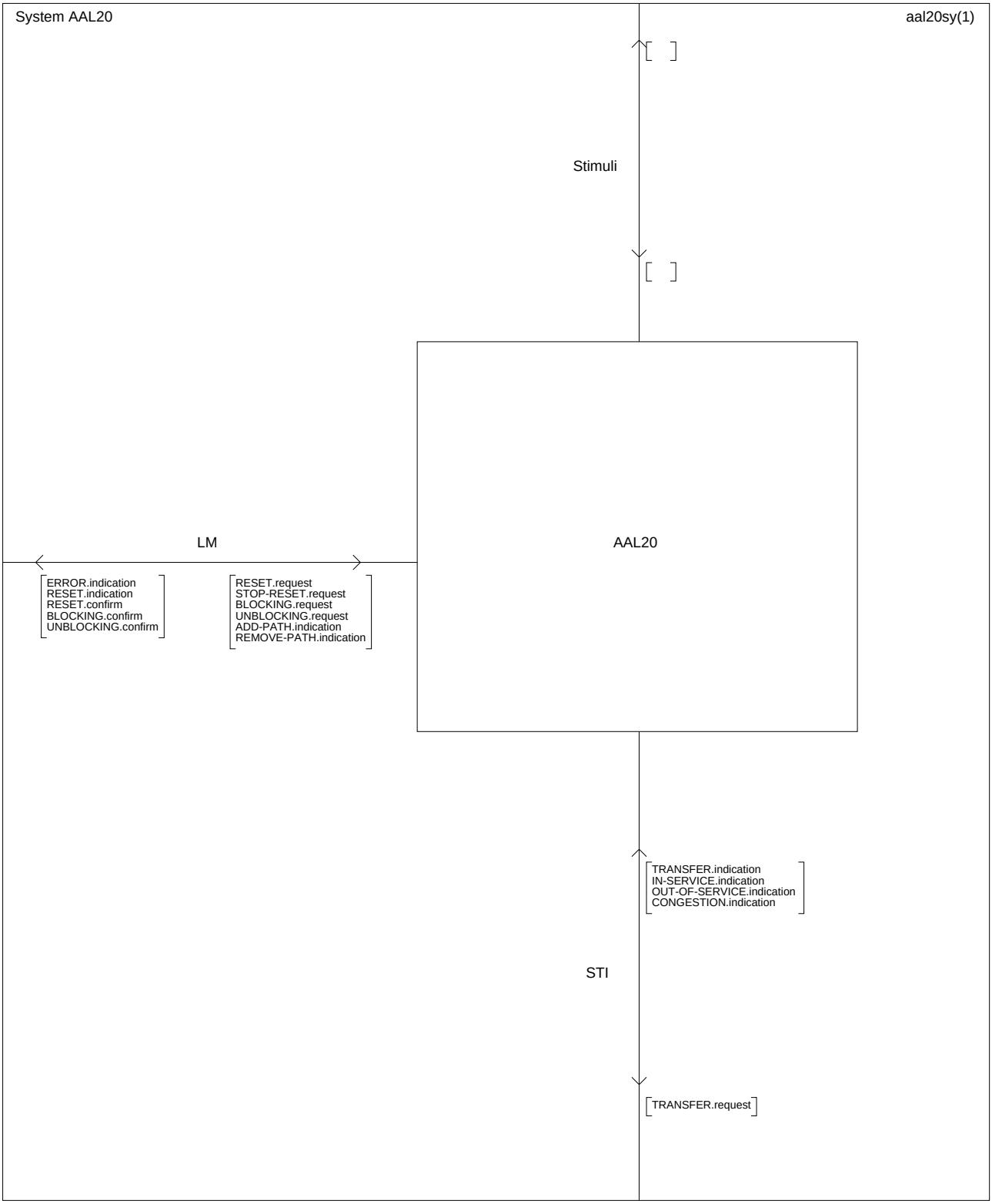


Figure B.1/Q.2630.1 – SDL system of the AAL type 2 signalling entity

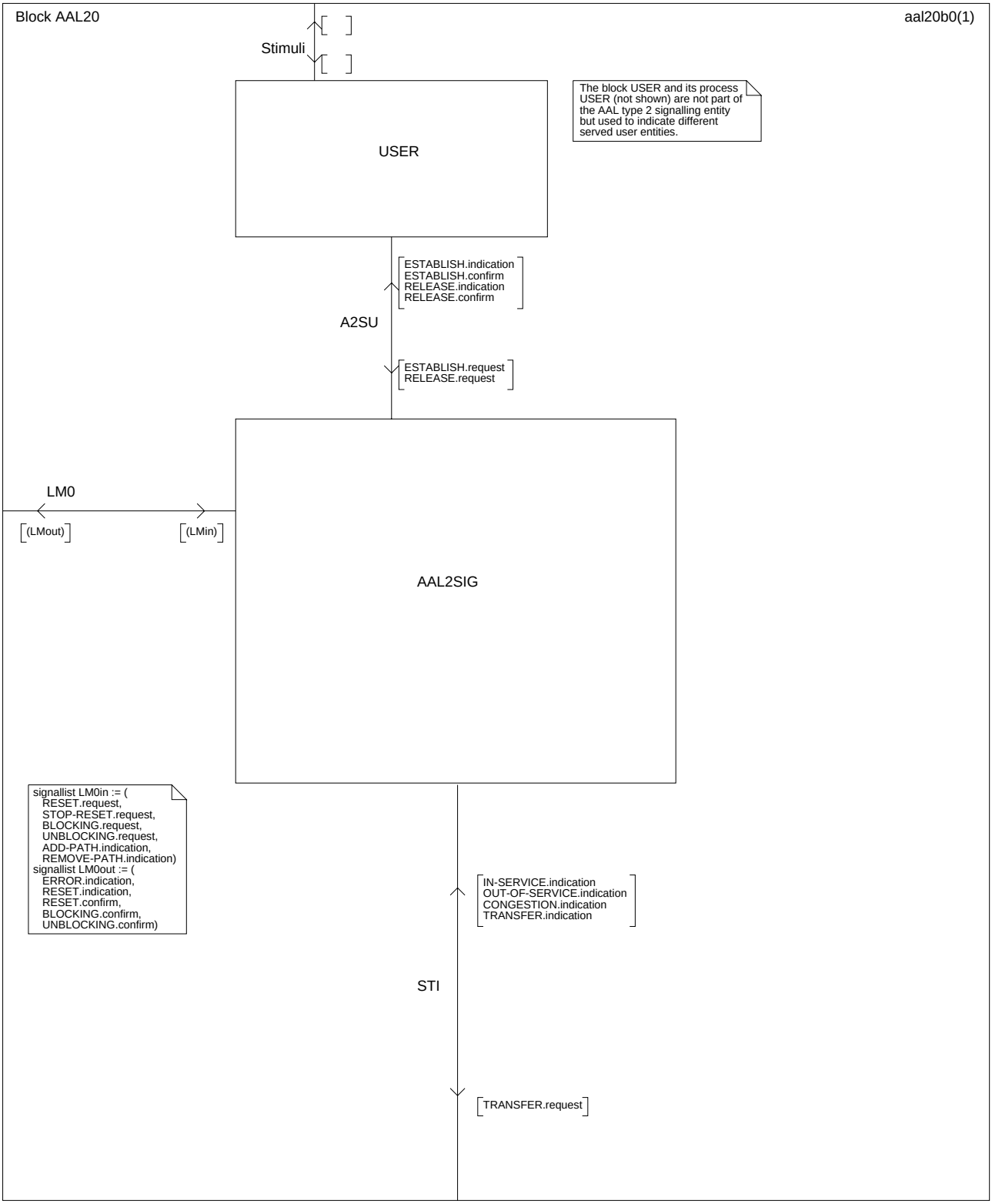


Figure B.2/Q.2630.1 – SDL block structure of the AAL type 2 signalling entity *(part 1 of 4)*

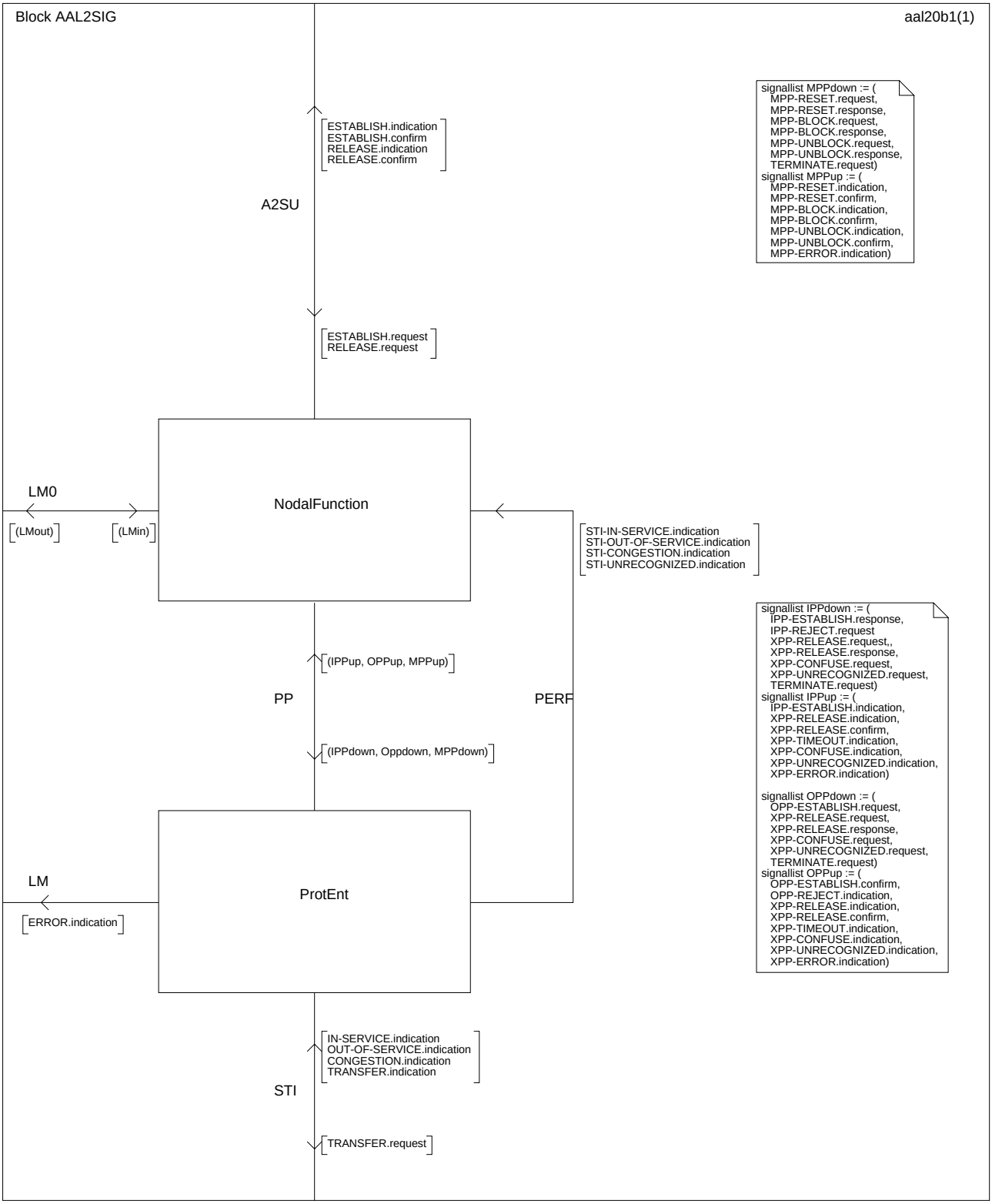


Figure B.2/Q.2630.1 – SDL block structure of the AAL type 2 signalling entity *(part 2 of 4)*

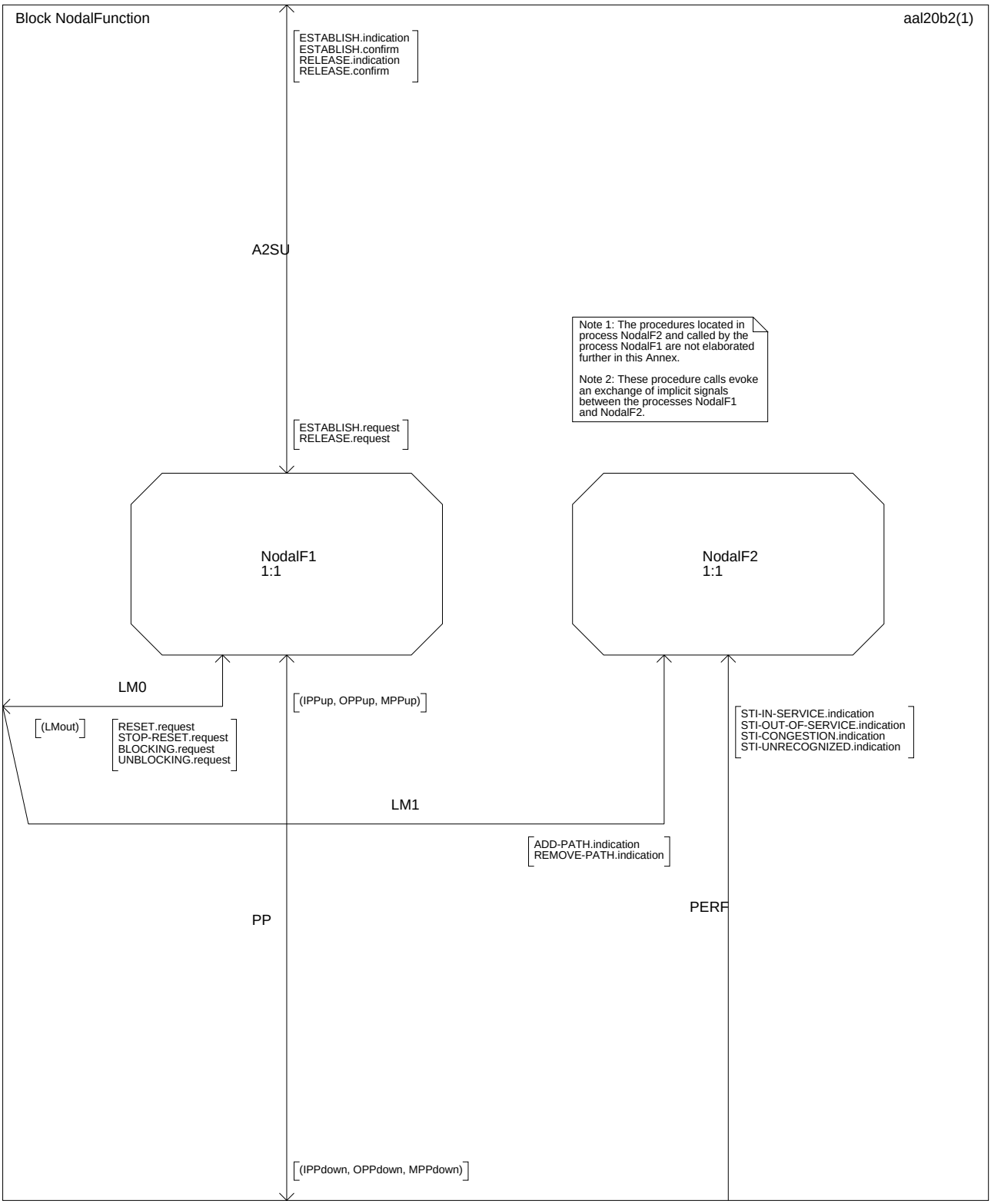


Figure B.2/Q.2630.1 – SDL block structure of the AAL type 2 signalling entity *(part 3 of 4)*

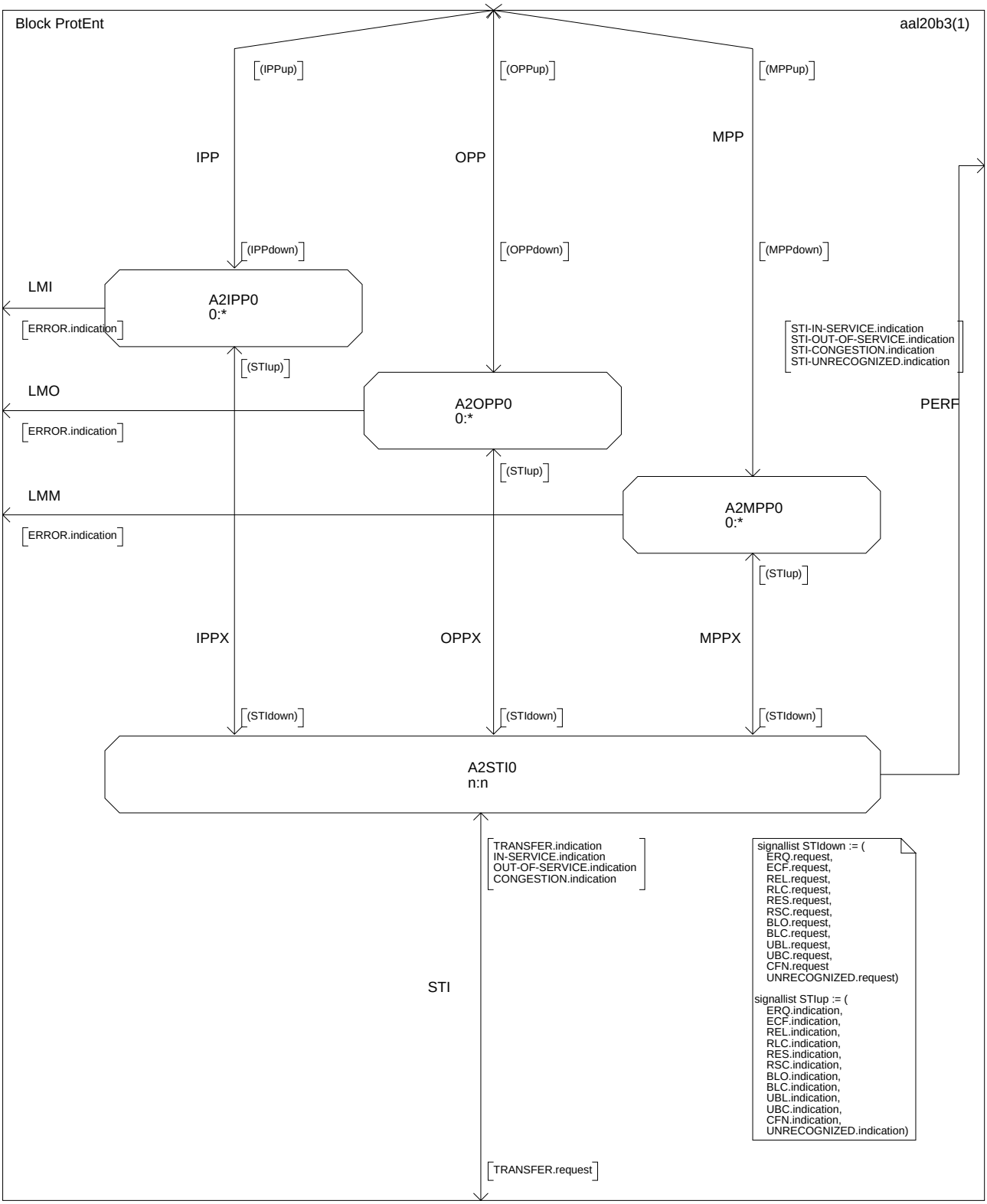


Figure B.2/Q.2630.1 – SDL block structure of the AAL type 2 signalling entity *(part 4 of 4)*

B.4 SDL specification for the nodal function

B.4.1 Introduction

In Figure B.2 (part 3 of 4) it is shown that the nodal function is separated into:

a) *Nodal Function 1*

This entity contains all functionality that is defined in detail in 8.2 and, therefore, can be specified precisely in SDL diagrams.

b) *Nodal Function 2*

This entity contains all functionality that contains implementation specific parts, only results are defined in detail in 8.2 (but not their internal mechanisms). These parts are not defined in detail in this Annex either; only the expected result is documented.

Therefore, this clause defines the Nodal Function 1 (see B.4.2) and the procedures of the Nodal Function 2 (see B.4.3). The latter by their nature are not specified in SDL; nevertheless, the functionality expected from these procedures needs to be understood to complete the definition of the SDL diagrams for the AAL type 2 signalling entity.

The USER, i.e. the AAL type 2 served user, processes are not defined either, nor the interaction of these processes with the environment (e.g., for simulation stimuli). It is assumed that for every AAL type 2 connection a separate USER process exists (at both ends of the connection) and that its (SDL process) identity is used to route the signals.

NOTE – The processes of the protocol entities are described in 8.3.

B.4.2 SDL diagrams for the Nodal Function 1

B.4.2.1 Data structures

For the SDL model, the AAL type 2 signalling entity maintains a record of type "CRec" for every AAL type 2 connection from the instance in time where the connection is being established until it is released. The terms "preceeding" and "succeeding" in the following discussion relate to the direction of the connection establishment.

When such a record is looked up (with the protocol procedure entity instance's identifier), the record is organized such that the "incoming" part refers to the link from where a message has been received; this is reflected in the status value as described in Table B.1.

Table B.1/Q.2630.1 – 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

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

```
CRec ::= SEQUENCE {
     status         CRecStatus,         -- connection status
     incoming       HRec,               -- details incoming link
     outgoing       HRec }              -- details outgoing link

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
```

For the SDL model, the AAL type 2 signalling entity maintains a record of type "MRec" for every outgoing maintenance action from the instance in time where the maintenance action is being started until it is concluded. The structure of the MRec is defined in the ASN.1 fragment below:

```
MRec ::= SEQUENCE {
     status         MRecStatus,         -- maintenance action status
     ppus           PID,                -- ID of protocol entity
     srid           PID,                -- signalling relation identifier
                                        -- is ID of signalling transport entity
     ceid           CEID,               -- connection element identifier
     disp           BOOLEAN }           -- TRUE: originator is layer management
```

B.4.2.2 Primitives between Nodal Function 1 and the Protocol Entities

The interface to the AAL type 2 served user is defined in 5.1. The interface to layer management is defined in 5.3. The interface between the AAL type 2 signalling entity and the AAL type 2 protocol entities are summarized in Table B.2 and are defined after the table.

Table B.2/Q.2630.1 – Primitives and parameters exchanged between the Nodal Function 1 entity and the Protocol Entities

Primitive 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-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				

a) **IPP-ESTABLISH**

A newly created incoming protocol entity indicates the establish request (ERQ) message to Nodal Function 1 (with the indication primitive) together with the nodal signalling association identifier "sri". If the establishment terminates with a served user at this node or if the AAL type 2 nodes beyond the succeeding link acknowledged the establishment, the establish confirm (ECF) message is conveyed to the incoming protocol entity (with the response primitive) for transmission on the preceeding link.

b) **IPP-REJECT**

If the establishment can not be completed at this AAL type 2 node or at any AAL type 2 node beyond the succeeding link, the establishment is rejected with a release confirm (RLC) message that is conveyed to the incoming protocol entity (with the request primitive) for transmission on the preceeding link.

c) **OPP-ESTABLISH**

If an establishment is to be continued on a succeeding link the establish request (ERQ) message is conveyed to the newly created outgoing protocol entity (with the request primitive). If the establish confirm (ECF) message is received by this outgoing protocol entity, it is communicated to Nodal Function 1 (with the confirm primitive).

d) **OPP-REJECT**

If the AAL type 2 nodes beyond the succeeding link cannot accept the establishment, the outgoing protocol entity receives a release confirm (RLC) message which is communicated to the Nodal Function 1 (with the indication primitive).

e) **XPP-RELEASE**

An incoming or outgoing protocol entity is instructed to start release procedures with a release request (REL) message, this message is conveyed to the protocol entity (with the request primitive). If an incoming or outgoing protocol entity receives such a release request (REL) message, this message is conveyed to Nodal Function 1 (with the indication primitive). After receipt of the release request (REL) message, the nodal function releases resources and confirms the release with a release confirm (RLC) message; this message is conveyed to the protocol entity (with the response primitive). If an incoming or outgoing protocol entity receives a release confirm (RLC) message, this message is conveyed to Nodal Function 1 (with the confirm primitive).

f) **XPP-TIMEOUT**

At several stages, an incoming or outgoing protocol entity keeps a timer running. If such a timer expires, this event is communicated to Nodal Function 1 (with the indication primitive); no parameters need to be conveyed.

g) **XPP-CONFUSE**

If AAL type 2 nodes on a different functional level communicate with each other, not all information transmitted by one node may be understood by the other node. In such cases, the compatibility mechanism may require the transmission of a confuse (CNF) message; this message is conveyed to the incoming or outgoing protocol entity (with the request primitive) for transmission. Upon receipt of a confuse (CNF) message by an incoming or outgoing protocol entity, this message is conveyed to Nodal Function 1 (with the indication primitive).

h) **XPP-UNRECOGNIZED**

If AAL type 2 nodes on a different functional level communicate with each other, not all information transmitted by one node may be understood by the other node. In such cases, an unrecognized message might be received by an incoming or outgoing protocol entity; this message is conveyed to Nodal Function 1 (with the indication primitive). The compatibility mechanism may require the relaying of this message on a further link; the unrecognized message is transmitted to the incoming or outgoing protocol entity (with the request primitive) for transmission.

i) **MPP-BLOCK**

A newly created maintenance protocol entity is requested to transmit a block request (BLO) message (with the request primitive). Upon receipt of such a block request (BLO) message, a newly created maintenance protocol entity conveys this message to Nodal Function 1 (with the indication primitive). The block confirm (BLC) message to be returned is conveyed to the maintenance protocol entity (with the response primitive) for transmission. If a maintenance protocol entity receives a block confirm (BLC) message, this message is conveyed to Nodal Function 1 (with the confirm primitive).

j) **MPP-UNBLOCK**

A newly created maintenance protocol entity is requested to transmit an unblock request (UBL) message (with the request primitive). Upon receipt of such a unblock request (UBL) message, a newly created maintenance protocol entity conveys this message to Nodal Function 1 (with the indication primitive). The unblock confirm (UBC) message to be returned is conveyed to the maintenance protocol entity (with the response primitive) for transmission. If a maintenance protocol entity receives an unblock confirm (UBC) message, this message is conveyed to Nodal Function 1 (with the confirm primitive).

k) **MPP-RESET**

A newly created maintenance protocol entity is requested to transmit a reset request (RES) message (with the request primitive). Upon receipt of such a reset request (RES) message, a newly created maintenance protocol entity conveys this message to Nodal Function 1 (with the indication primitive). The reset confirm (RSC) message to be returned is conveyed to the maintenance protocol entity (with the response primitive) for transmission. If a maintenance protocol entity receives a reset confirm (RSC) message, this message is conveyed to Nodal Function 1 (with the confirm primitive).

l) **MPP-ERROR**

Errors detected by a maintenance protocol entity are brought to the attention of Nodal Function 1 (with the indication primitive); such errors include timeouts.

m) **TERMINATE**

At any time, Nodal Function 1 can terminate a maintenance protocol entity (with the request primitive); no parameters need to be conveyed

Besides the AAL type 2 signalling messages, the following parameters are conveyed:

aa) **sri**

The parameter "sri" is of type PId (SDL Process ID) is used in indication primitives and indicates the nodal signalling relation.

ab) **sqc**

The parameter "sqc" is used in MPP-BLOCK.request and MPP-RESET.response primitives to assure the sequence integrity for Block Requests (BLO) and the Reset Confirm (RCF) messages.

ac) **cause**

The parameter "cause" is used to indicate the type of error a maintenance protocol entity is indicating.

ad) **msg**

The parameter "msg" contains a complete unrecognized AAL type 2 signalling message.

The reaction to input signal events is described in parts 1 to 15 (of 25) in Figure B.3.

B.4.2.3 Procedures

The procedures are described in parts 16 to 21 (of 25) in Figure B.3.

The functions "**Compatibility**" (in parts 19 and 20 of Figure B.3) and "**MsgCompatibility**" (in part 21 of Figure B.3) perform the compatibility check for parameters (in recognized messages) and determine the compatibility action for unrecognized messages; they return a value that is defined by the following ASN.1 structure:

```
Compat ::= SEQUENCE {                    -- return value of procedure
                                            Compatibility
    course              ENUMERATED {pass,passcnf,dcrd,dcrdcnf,release},
    cause               CAUSE }          -- cause if not "pass"
```

The course takes the following values:

pass: Parameters are all recognized or may be passed on; unrecognized parameters that must be discarded are removed from the message within this function.

passcnf: The message contained unrecognized parameters which have been removed from the message within this function; a notification was requested.

dcrd: The message contained unrecognized parameters; non-recognition of (at least one of) the parameter required the discarding of the entire message.

dcrdcnf: Same as "dcrd" with notification requested in addition.

release: The message contained unrecognized parameters; non-recognition of (at least one of) the parameter required the release of the connection.

NOTE – The value "passcnf" is not returned by "MsgCompatibility".

The function "**LookupCRec**" (in part 16 of Figure B.3) 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; in addition, the crec...peer part must show the value "remote". 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.

The function "**LookupMRec**" (in part 16 of Figure B.3) 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.

The function "**FindNextCRec**" (in part 16 of Figure B.3) searches all records of type "CRec" 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). Before the pointer is returned, the contents of the record are adjusted as defined above for the procedure "LookupCRec".

The function "**FindNextPath**" (in part 16 of Figure B.3) 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).

The function "**GetNextParam**" (in part 16 of Figure B.3) 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).

The function "**GetNextField**" (in part 16 of Figure B.3) parses the parameter 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).

The function "**allocate**" and the procedure "**release**" (in part 16 of Figure B.3) act as placeholders for a memory management system that allocates and releases records of type "CRec" and "MRec".

The procedure "**StartReset**" (in part 17 of Figure B.3) calls the procedure "ResetConn" to release AAL type 2 connections affected by the reset. If more than a single channel is reset, the procedure "ResetUnblock" is called to set all affected AAL type 2 paths to "remotely unblocked". A Reset (RES) message is constructed and submitted to a newly created management protocol entity. Before returning, a new record of type "MRec" is allocated and filled.

The procedure "**StartBlocking**" (in part 17 of Figure B.3) calls on the procedure "BLOCKING" to record the requested blocking before constructing a Blocking (BLO) message and submitting the message to a newly created management protocol entity. Before returning, a new record of type "MRec" is allocated and filled.

The procedure "**StartUnblocking**" (in part 17 of Figure B.3) tests (by calling the procedure "Blocked") whether the indicated path(s) are already locally unblocked; if they are, the UNBLOCK.confirm primitive is issued terminating the unblocking. Otherwise, an Unblocking (UBL) message is constructed and submitted to a newly created management protocol entity. Before returning, a new record of type "MRec" is allocated and filled.

The procedure "**ResetConn**" (in part 18 of Figure B.3) calls the function "FindNextCRec" to find the AAL type 2 links affected by the reset operation. For each such link the associated protocol entity is terminated. In an AAL type 2 service endpoint, a RELEASE.confirm primitive is sent to the served user. In an AAL type 2 switch, release procedures are initiated towards the remote served user. It is

assured that resources allocated to such links are released; resources in the AAL type 2 path(s) that are reset are freed by the reset procedure itself.

The procedure "**ResetBlock**" (in part 18 of Figure B.3) assures that far all "locally blocked" AAL type 2 paths affected by a reset operation (initiated by another node) a blocking procedure is initiated (before the reset is confirmed).

The procedure "**ResetUnblock**" (in part 18 of Figure B.3) sets all "remotely blocked" AAL type 2 paths affected by the reset operation to "remotely unblocked".

B.4.2.4 Macros

The macros are described in parts 22 to 25 (of 25) of Figure B.3.

The macro "**Construct ERQmsg**" (in part 22 of Figure B.3) provides the necessary details for the construction of the ERQ message. In particular, parameters are added to the message dependent on the parameters in the ESTABLISH.request primitive from the served user.

The macro "**ReturnConfuse**" (in part 23 of Figure B.3) returns a confuse (CNF) message to the sender of the last message that caused the confusion.

The macro "**ReturnReject**" (in part 23 of Figure B.3) constructs a release confirm (RLC) message with a cause (CAU) parameter and returns it to the sender of the establish request (ERQ) message; the macro "**SendReject**" constructs a release confirm (RLC) message with a cause (CAU) parameter and sends it towards the sender of the establish request (ERQ) message (after the ERQ message has been processed, e.g. if the succeeding link has been reset.

The macro "**ReturnRelease**" (in part 23 of Figure B.3) constructs a release (REL) message and returns it to the sender of the last message; this may happen as a response to an unrecognized message or a message that caused confusion. The macro "**SendRelease**" constructs a release (REL) message and sends it on either the preceeding or succeeding link; this may happen as a response to an unrecognized message, or in conjunction with a reset operation.

The macro "**Extract ERQparameters**" (in part 24 of Figure B.3) extracts the information in an establish request (ERQ) message and prepares the parameters of the ESTABLISH.indication primitive.

The macro "**ValidREL**" (in part 24 of Figure B.3) assures that a cause (CAU) parameter is in the release (REL) message; if the parameter is absent, it is added within this macro ("Normal, unspecified", no diagnostics). The macro "**ValidRLC**" checks whether a cause (CAU) parameter is in the release complete (RLC) message when none is expected; if the parameter is present the cause (resulting from a confusion) is conveyed to layer management. The macro "**ValidRLCR**" assures that a cause (CAU) parameter is in the release complete (RLC) message when expected; if the parameter is absent, it is added within this macro ("Temporary failure", no diagnostics). The macro "**ValidCNF**" checks whether a cause (CAU) parameter is in the confusion (CNF) message; no parameter is added in this case.

The macros "**Construct ECFmsg**", "**Construct RLCmsg**", "**Construct RSCmsg**", "**Construct BLCmsg**", and "**Construct UBCmsg**" (in part 25 of Figure B.3) indicate that the respective messages are constructed without any parameters.

The macros "**Construct RESmsg**", "**Construct BLOmsg**", and "**Construct UBLmsg**" (in part 25 of Figure B.3) indicate that the respective messages are constructed containing a connection element identifier (CEID) parameter.

The macros "**Construct RELmsg**", "**Construct RLCmsgR**", "**Construct RSCmsgC**", "**Construct BLCmsgC**", "**Construct UBCmsgC**", and "**Construct RELmsg**" , and "**Construct CNFmsg**" (in part 25 of Figure B.3) indicate that the respective messages are constructed containing a cause (CAU) parameter.

NOTE – The release (REL) and confuse (CNF) message always contain a cause parameter. The release confirm (RLC) message contains a cause parameter in response to an establish request (ERQ) message (rejection of the establishment), or in response to a release message in conjunction with compatibility notifications. The reset confirm (RSC), block confirm (BLC), and unblock confirm (UBC) messages contain a cause parameter in conjunction with compatibility notifications.

B.4.3 Procedures in the nodal function

The function "**PathRes**" performs a connection admission control followed by the resource reservation on an incoming link (the preceeding link) during connection establishment; it 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 – 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 determines the availability of a route with enough AAL type 2 path resources to the succeeding AAL type 2 node. This may include a function to select a route which has available resources 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 (in the switched case);
- The Test Connection Indicator;
- Link information (link characteristics), and
- Other information (including SSCS Information).

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"
```

The function "**SwitchRoute**" (in the nodal function) 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"
```

The procedure "**PathRel**" releases resources associated with an AAL type 2 path; those are designated either by "CRec.incoming" or "CRec.outgoing".

The procedure "**SwitchRel**" releases resources associated with an AAL type 2 connection inside an AAL type 2 node.

The procedure "**ResetRel**" releases AAL type 2 connection resourcess associated with (one of) the channel(s) that is subject to a reset maintenance procedure.

The procedure "**AddCompatibility**" completes a message with the appropriate compatibility information in the message compatibility field as well as all the parameter compatibility fields.

The function "**PassOnPossible**" returns the value "TRUE" if pass on of the unrecognized message or of an unrecognized parameter within a recognized message is possible; otherwise, "FALSE" is returned.

The function "**ParamKnown**" returns the value "TRUE" if the parameter is recognized; otherwise, "FALSE" is returned.

The function "**FieldRecognized**" returns the value "TRUE" if the value of the field is recognized; otherwise, "FALSE" is returned.

The function "**PassConfuse**" returns the value "TRUE" if a received CNF message must be passed on; otherwise, "FALSE" is returned.

The function "**Valid**" returns the value "TRUE" if the CEID within a given signalling relation is valid; otherwise, "FALSE" is returned.

The function "**ValidateLink**" determines whether the CEID information in the ERQmsg designates a valid link in the nodal signalling association "sri". It returns the value "TRUE" if the information is valid ("FALSE" otherwise).

The procedure "**BLOCKING**" sets the state of a path (within a signalling relation) to locally blocked or unblocked and remotely blocked or unblocked.

The function "**Blocked**" returns the value "TRUE" if the AAL type 2 path indicated is locally or remotely blocked (as indicated with the 2nd parameter); otherwise, "FALSE" is returned.

B.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:

B.4.4.1 General message and parameter structure

```
Message ::= SEQUENCE {     -- general definition of the AAL type 2 signalling
                              messages
    dsaid                  SAID,                    -- destination
                                                       signalling
                                                       association ID
    msgID                  MessageID,               -- message identifier
    msgcompat              BIT STRING (SIZE (8)),   -- message compatibility
    parameters             SET OF Parameter }       -- message parameters
```

```
Parameter ::= SEQUENCE {   -- general definition of the AAL type 2 signalling
                              parameters
    paramid                BIT STRING (SIZE (8)),   -- parameter identifier
    paramcompat            BIT STRING (SIZE (8)),   -- parameter
                                                       compatibility
    paramlength            INTEGER (0 .. 255),      -- parameter length
    fields                 SEQUENCE OF Fields }     -- parameter fields
```

```
Fields ::= CHOICE {                 -- general definition of the AAL type 2
                                       signalling parameters
    fldtyp1                         FixedSizeField,
    fldtyp2                         VariableSizeField }
FixedSizeField ::= fixedfield       OCTET STRING (SIZE (1 .. 255))
VariableSizeField ::= SEQUENCE {
    fieldlength                     INTEGER (0 .. 254),             -- field
                                                                       length
    variablefield                   OCTET STRING (SIZE (0 .. 254)) }
```

```
MessageID                           BIT STRING (SIZE (8))           -- message
                                                                      identifier
erq                                 MessageID ::= '00000101'H       -- Establish
                                                                       request
ecf                                 MessageID ::= '00000100'H       -- Establish
                                                                       confirm
rel                                 MessageID ::= '00000111'H       -- Release
                                                                       request
rlc                                 MessageID ::= '00000110'H       -- Release
                                                                       confirm
res                                 MessageID ::= '00001001'H       -- Reset
                                                                       request
rsc                                 MessageID ::= '00001000'H       -- Reset
                                                                       confirm
blo                                 MessageID ::= '00000010'H       -- Block
                                                                       request
blc                                 MessageID ::= '00000001'H       -- Block
                                                                       confirm
ubl                                 MessageID ::= '00001011'H       -- Unblock
                                                                       request
ubc                                 MessageID ::= '00001010'H       -- Unblock
                                                                       confirm
cnf                                 MessageID ::= '00000011'H       -- Confusion
```

B.4.4.2 Detailed parameter structure

```
CAU ::= SEQUENCE {                  -- definition of the essentials of the cause
                                       (CAU) parameter
    org                             BIT STRING (SIZE (2)) ('00'B),  -- origin
                                                                       ITU-T
    value                           INTEGER (1 .. 127),             -- cause
                                                                       value
    diagnostics                     OCTET STRING (SIZE (0 .. 252)) }
```

```
CEID ::= SEQUENCE {                 -- definition of the essentials of the
                                       CEID parameter
    path                            AAL2Path,                       -- AAL type
                                                                       2 path
    cid                             CID }                           -- CID
```

```
OSAID ::=                           SAID          -- definition of the OSAID
                                                     parameter
SAID ::=                            OCTET STRING (SIZE (4))    -- definition of
                                                                  the SAID
unknown                             SAID ::= '00000000'H       -- definition of
                                                                  the 'unknown'
                                                                  value
```

```
TCI ::= OCTET STRING (SIZE (0))  -- definition of the essentials of the TCI
                                    parameter
```

```
-- The following parameters are handled but never interpreted in the
   SDL definition, hence, no details are needed
ESEA ::= OCTET STRING (SIZE (3 .. 17))     -- definition of the essentials of
                                              the ESEA parameter
NSEA ::= OCTET STRING (SIZE (20))          -- definition of the essentials of
                                              the NSEA parameter
ALC ::= OCTET STRING (SIZE (12))           -- definition of the essentials of
                                              the ALC parameter
SUGR ::= OCTET STRING (SIZE (4))           -- definition of the essentials of
the SUT parameter
SUT ::= OCTET STRING (SIZE (1 .. 255))     -- definition of the essentials of
                                              the SUGR parameter
SSIA ::= OCTET STRING (SIZE (8))           -- definition of the essentials of
                                              the SSIA parameter
SSIM ::= OCTET STRING (SIZE (3))           -- definition of the essentials of
                                              the SSIM parameter
SSISA ::= OCTET STRING (SIZE (14))         -- definition of the essentials of
                                              the SSISA parameter
SSISU ::= OCTET STRING (SIZE (7))          -- definition of the essentials of
                                              the SSISU parameter
```

NOTE – The parameters not interpreted by Nodal Function 1 are represented only summarily; these parameters with the exception of "SUGR" and "SUT" are interpreted by Nodal Function 2, though.

B.4.4.3 Detailed parameter list structure for the messages

```
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
     alc                ALC OPTIONAL,      -- link characteristics
     osaid              OSAID,             -- originating signalling
                                               association ID
     sugr               SUGR OPTIONAL,     -- served user generated reference
     sut                SUT OPTIONAL,      -- serverd user transport
     ssis   CHOICE {                       -- SSCS information
     ssia               SSIA,              -- service specific information
                                              (audio)
     ssim               SSIA,              -- service specific info.
                                              (multirate)
     ssisa              SSIA,              -- service specific info.
                                              (SAR-assured)
     ssisu              SSIA } OPTIONAL,   -- service specific info.
                                              (SAR-unassured)
     tci                TCI OPTIONAL }     -- test connection indicator
```

```
ECFmsg ::= SEQUENCE {   -- definition of the essentials of the ECF message
     osaid              OSAID }            -- originating signalling
                                           association ID
```

```
RELmsg ::= SEQUENCE {   -- definition of the essentials of the REL message
     cause              CAU }              -- cause
```

```
RLCmsg ::= SEQUENCE {   -- definition of the essentials of the RLC message
    cause               CAU OPTIONAL }     -- cause
```

```
RESmsg ::= SEQUENCE {   -- definition of the essentials of the RES message
    ceid                CEID,              -- connection element identifier
    osaid               OSAID }            -- originating signalling
                                           association ID
```

```
RSCmsg ::= SEQUENCE {   -- definition of the essentials of the RSC message
    cause               CAU OPTIONAL }     -- cause
```

```
BLOmsg ::= SEQUENCE {   -- definition of the essentials of the BLO message
    ceid                CEID,              -- connection element identifier
    osaid               OSAID }            -- originating signalling
                                              association ID
```

```
BLCmsg ::= SEQUENCE {   -- definition of the essentials of the BLC message
    cause               CAU OPTIONAL }     -- cause
```

```
UBLmsg ::= SEQUENCE {   -- definition of the essentials of the UBL message
    ceid                CEID,              -- connection element identifier
    osaid               OSAID }            -- originating signalling
                                              association ID
```

```
UBCmsg ::= SEQUENCE {   -- definition of the essentials of the UBC message
    cause               CAU OPTIONAL }     -- cause
```

```
CNFmsg ::= SEQUENCE {   -- definition of the essentials of the CNF message
    cause               CAU }              -- cause
```

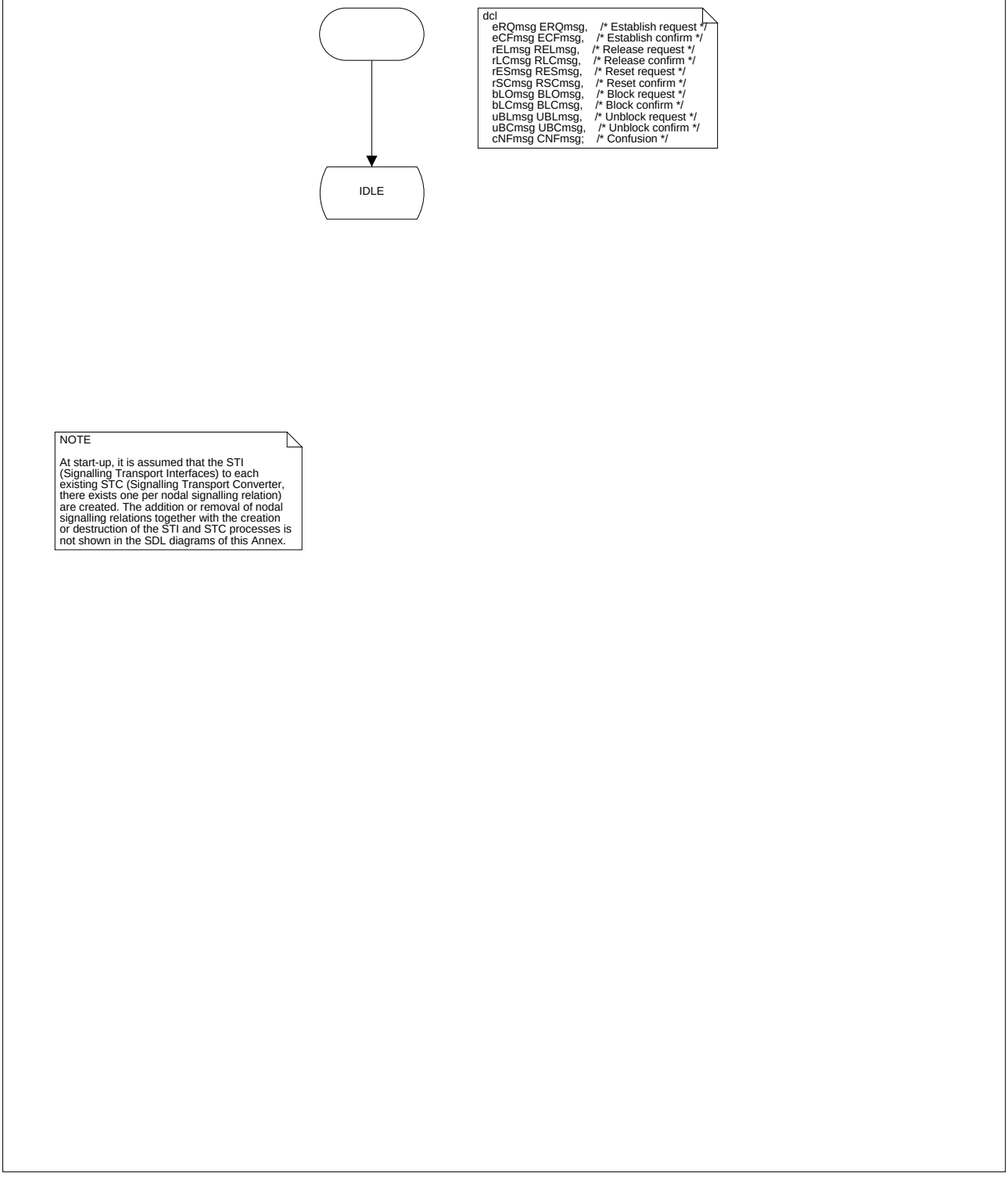


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 1 of 25)***

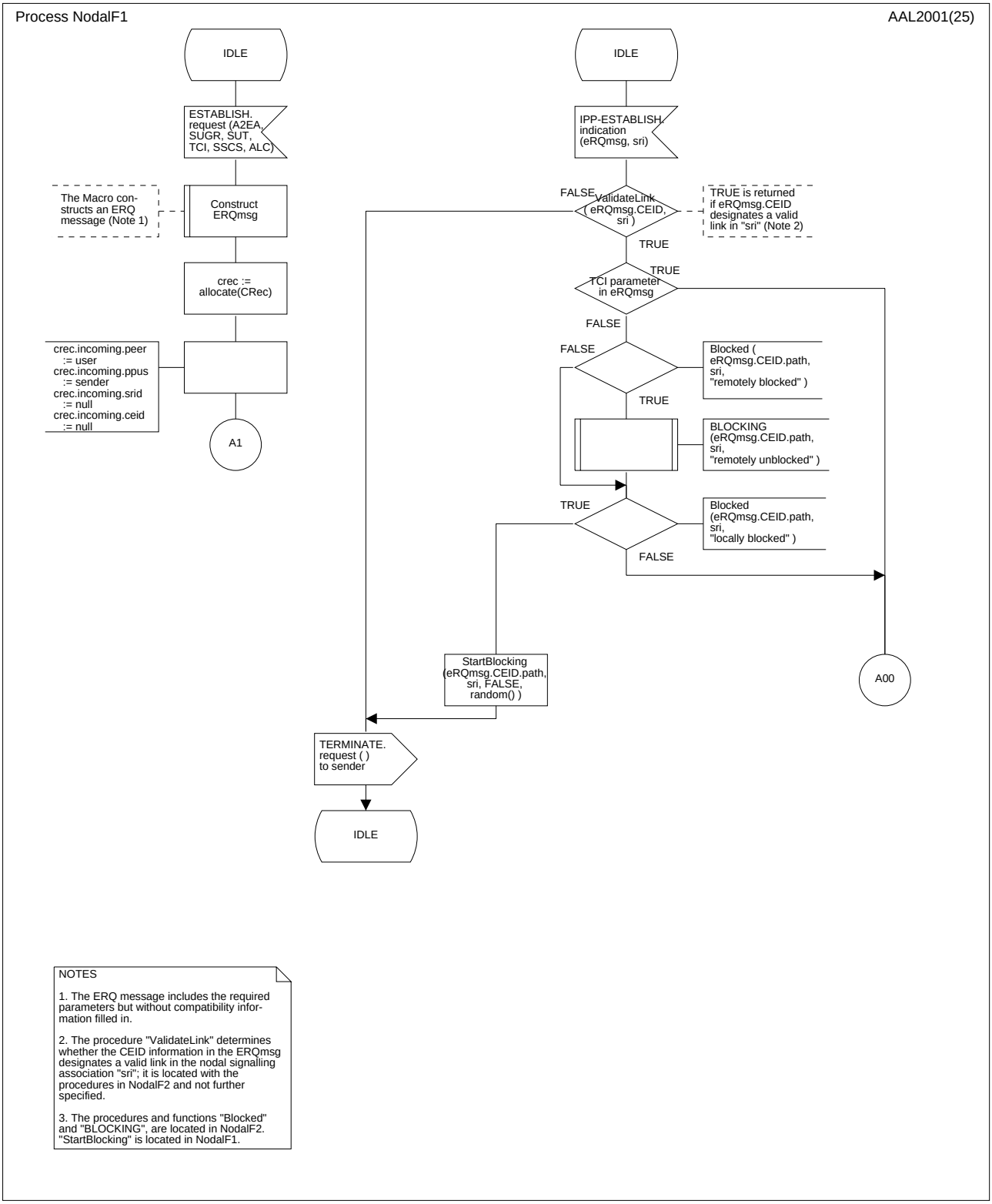


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 2 of 25)***

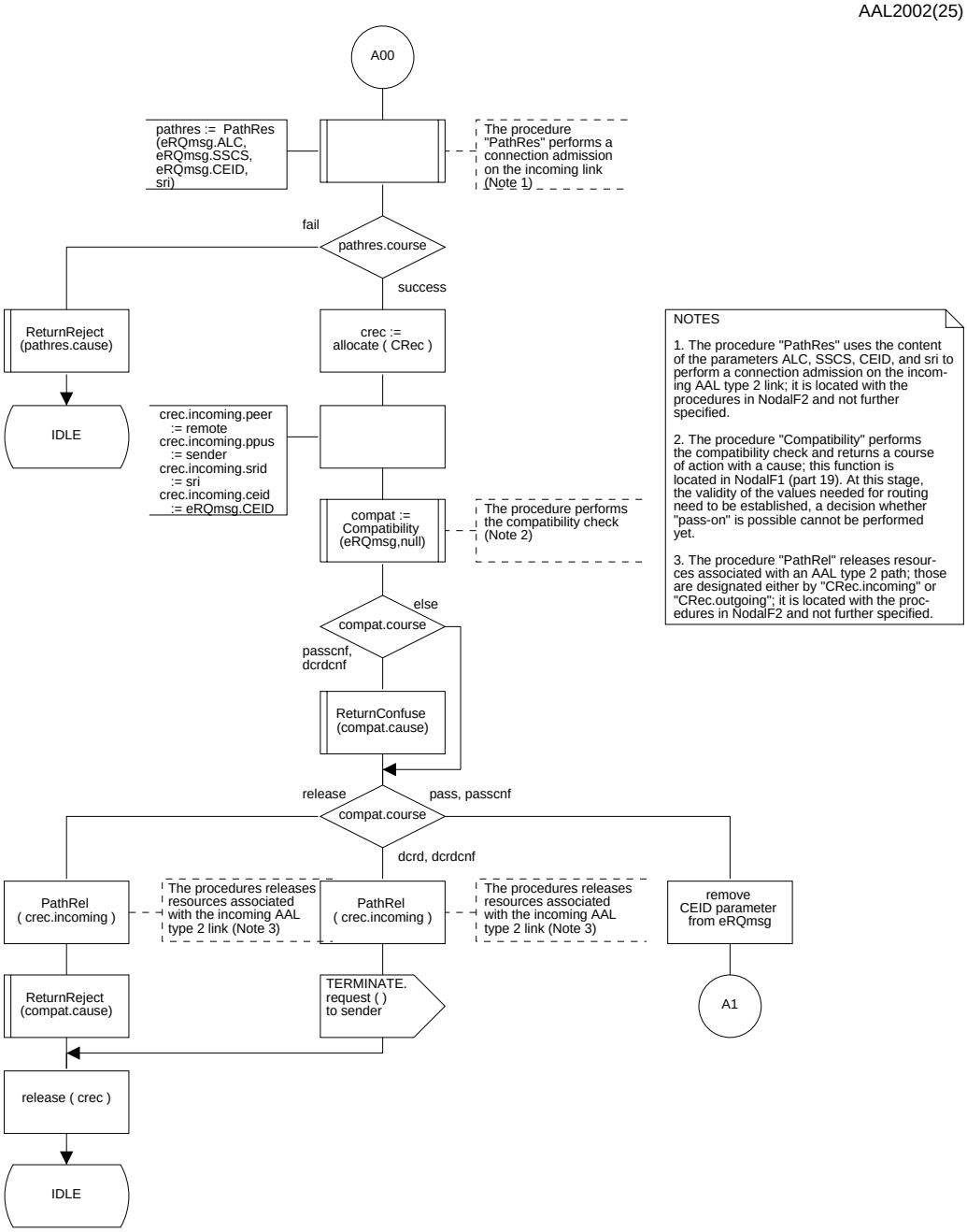


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 3 of 25)***

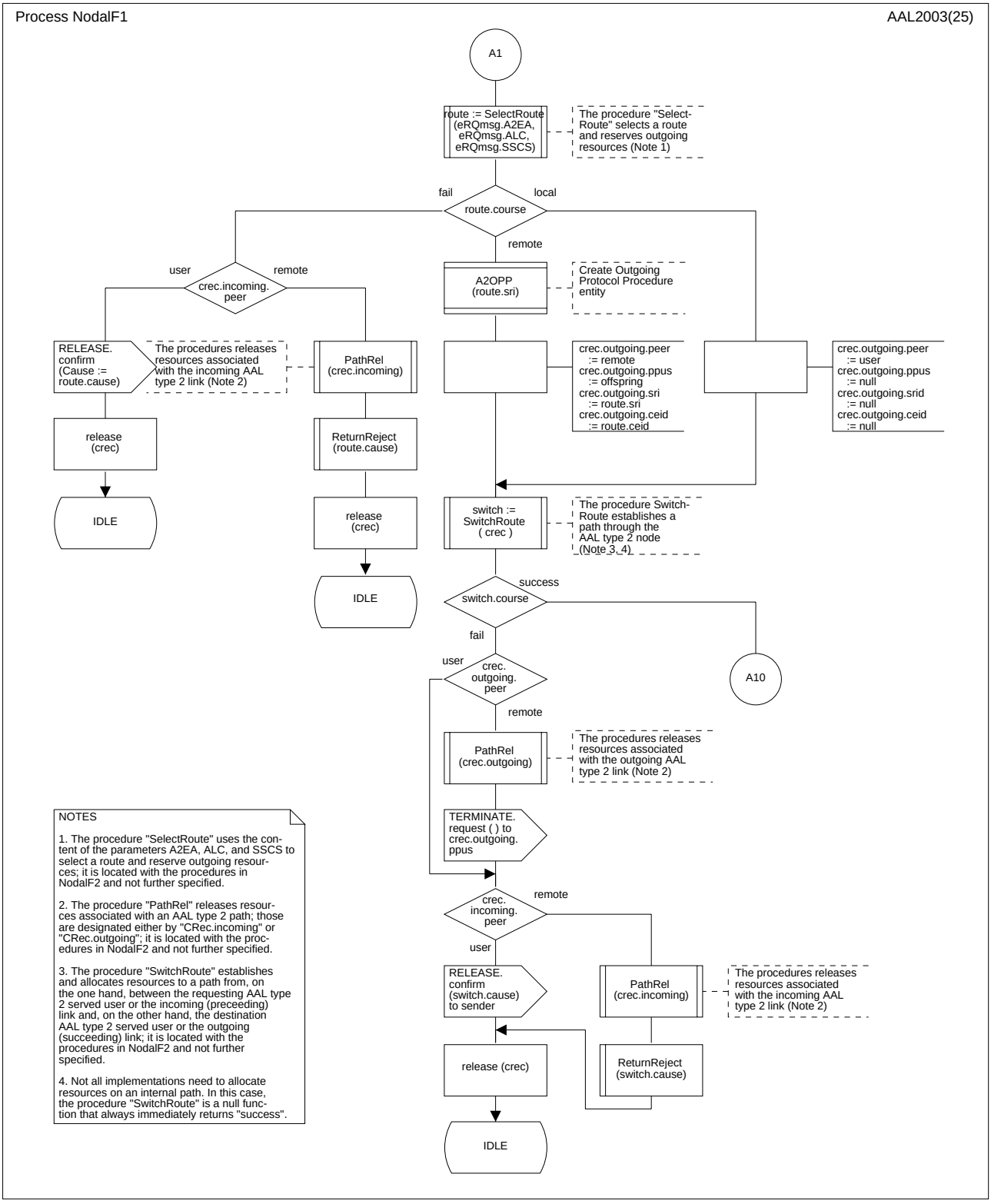


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 4 of 25)*

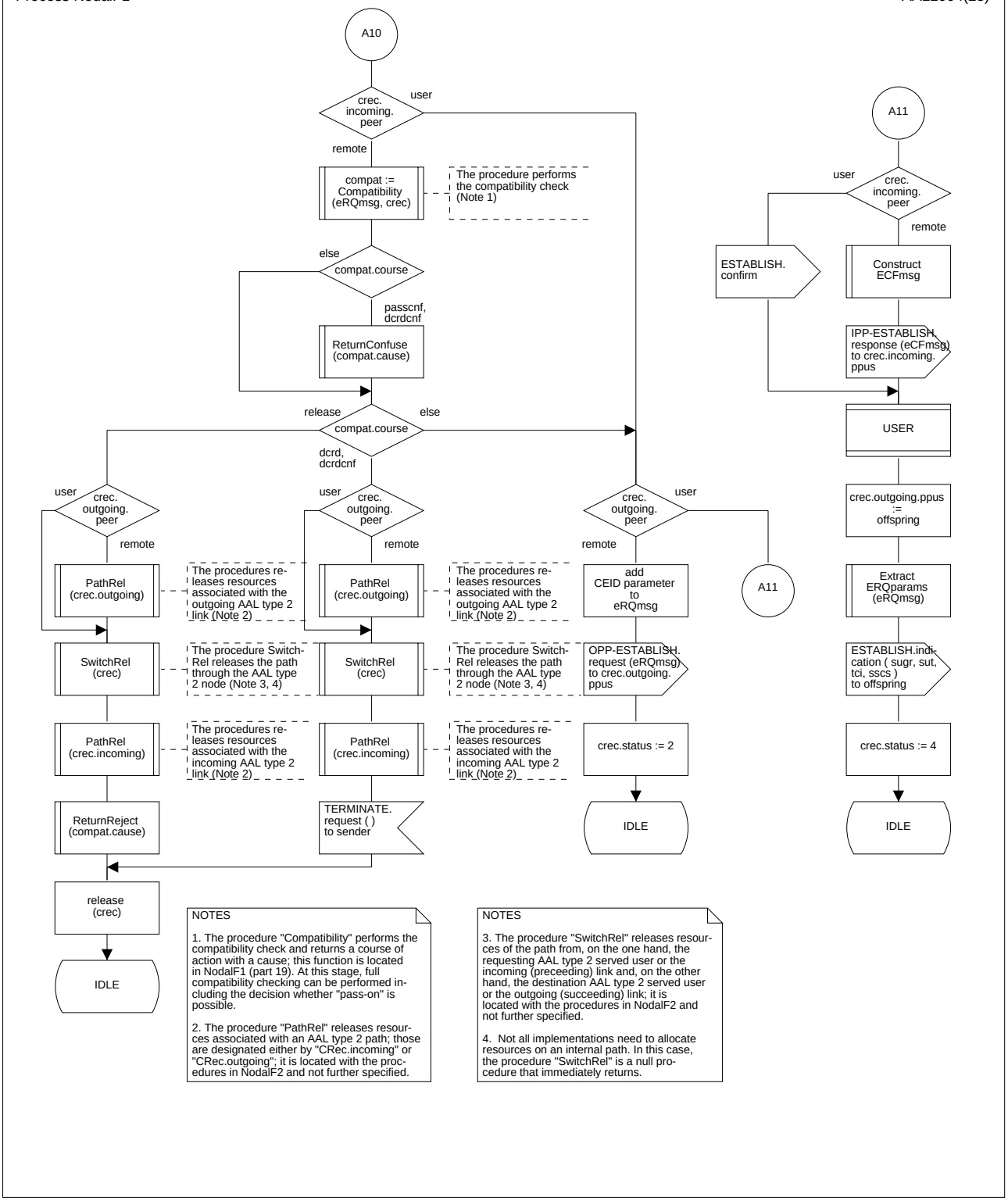


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 5 of 25)***

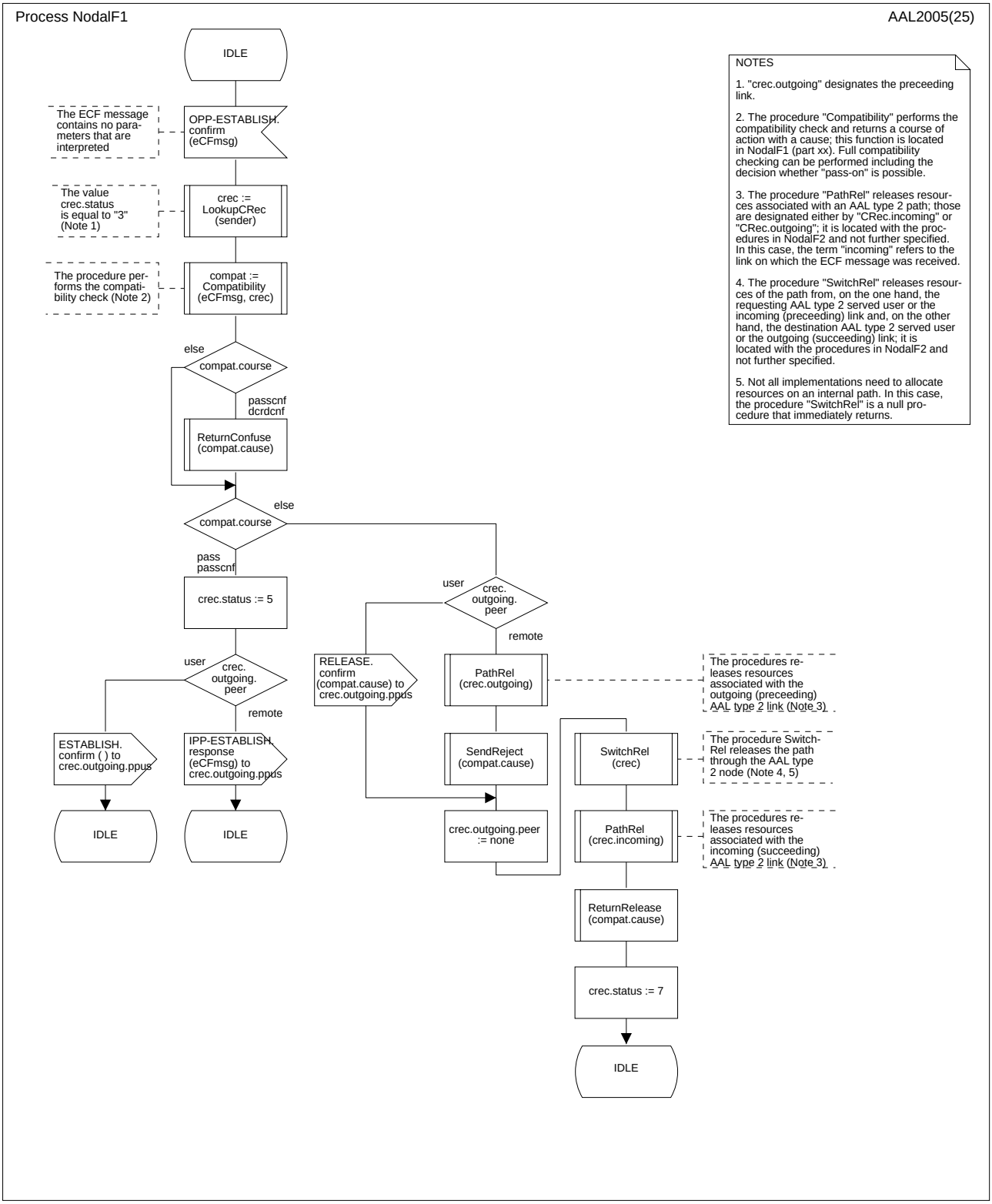
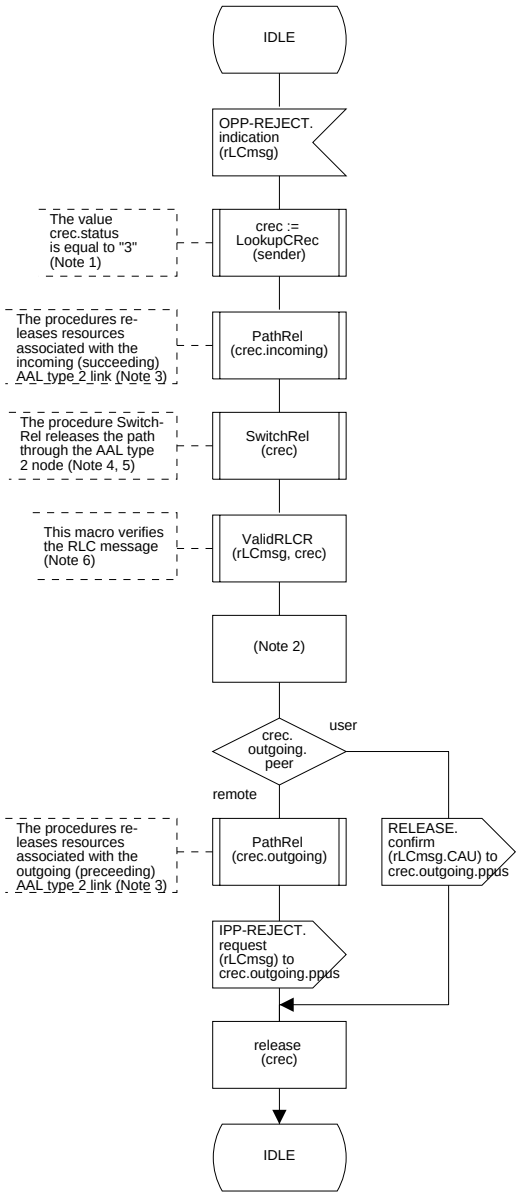


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 6 of 25)***



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; 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 B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 7 of 25)***

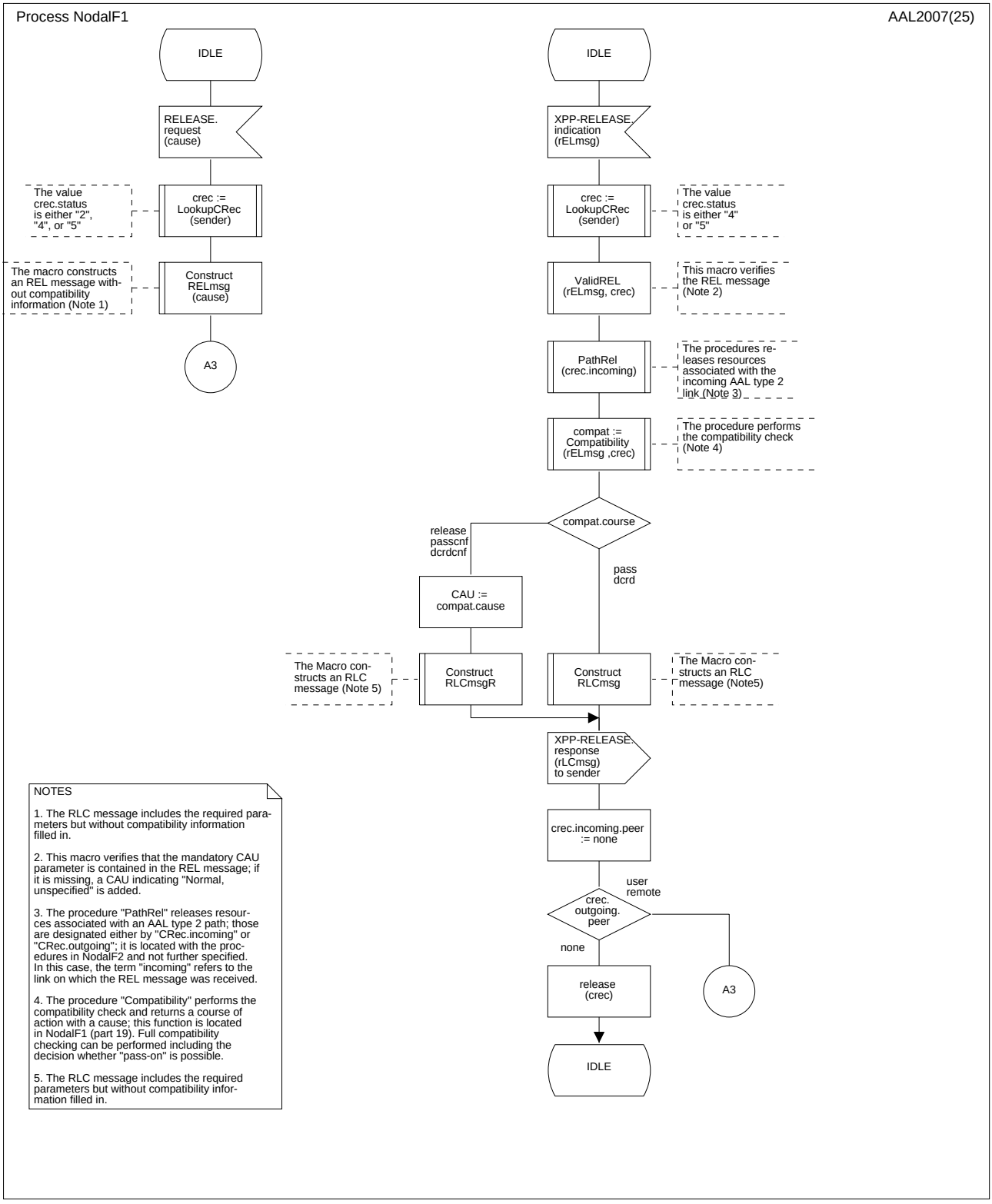
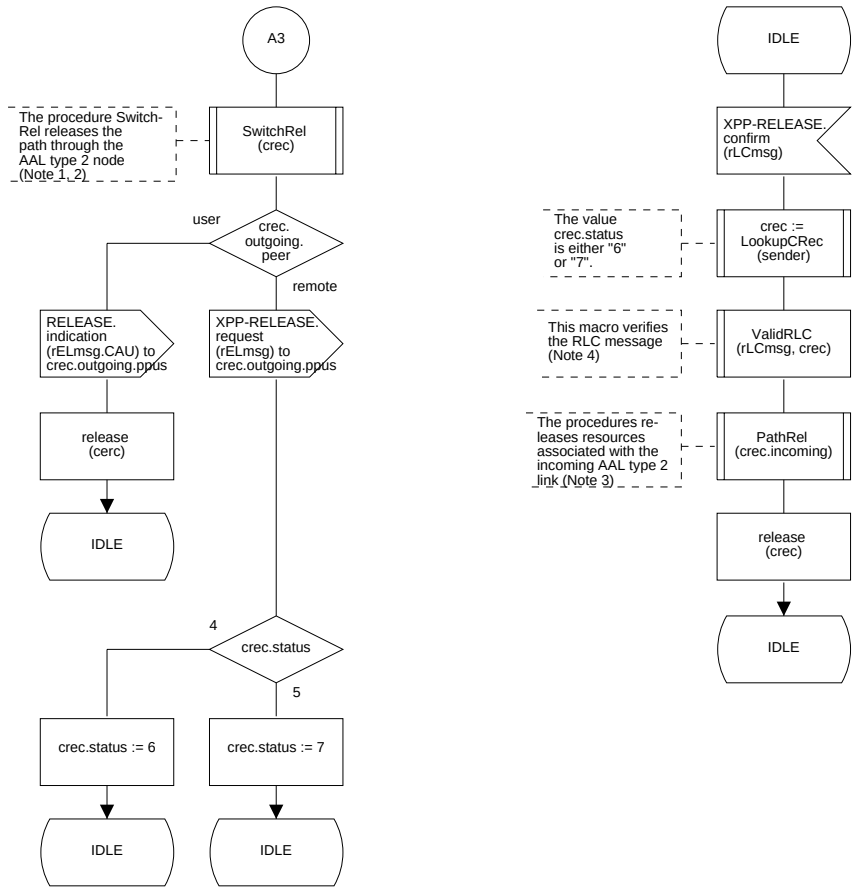


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 8 of 25)*



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 B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 9 of 25)***

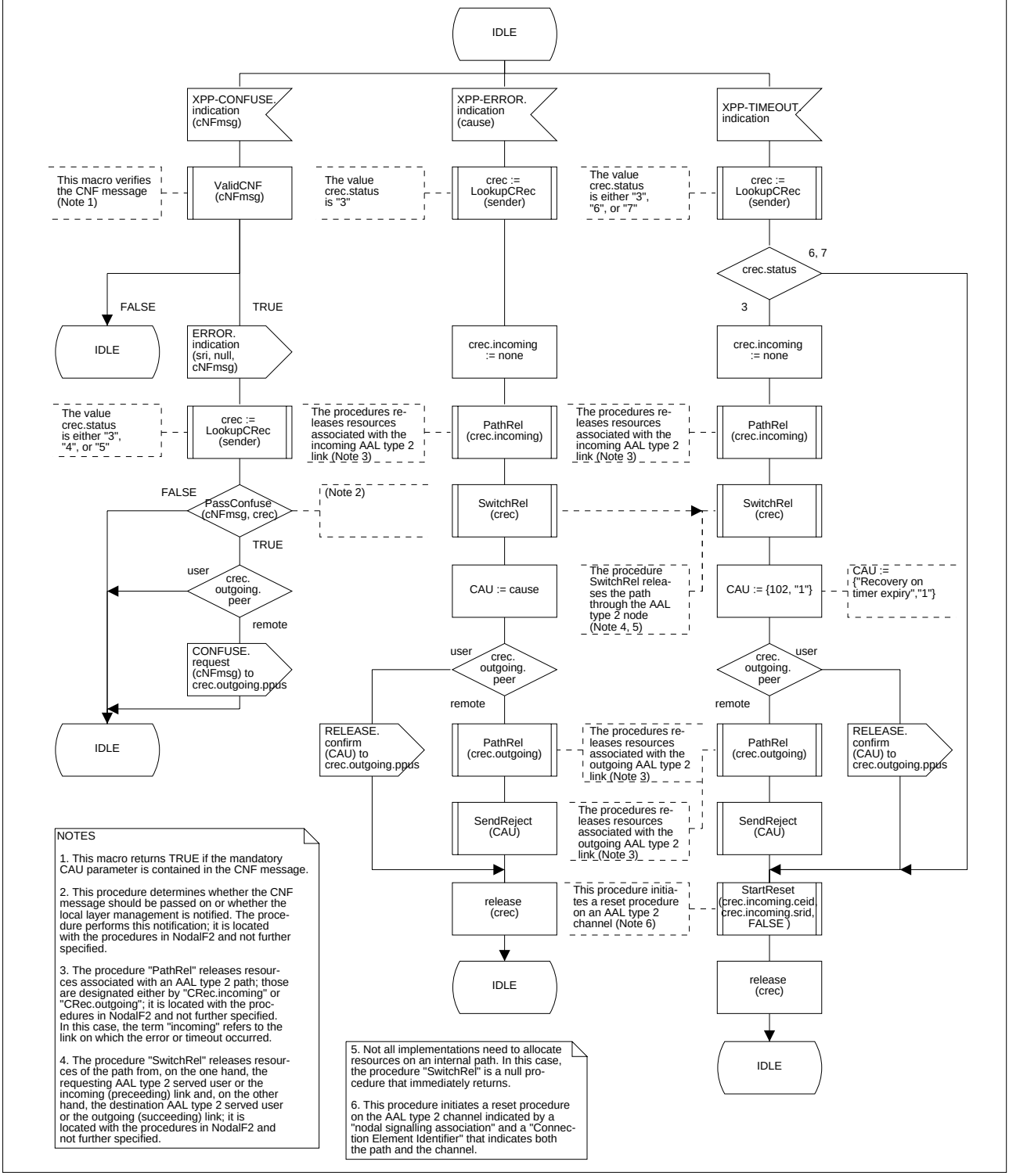


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 10 of 25)*

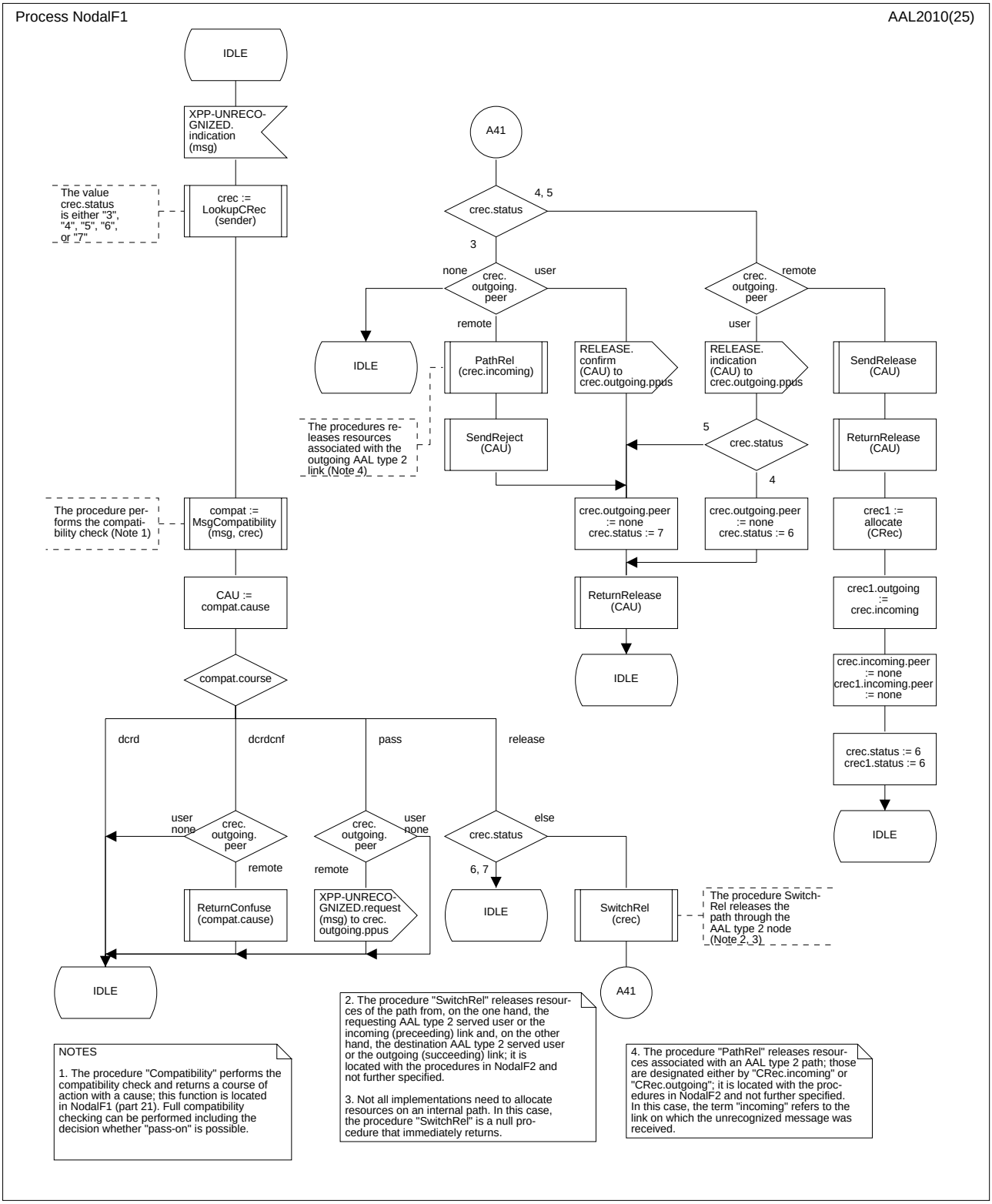


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 11 of 25)*

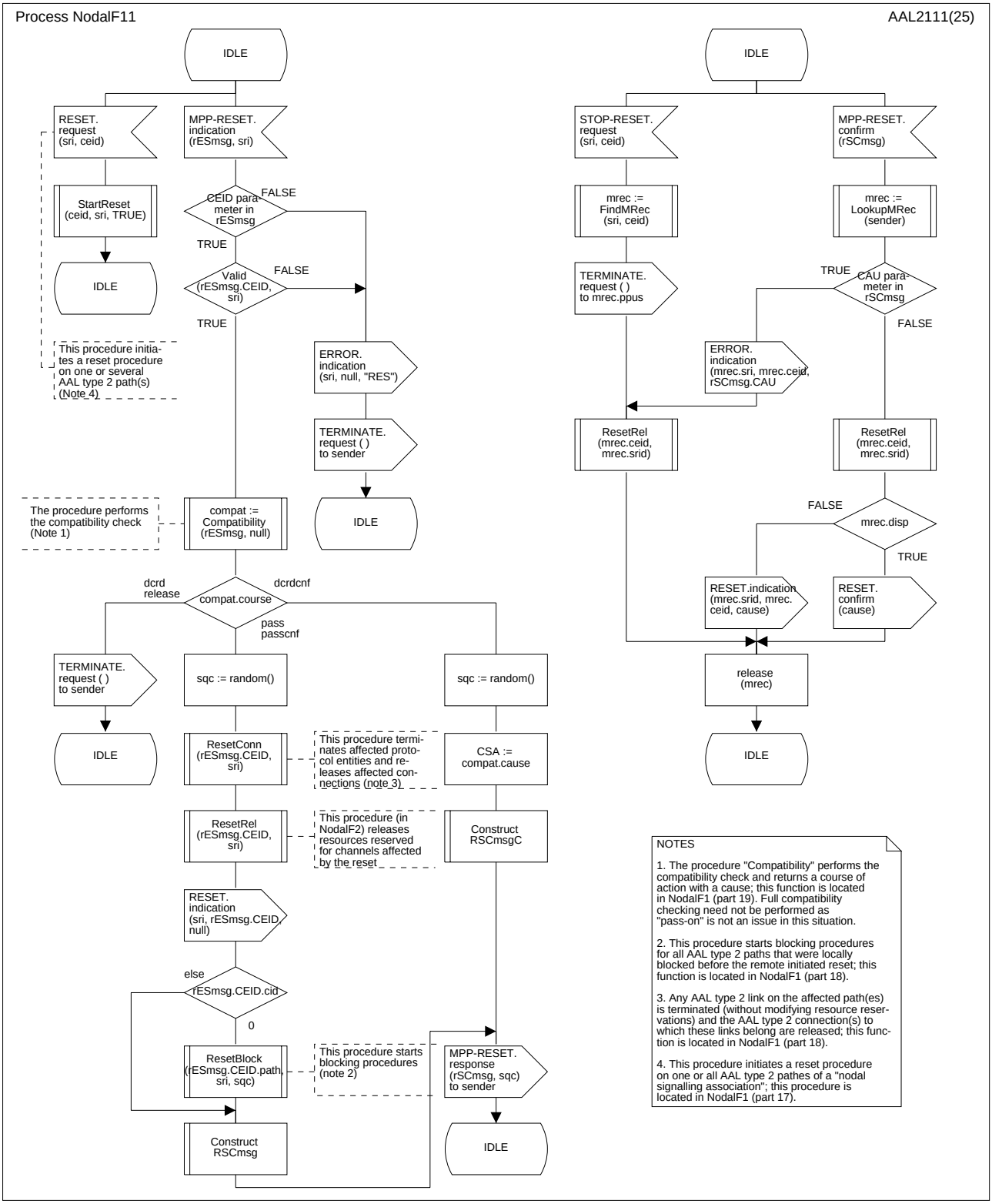


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 12 of 25)*

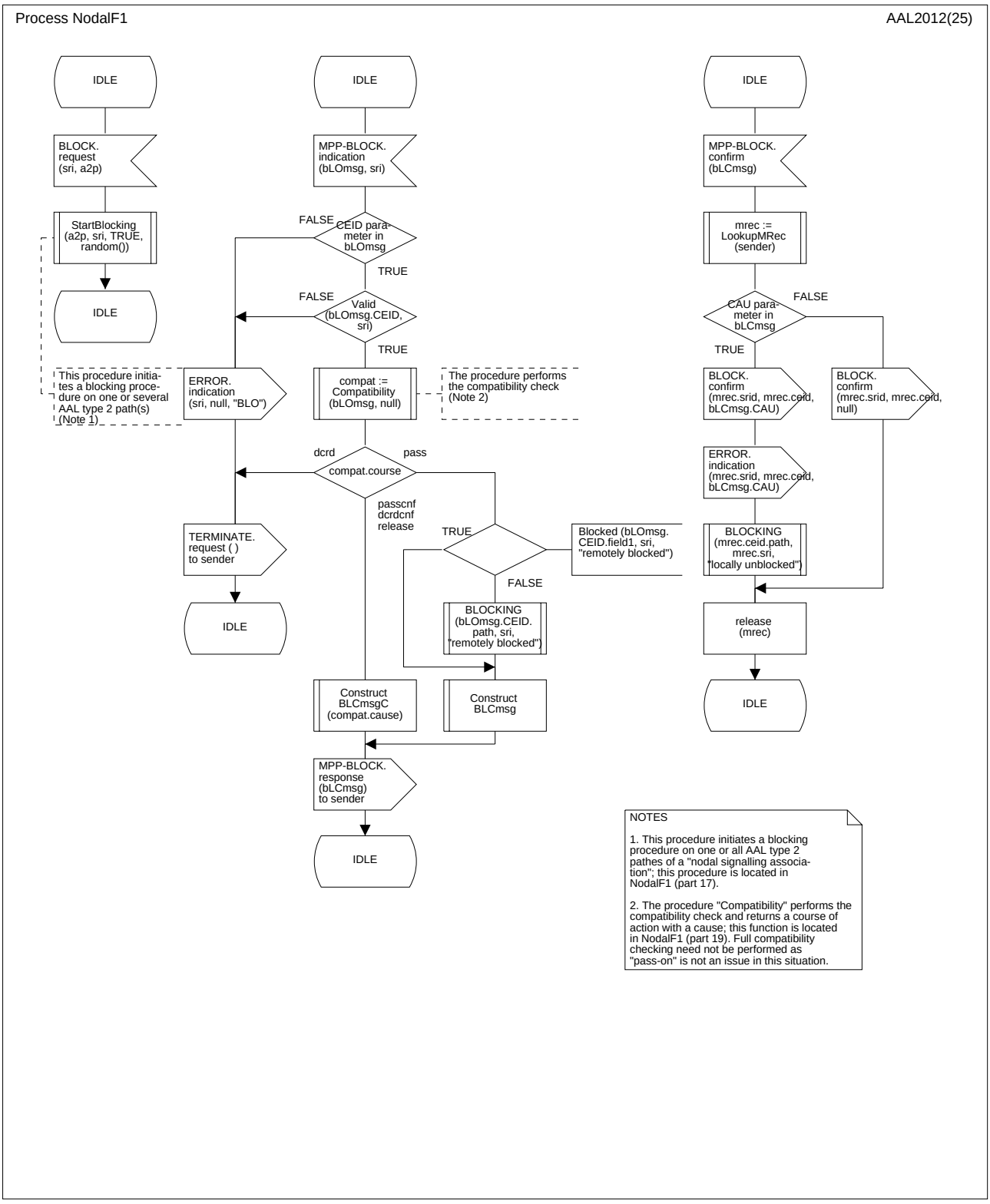


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 13 of 25)*

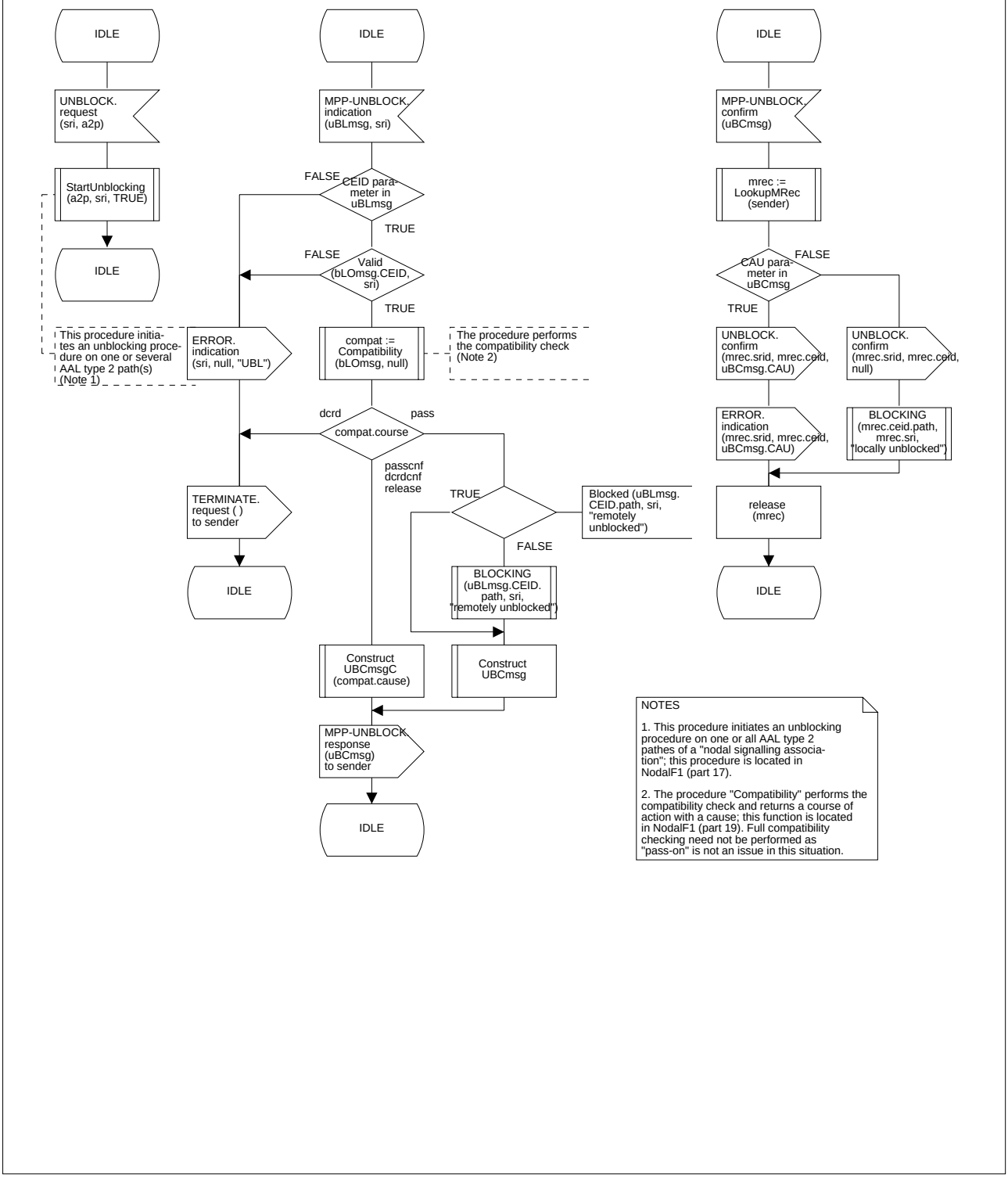


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 *(part 14 of 25)*

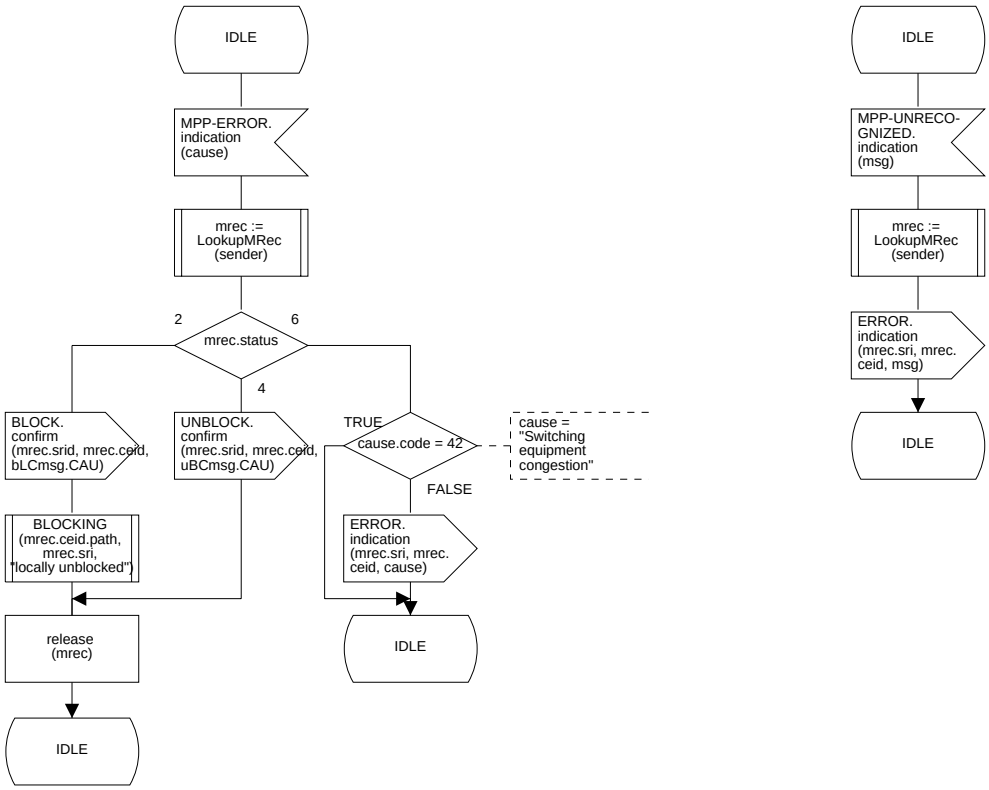


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 ***(part 15 of 25)***

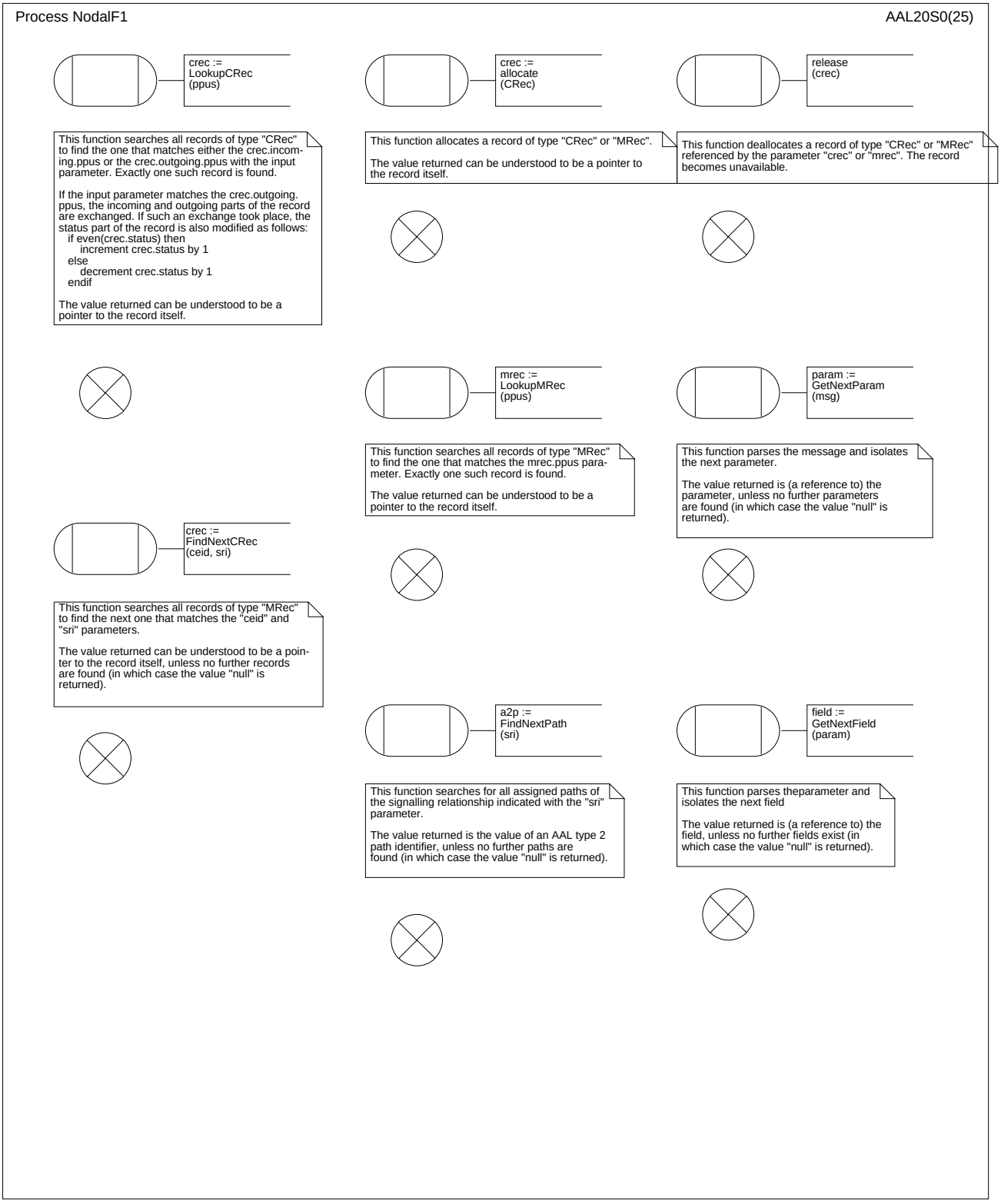


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (procedures) ***(part 16 of 25)***

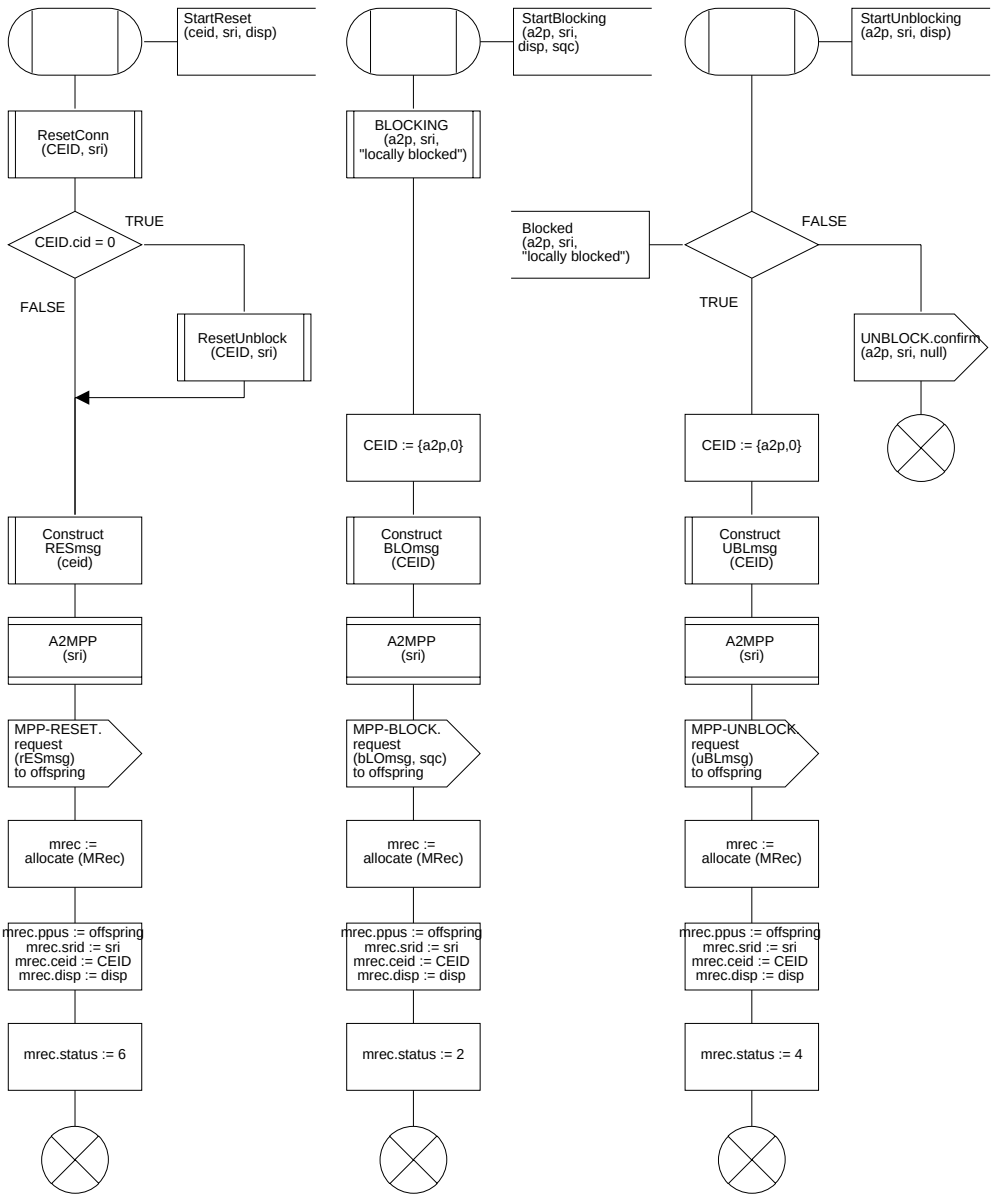


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (procedures) ***(part 17 of 25)***

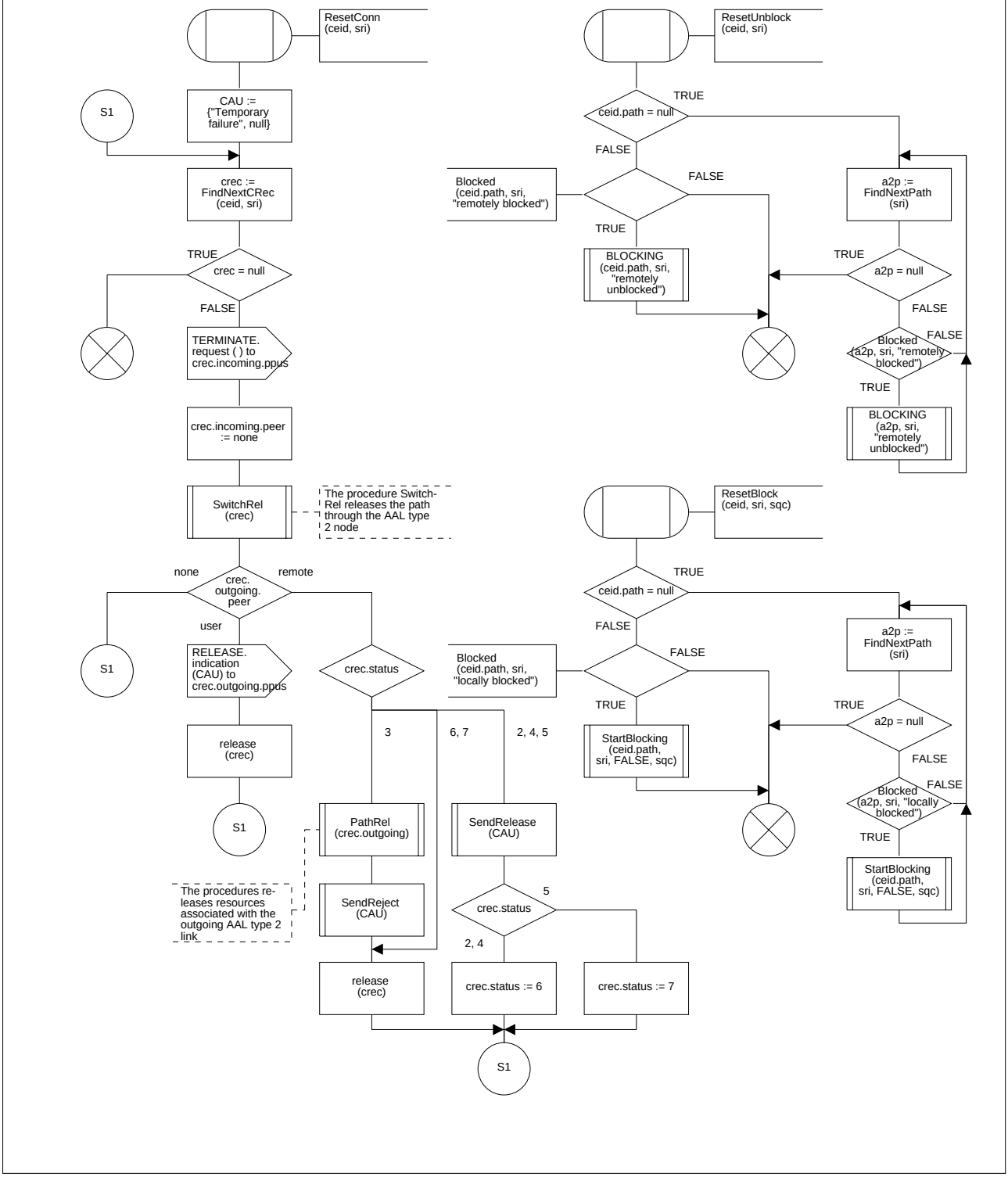


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (procedures) ***(part 18 of 25)***

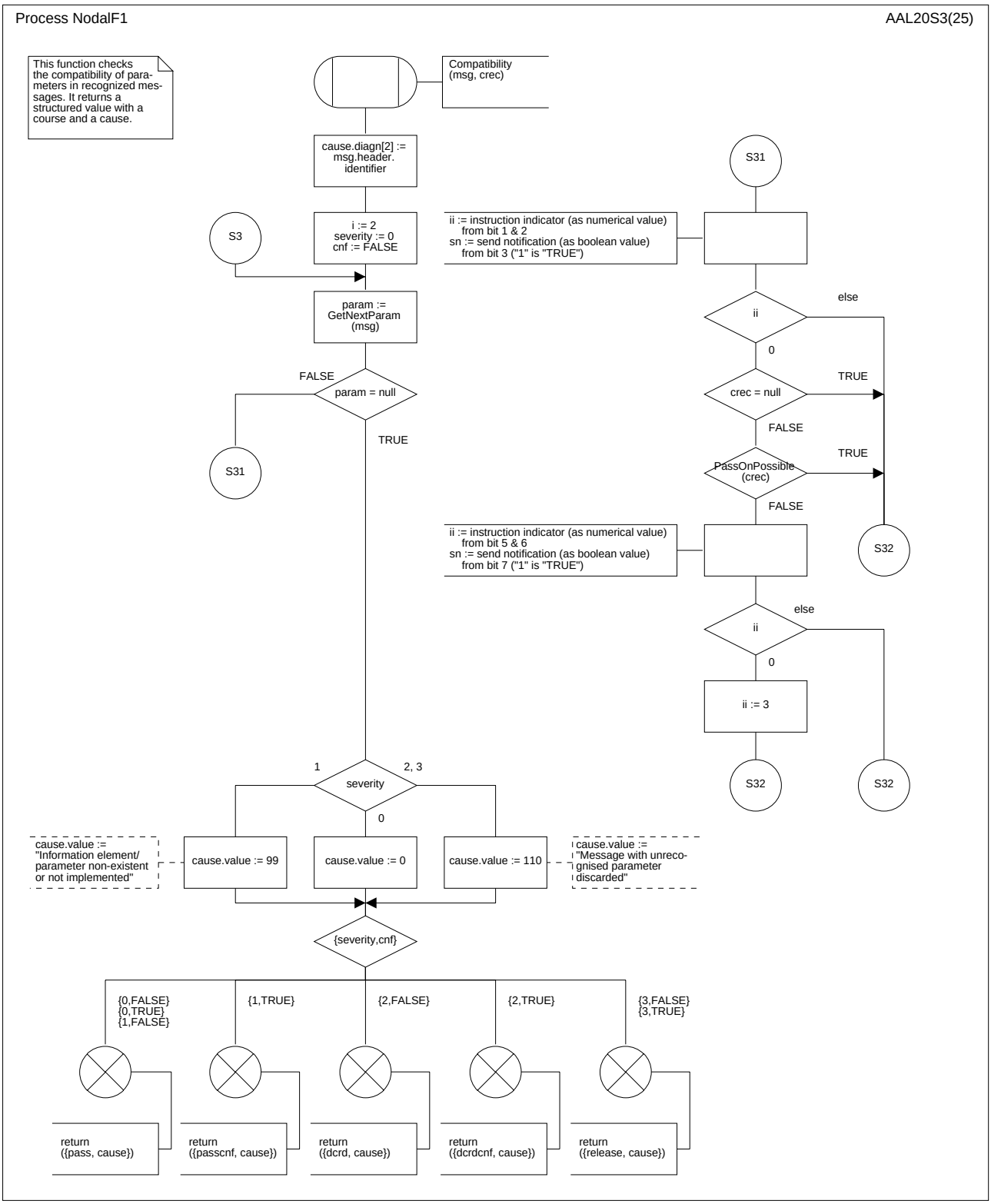


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (procedures) ***(part 19 of 25)***

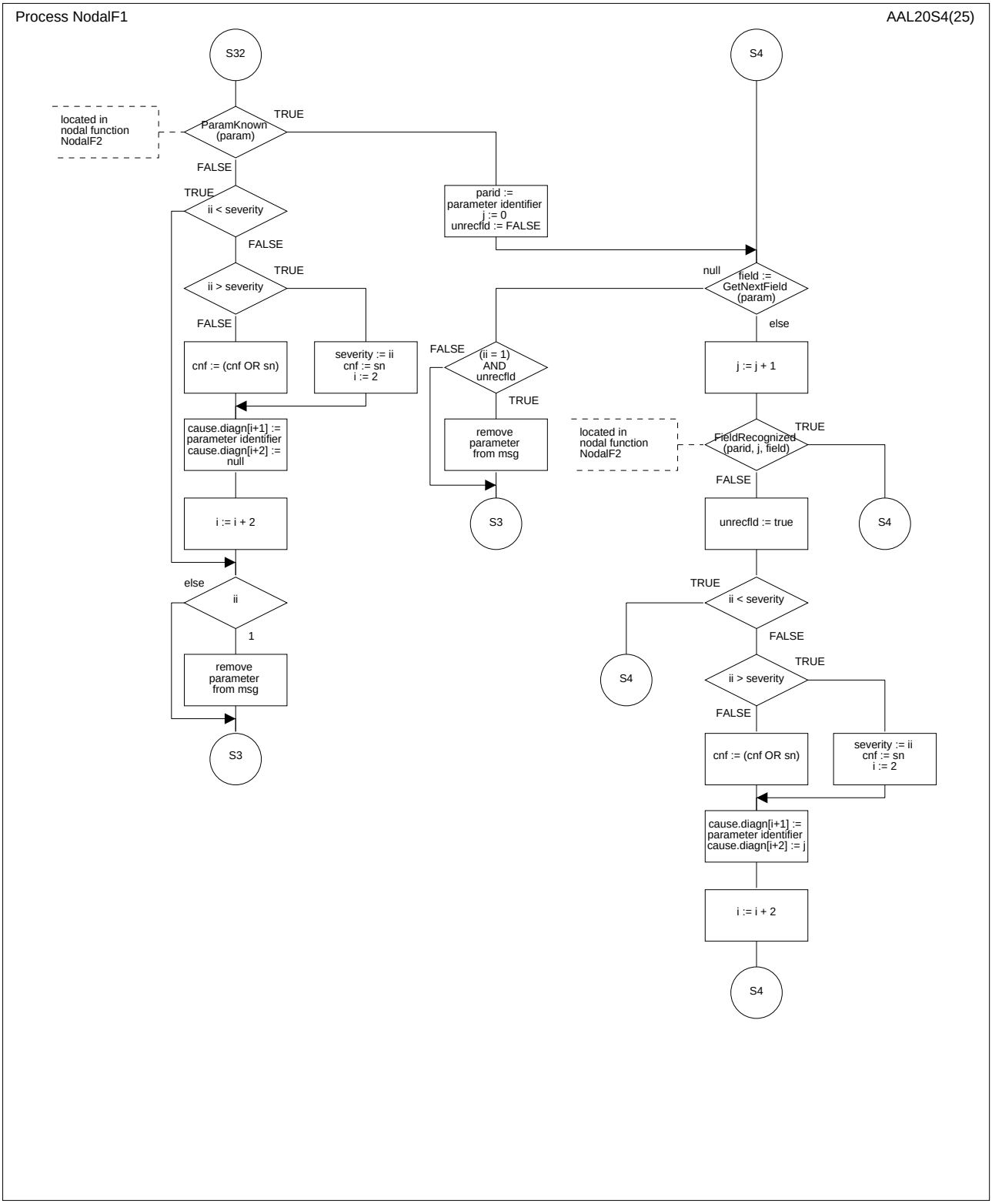


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (procedures) ***(part 20 of 25)***

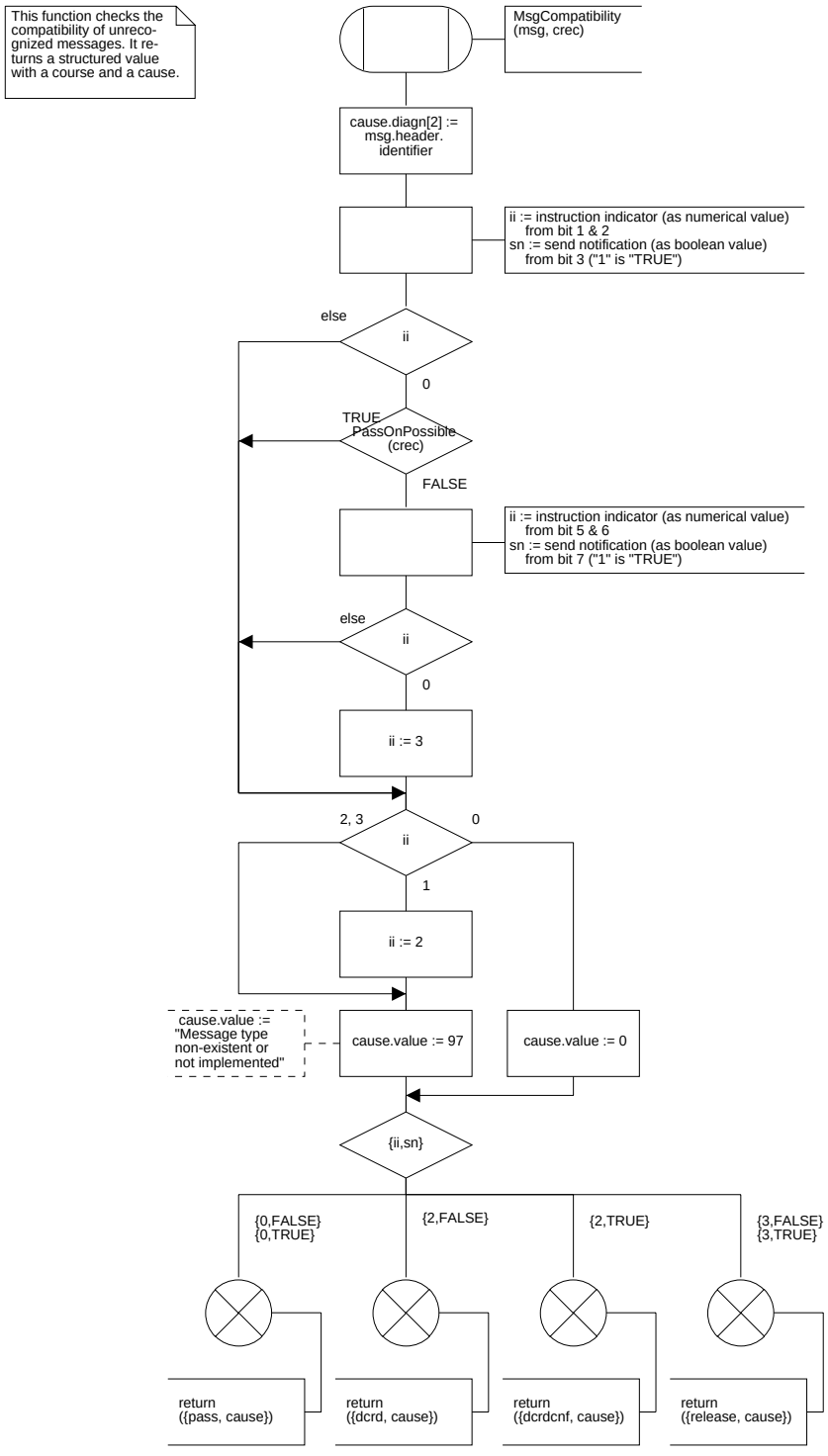


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (procedures) ***(part 21 of 25)***

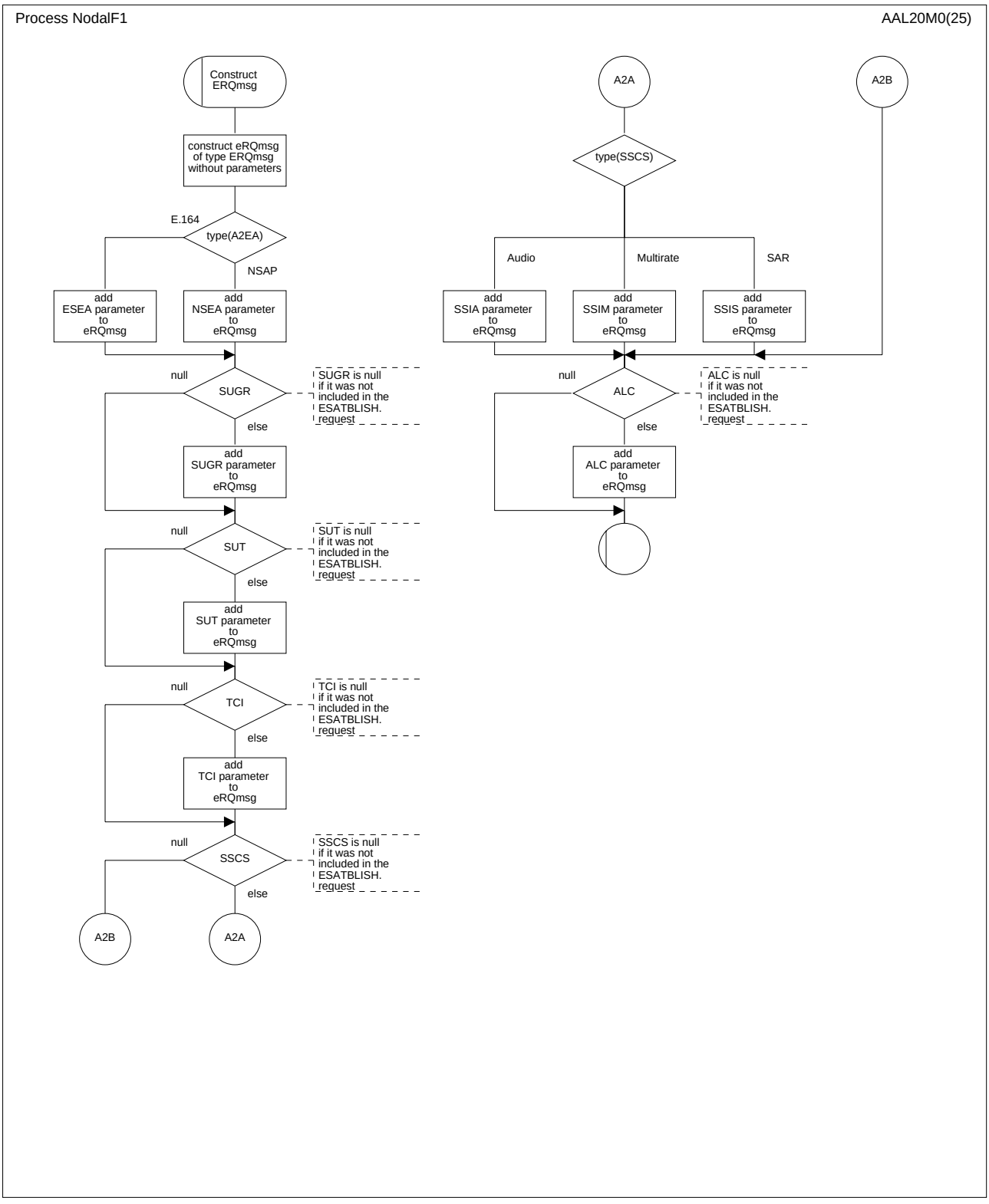


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (macros) ***(part 22 of 25)***

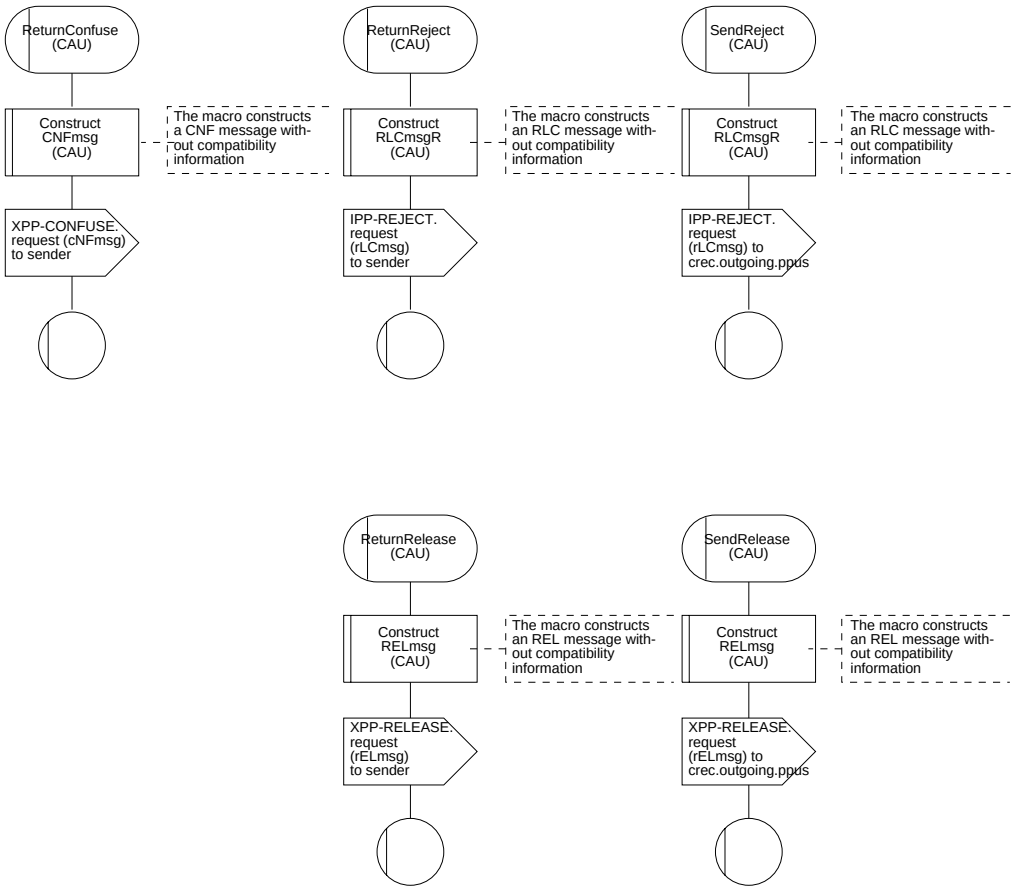


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (macros) ***(part 23 of 25)***

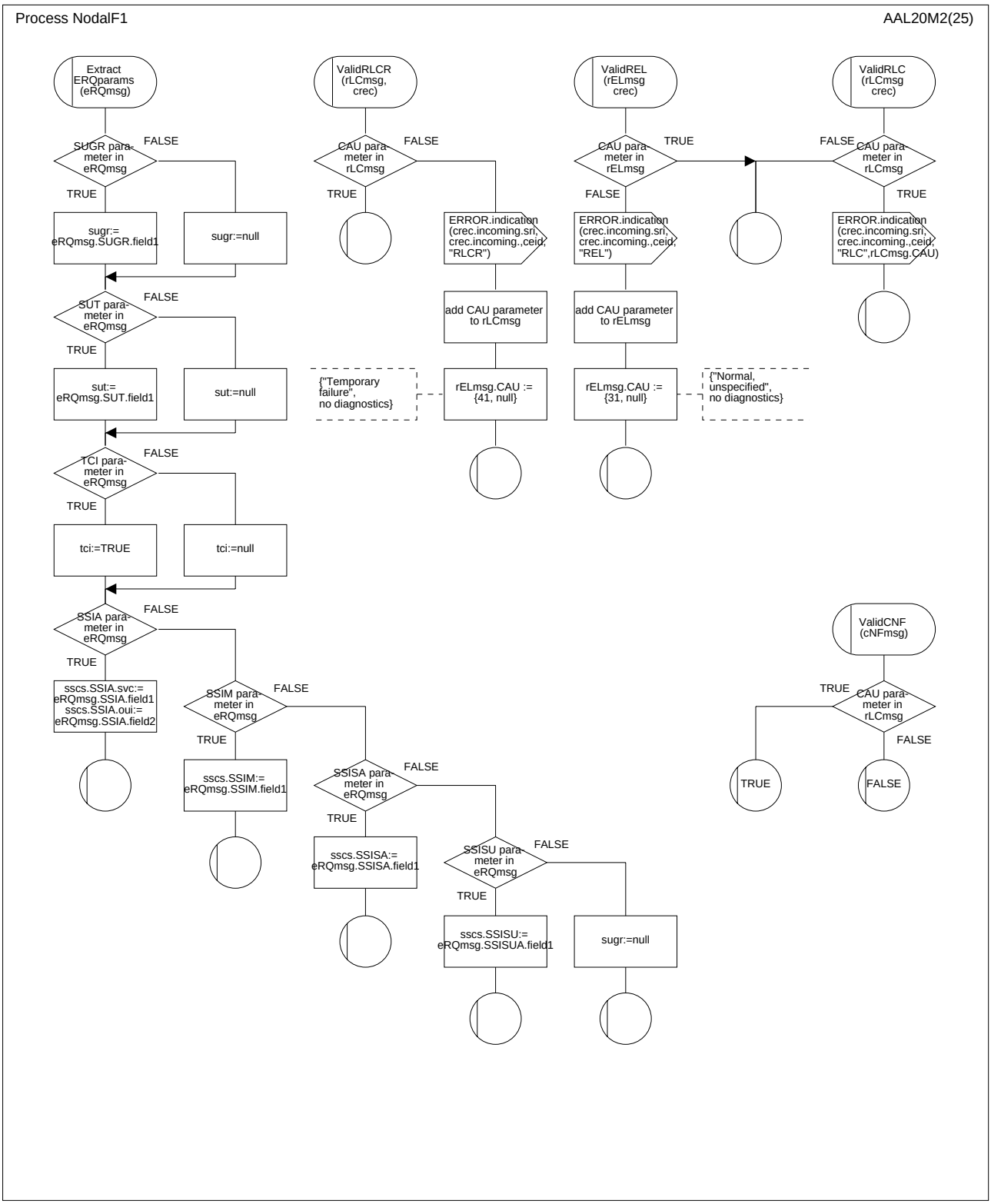


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (macros) *(part 24 of 25)*

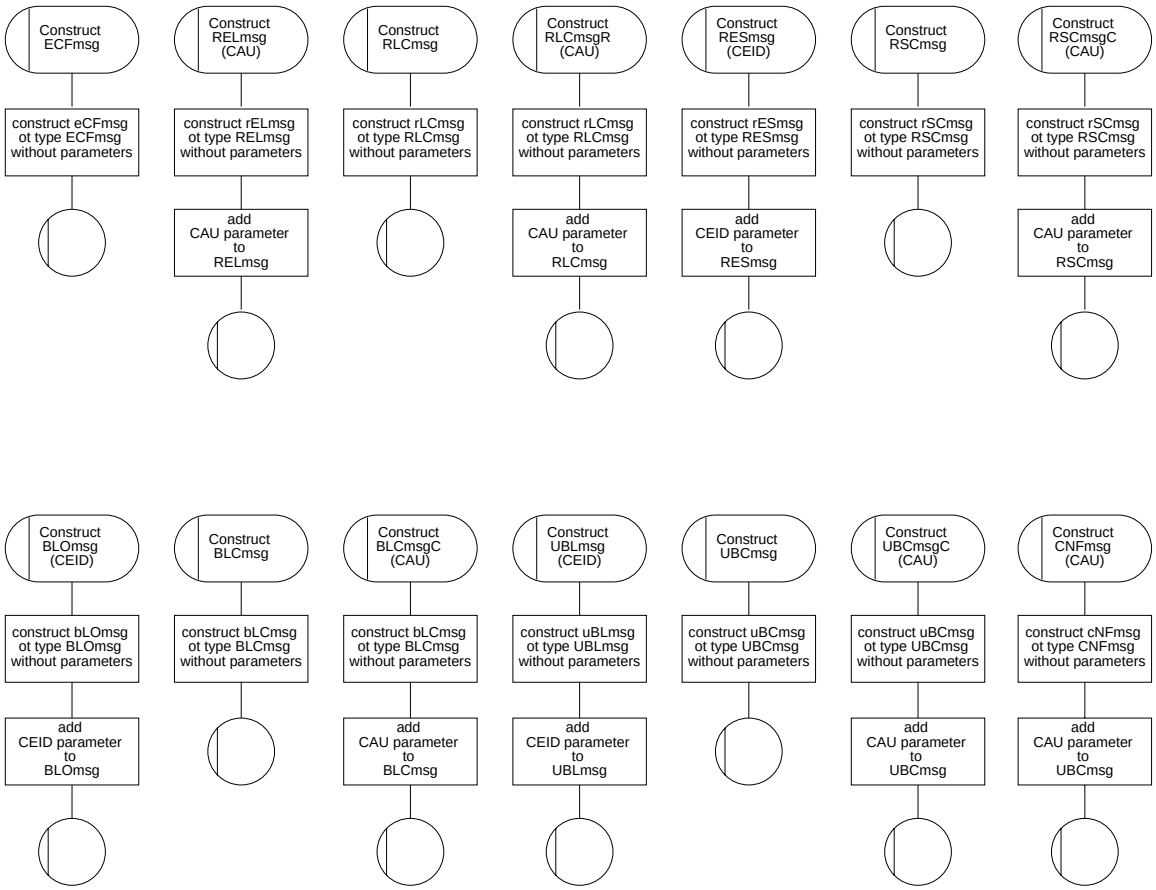


Figure B.3/Q.2630.1 – SDL diagram of the Nodal Function 1 (macros) ***(part 25 of 25)***

B.5 SDL diagrams for the protocol entities

B.5.1 Introduction

In Figure B.2 (part 3 of 3) it is shown that the protocol entity is separated into:

a) *Outgoing protocol procedures*

This entity contains all functionality that is defined in detail in 8.3.2 and, therefore, can be specified precisely in SDL diagrams.

b) *Incoming protocol procedures*

This entity contains all functionality that is defined in detail in 8.3.3 and, therefore, can be specified precisely in SDL diagrams.

c) *Maintenance protocol procedures*

This entity contains all functionality that is defined in detail in 8.3.4 and, therefore, can be specified precisely in SDL diagrams.

d) *Signalling Transport Interface*

This entity is not specified explicitly in this Recommendation, nevertheless, it is required for receiving primitives and messages from the signalling transport and dispatching them to the appropriate protocol entity instance and the "IN-SERVICE", "OUT-OF-SERVICE", and "CONGESTION" signals to Nodal Function 2. In some cases, such protocol entity instances have to be created by the Signalling Transport Interface (e.g. upon receipt of an ERQ, RES, BLO, or UBL message). This entity is also specified in SDL diagrams.

NOTE – In the transmit direction, this entity has no functionality.

Therefore, this clause defines the outgoing protocol procedures entity (see B.5.2.2), the incoming procedures entity (see B.5.2.3), the maintenance procedures entity (see B.5.2.4), and the signalling transport interface entity (see B.5.2.5).

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

B.5.2.1 Data structures

The SDL diagrams in this clause make use of the ASN.1 definitions in B.4.2.5.

B.5.2.2 SDL diagrams for the Outgoing Protocol Procedures

The SDL diagrams for the outgoing protocol procedure is described in parts 1 to 5 in Figure B.4.

B.5.2.3 SDL diagrams for the Incoming Protocol Procedures

The SDL diagrams for the incoming protocol procedure is described in parts 1 to 5 in Figure B.5.

B.5.2.4 SDL diagrams for the Maintenance Protocol Procedures

The SDL diagrams for the maintenance protocol procedure is described in parts 1 to 5 in Figure B.6.

B.5.2.5 SDL diagrams for the Signalling Transport Interface

The SDL diagrams for the signalling transport interface is described in parts 1 to 2 in Figure B.7.

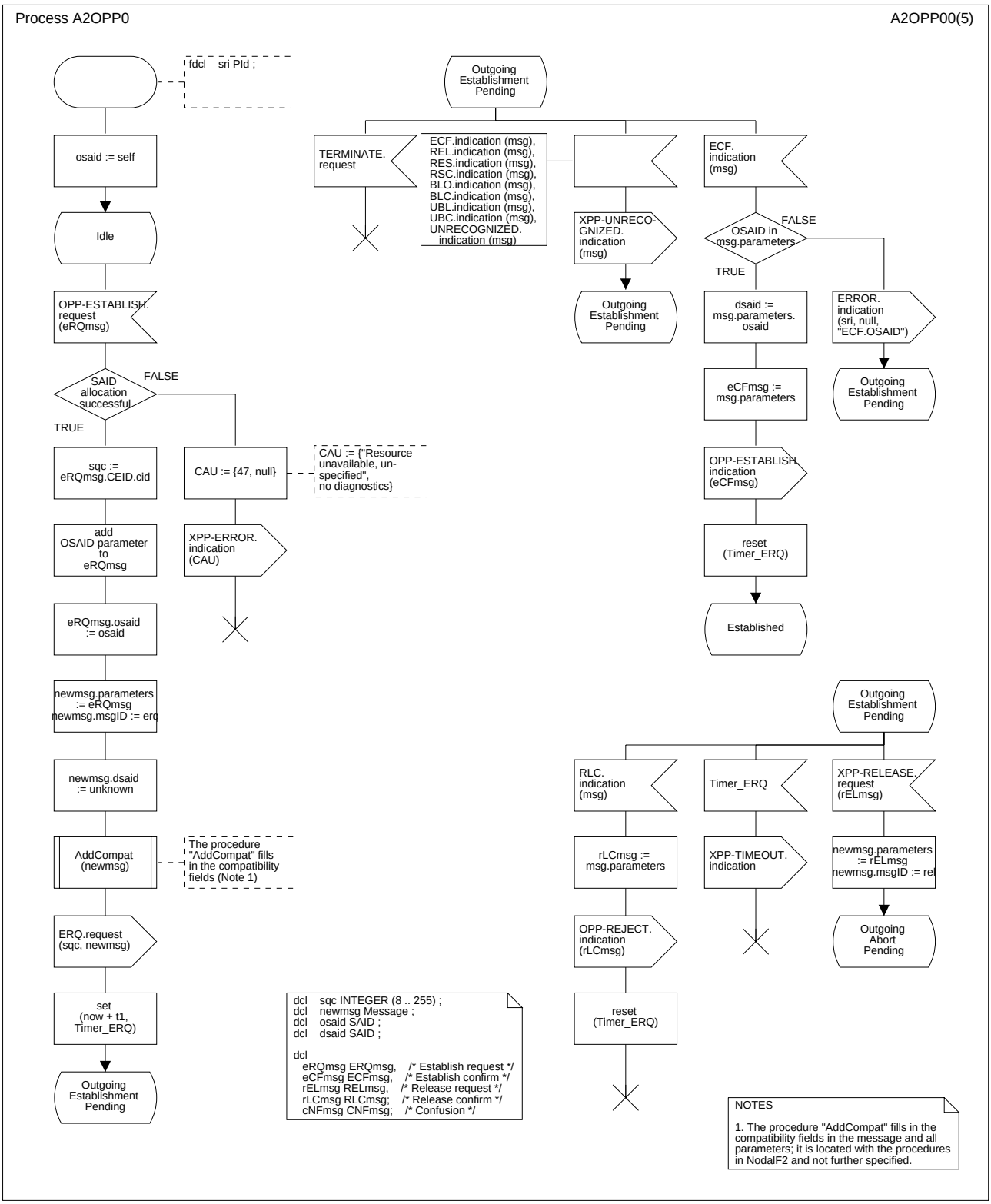


Figure B.4/Q.2630.1 – SDL diagram of the outgoing protocol procedure *(part 1 of 5)*

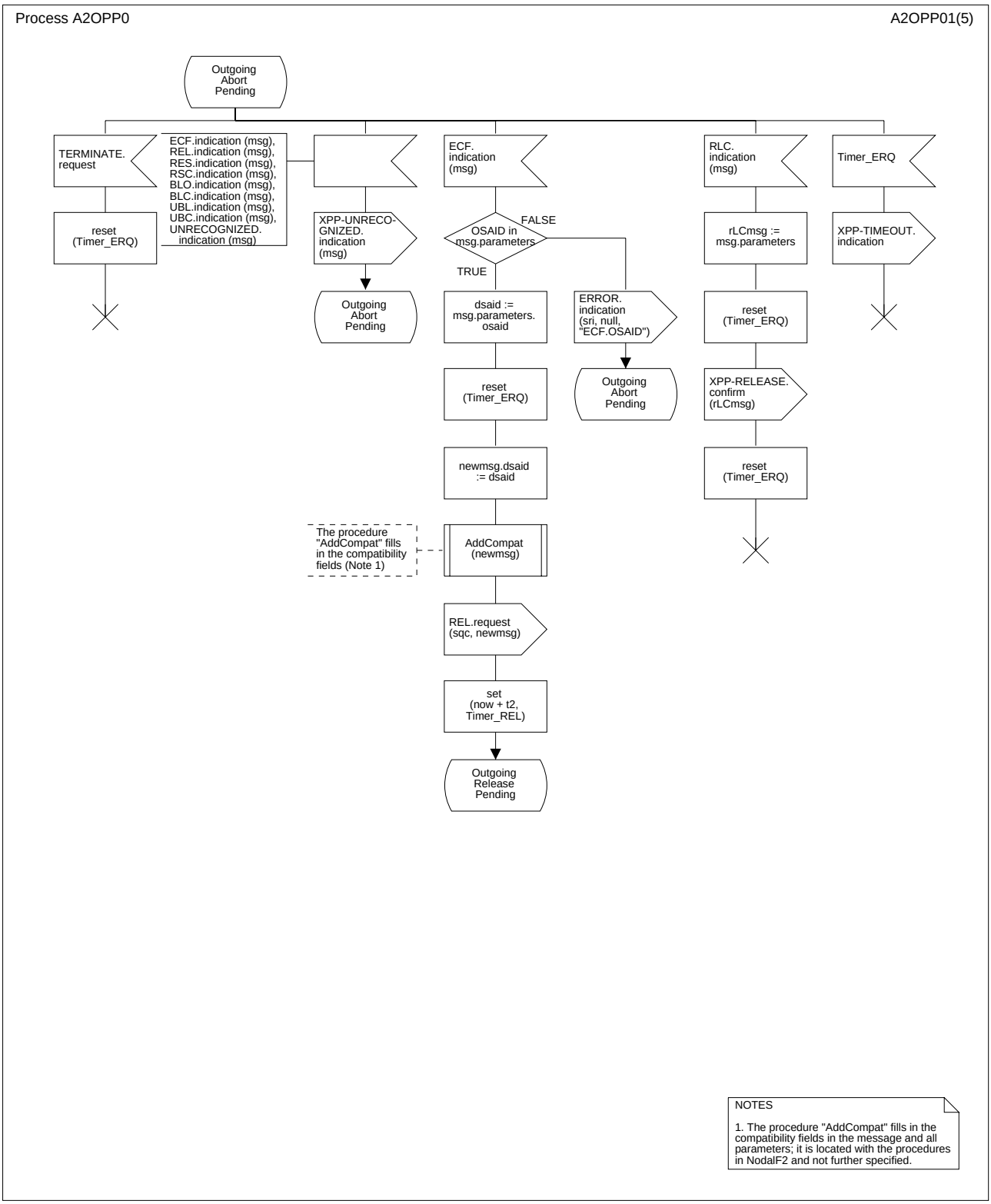


Figure B.4/Q.2630.1 – SDL diagram of the outgoing protocol procedure *(part 2 of 5)*

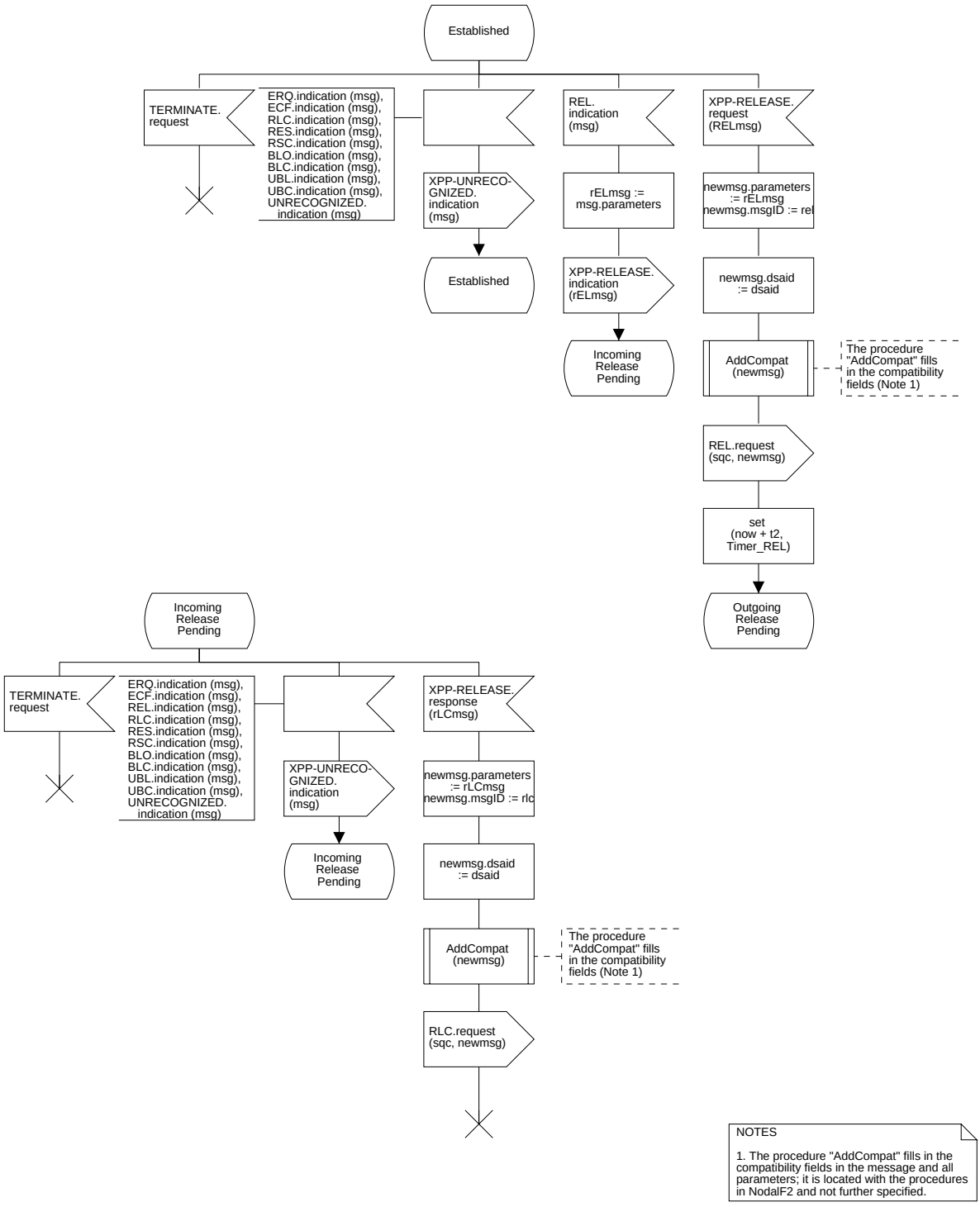


Figure B.4/Q.2630.1 – SDL diagram of the outgoing protocol procedure *(part 3 of 5)*

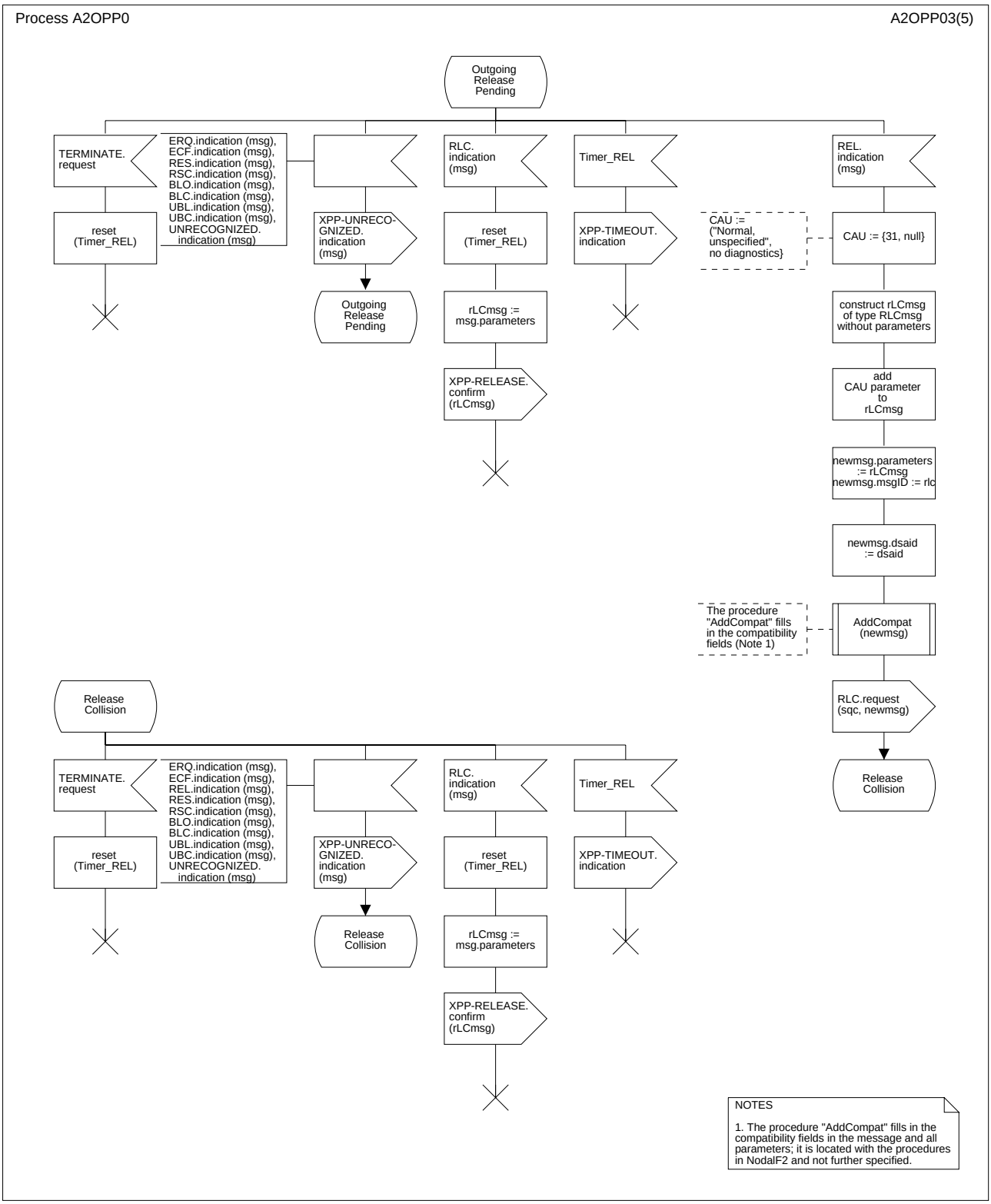


Figure B.4/Q.2630.1 – SDL diagram of the outgoing protocol procedure *(part 4 of 5)*

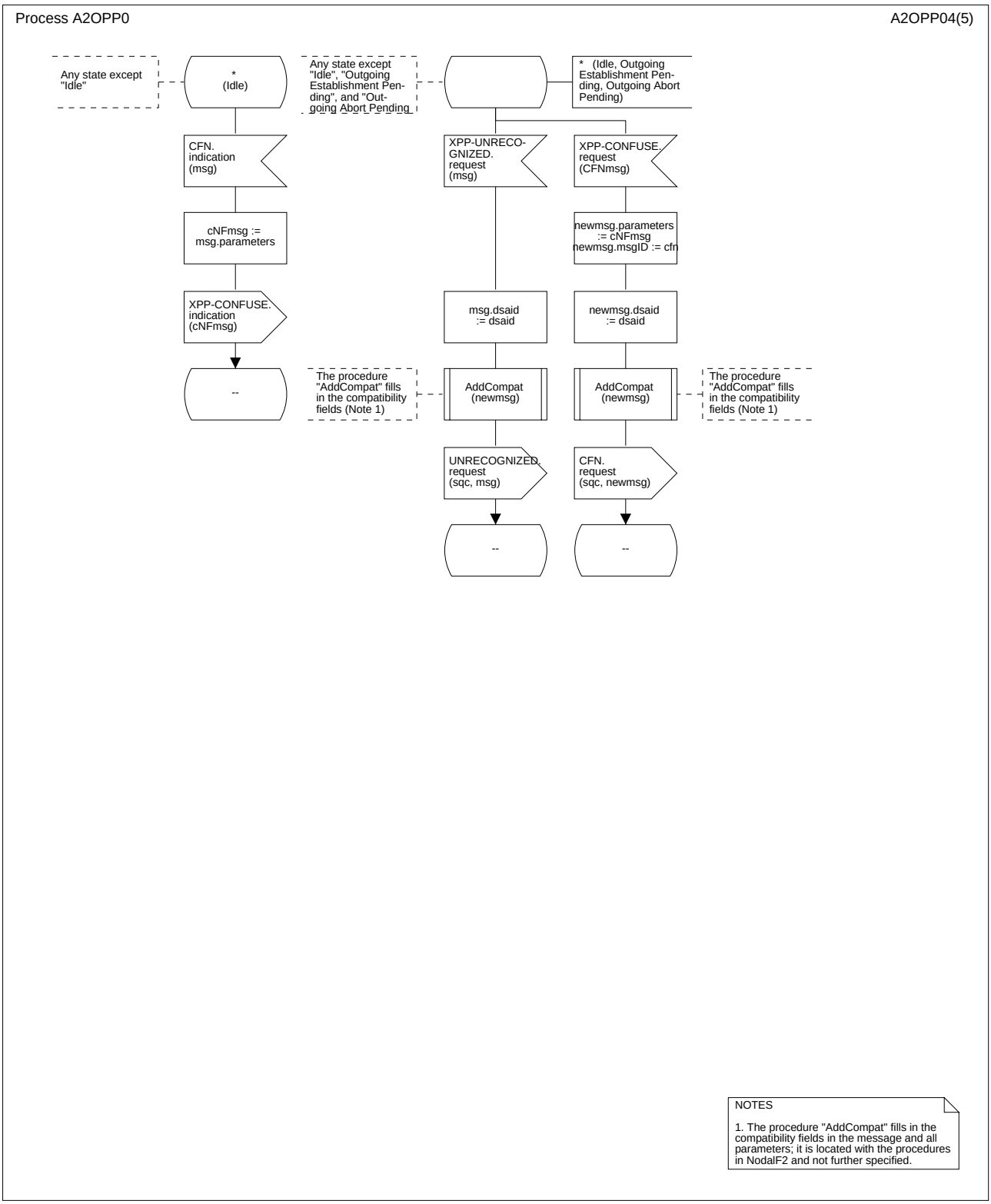


Figure B.4/Q.2630.1 – SDL diagram of the outgoing protocol procedure *(part 5 of 5)*

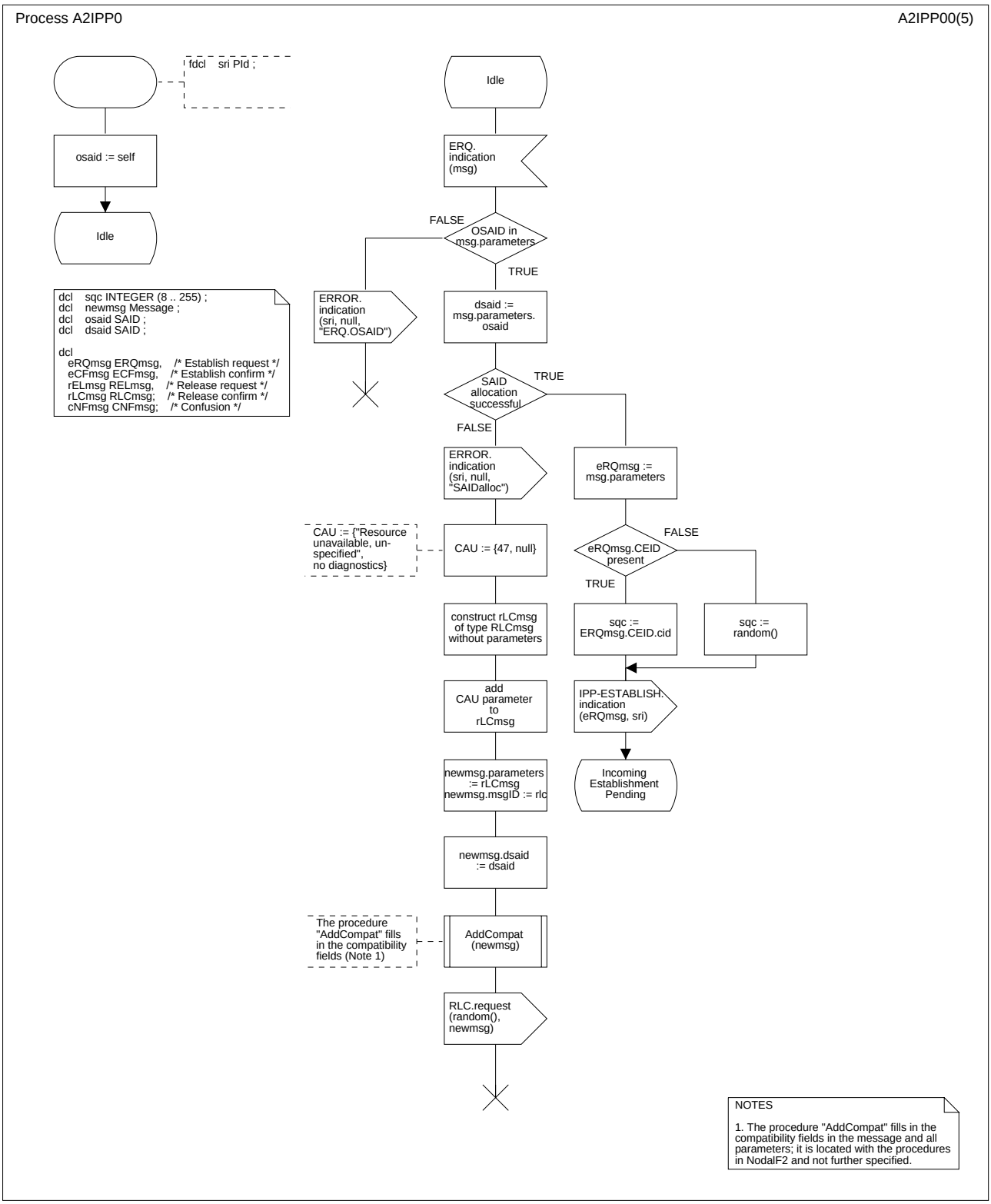


Figure B.5/Q.2630.1 – SDL diagram of the incoming protocol procedure *(part 1 of 5)*

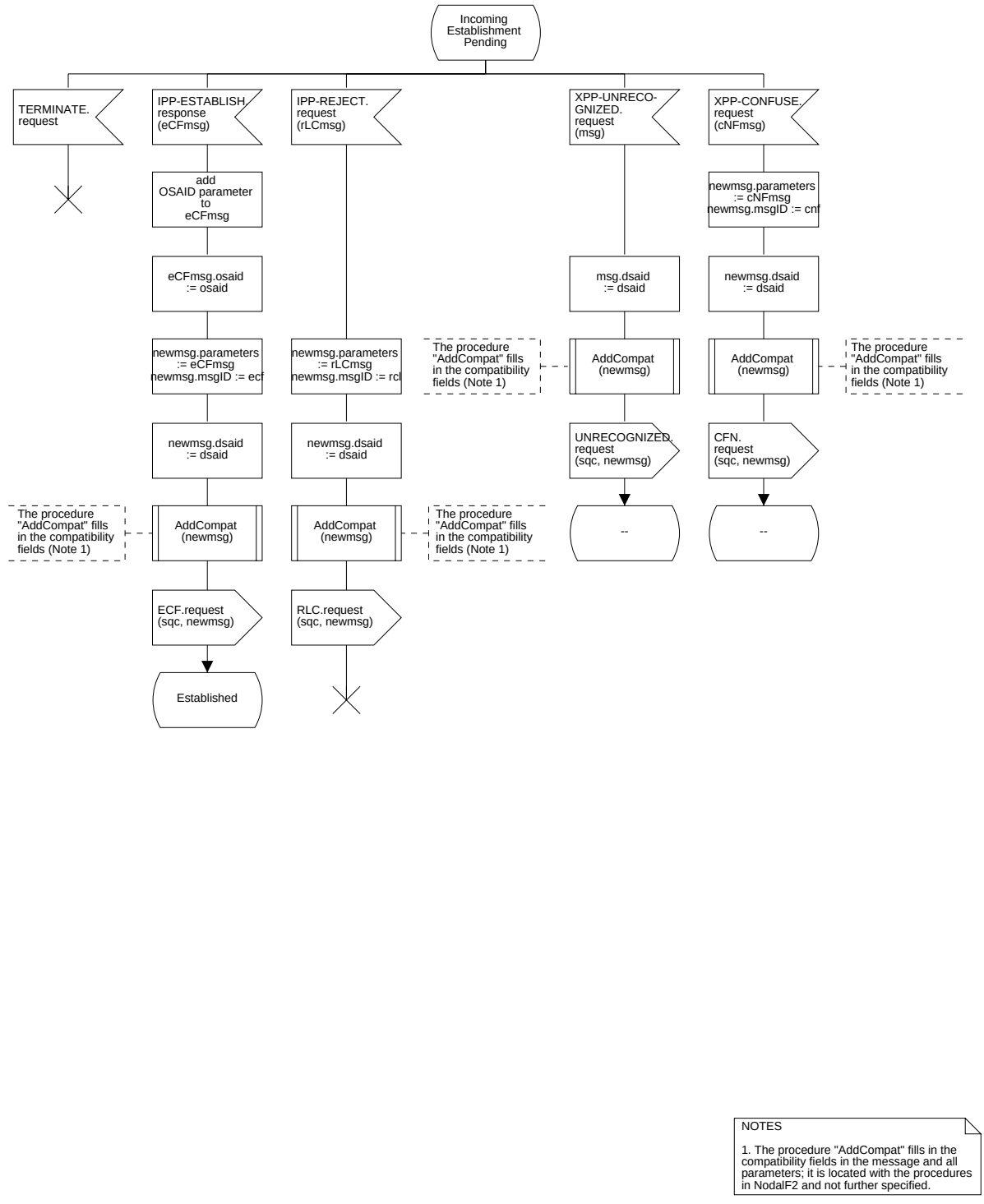


Figure B.5/Q.2630.1 – SDL diagram of the incoming protocol procedure *(part 2 of 5)*

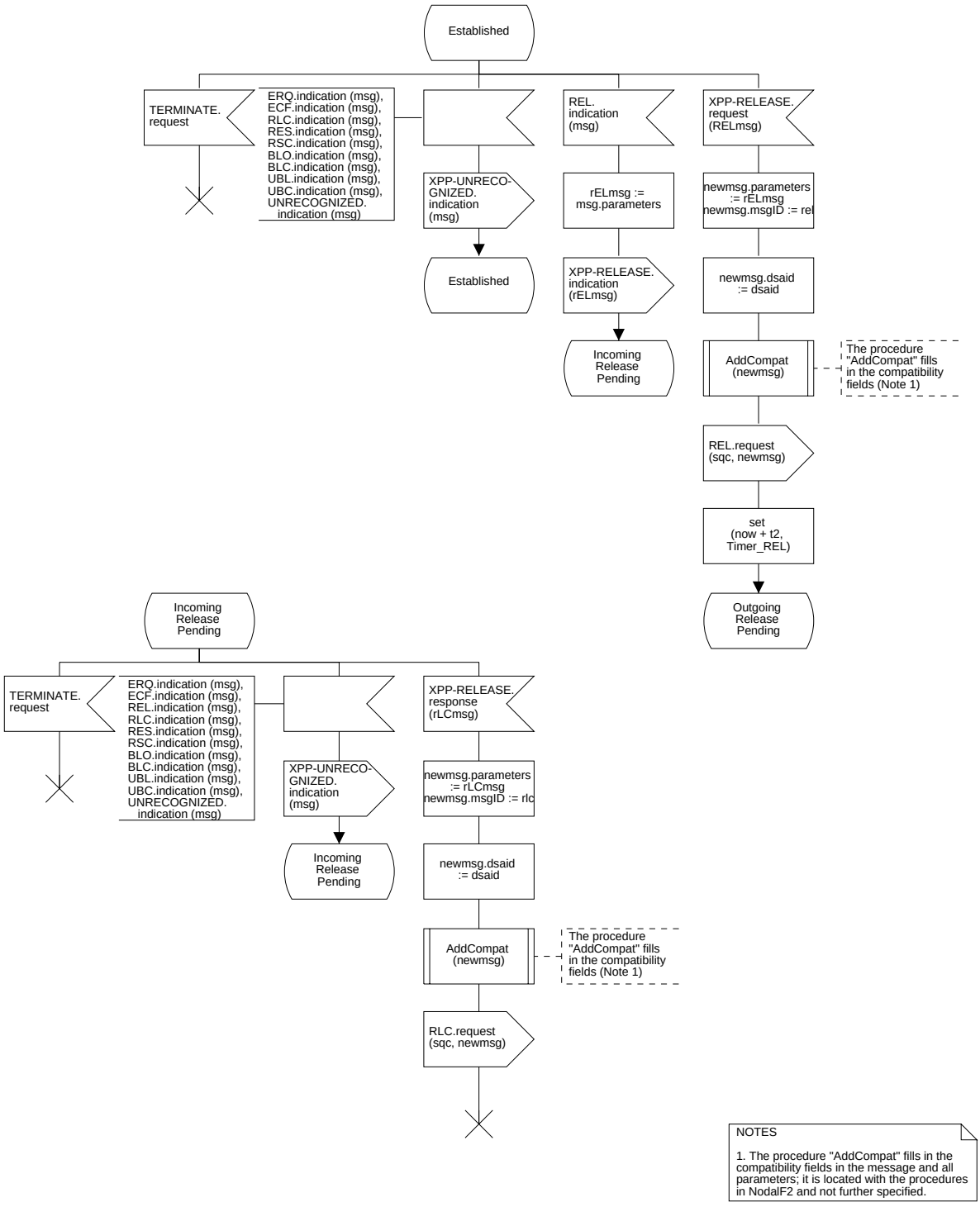


Figure B.5/Q.2630.1 – SDL diagram of the incoming protocol procedure *(part 3 of 5)*

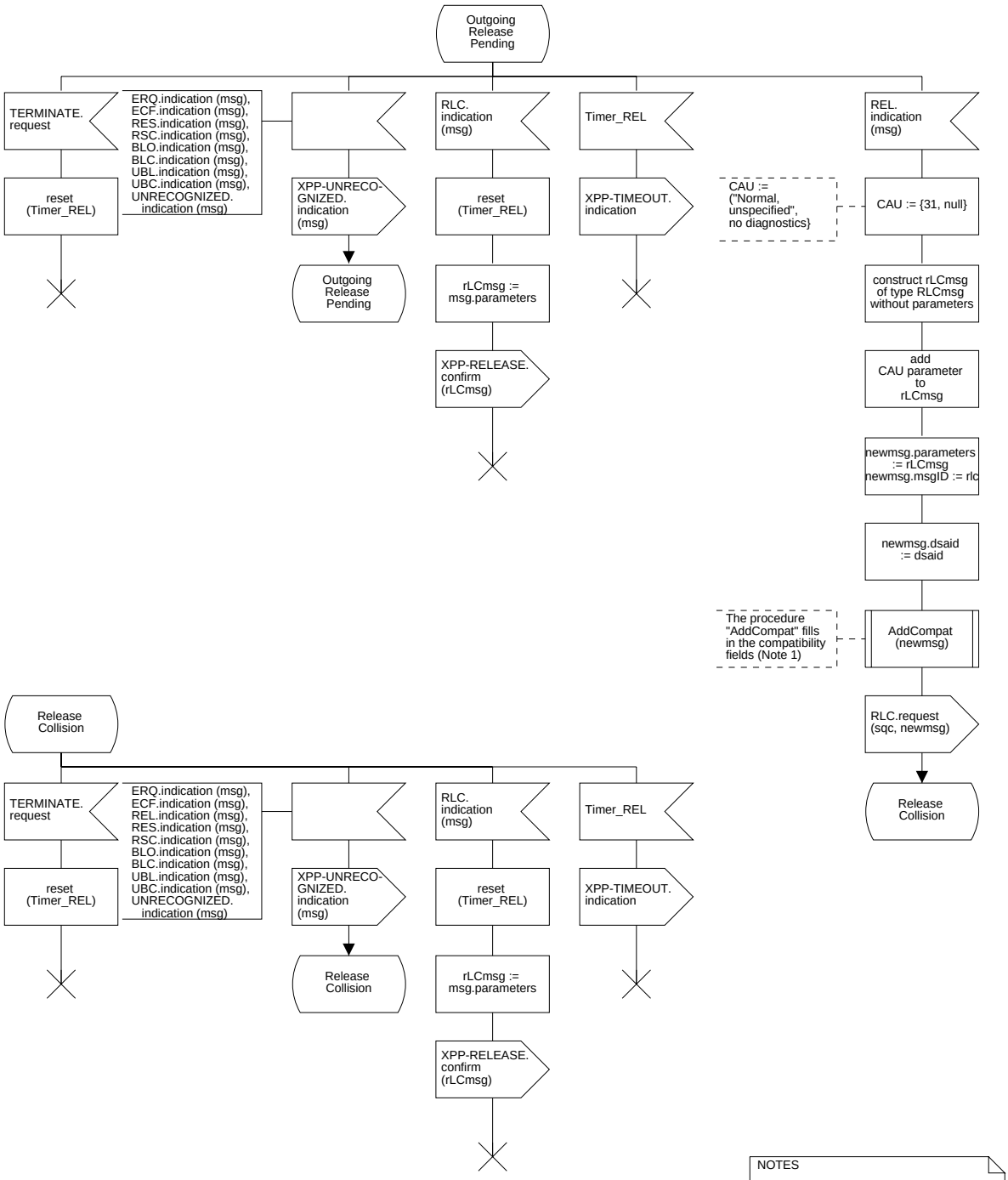


Figure B.5/Q.2630.1 – SDL diagram of the incoming protocol procedure *(part 4 of 5)*

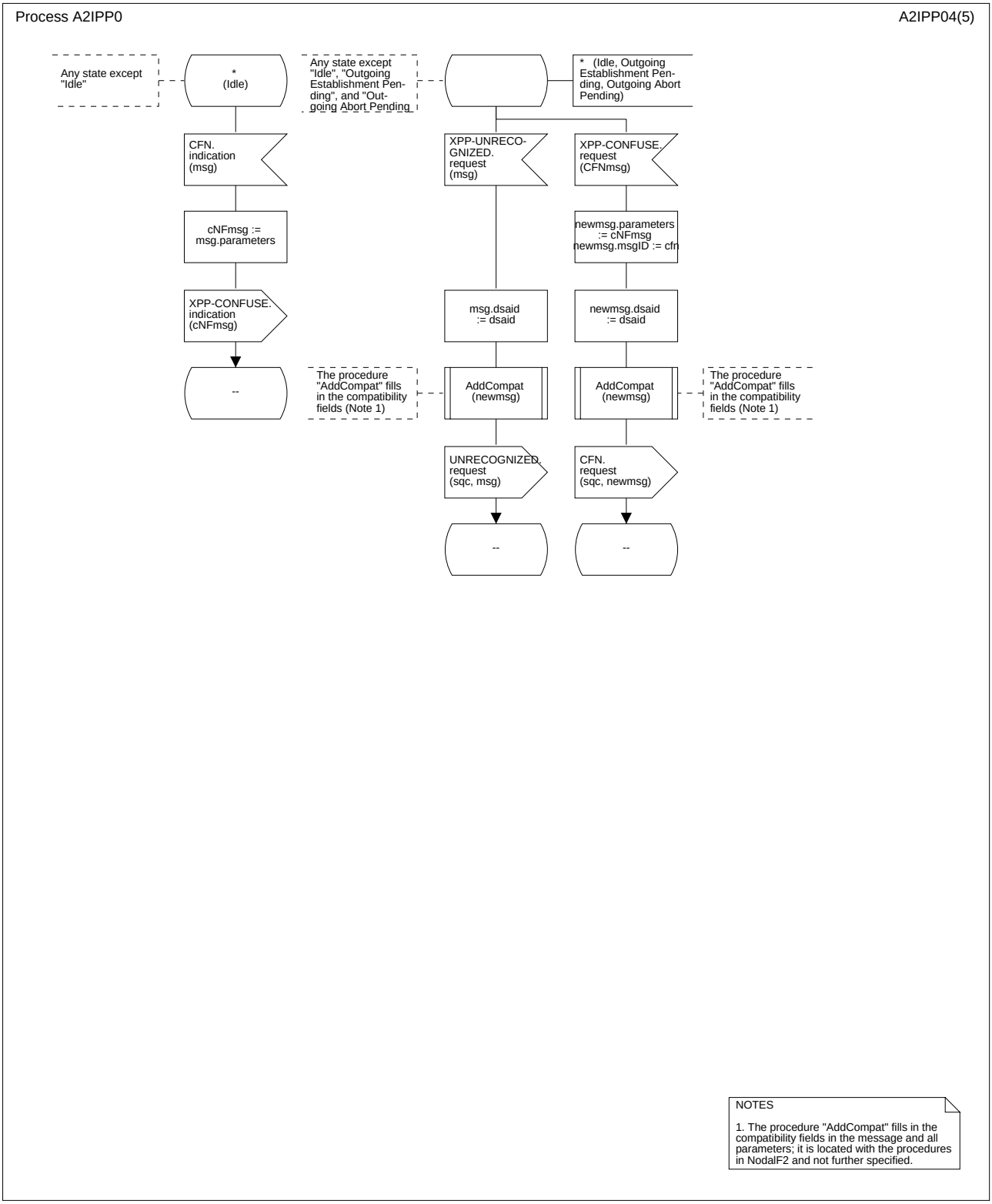


Figure B.5/Q.2630.1 – SDL diagram of the incoming protocol procedure *(part 5 of 5)*

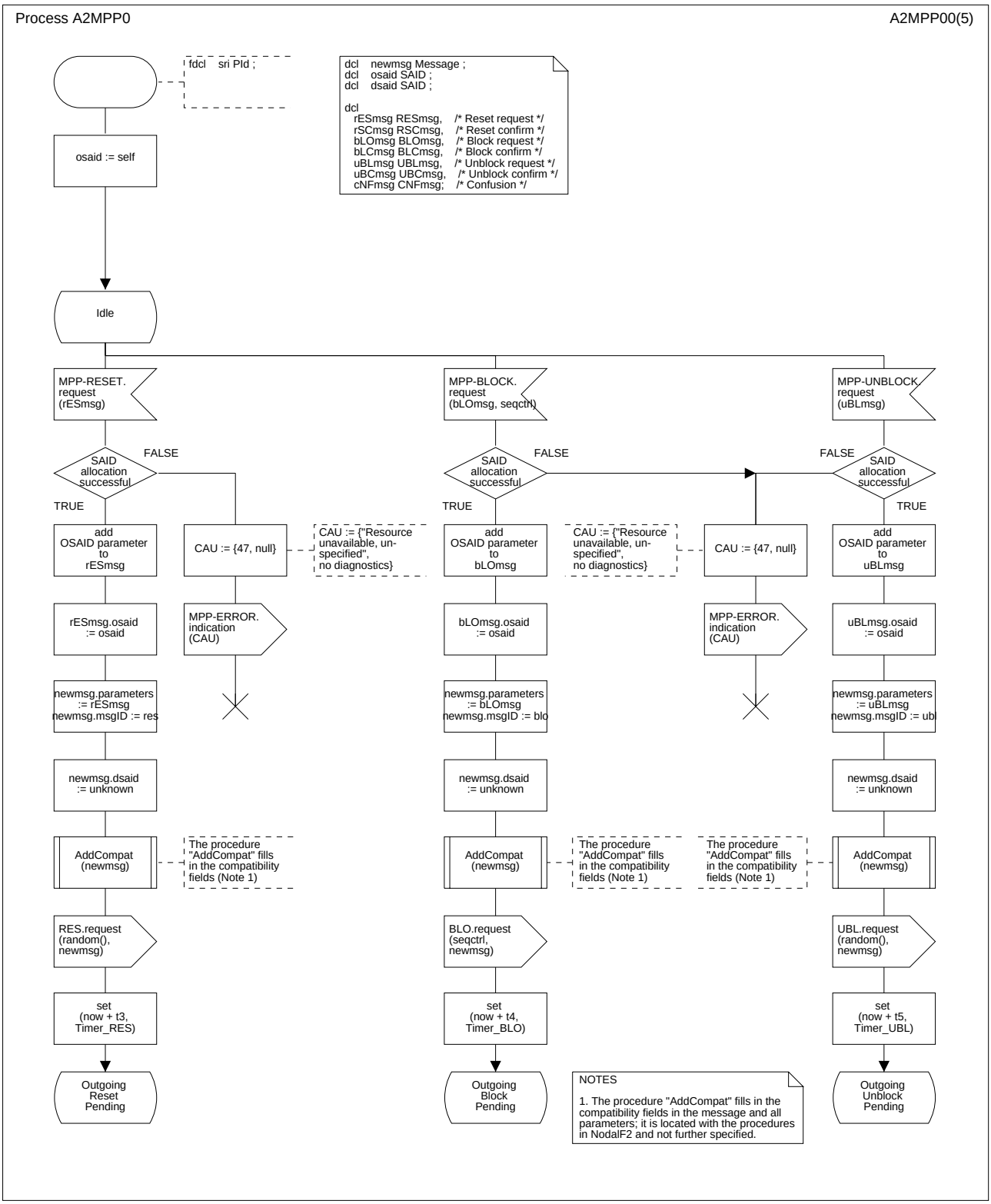


Figure B.6/Q.2630.1 – SDL diagram of the maintenance protocol procedure *(part 1 of 5)*

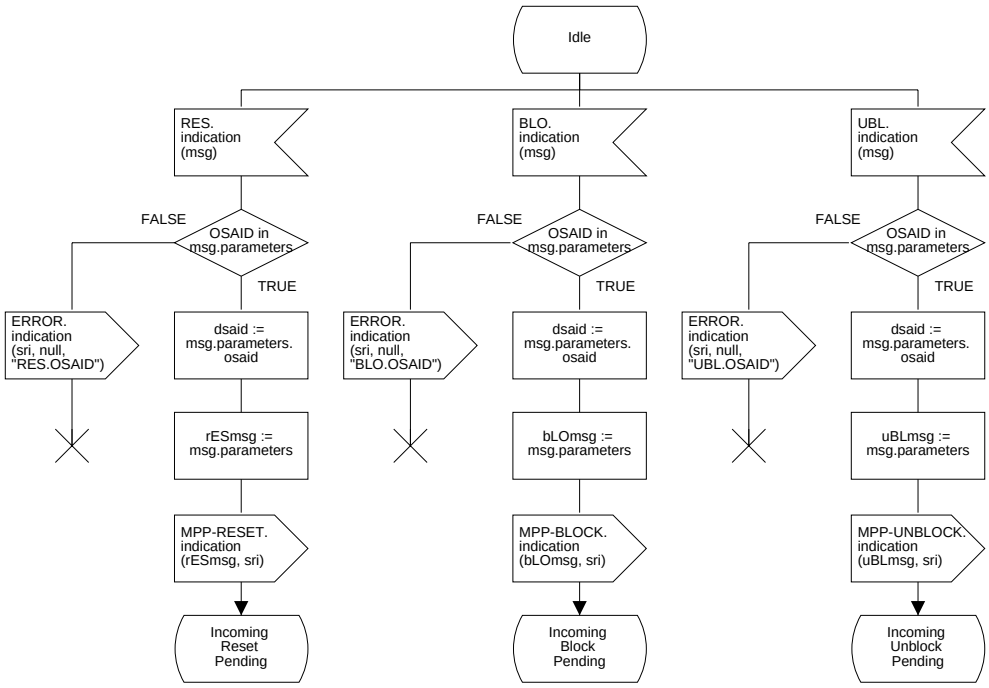


Figure B.6/Q.2630.1 – SDL diagram of the maintenance protocol procedure *(part 2 of 5)*

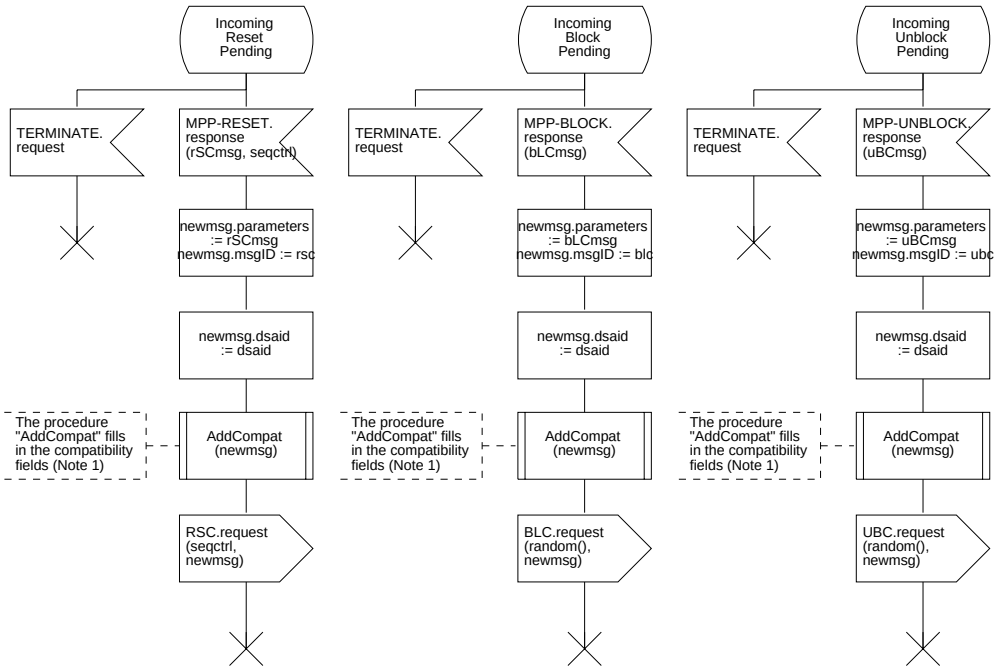


Figure B.6/Q.2630.1 – SDL diagram of the maintenance protocol procedure *(part 3 of 5)*

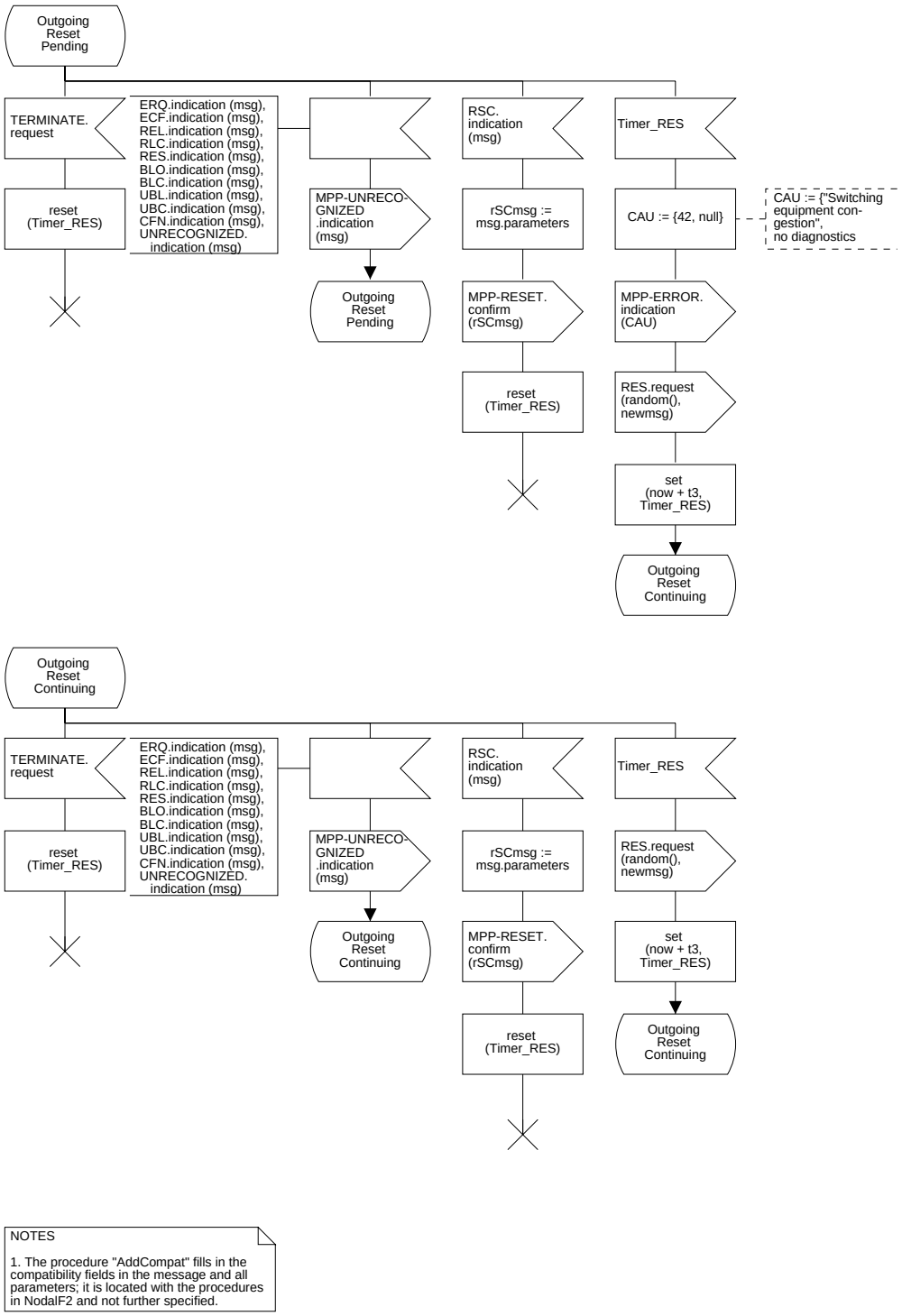


Figure B.6/Q.2630.1 – SDL diagram of the maintenance protocol procedure *(part 4 of 5)*

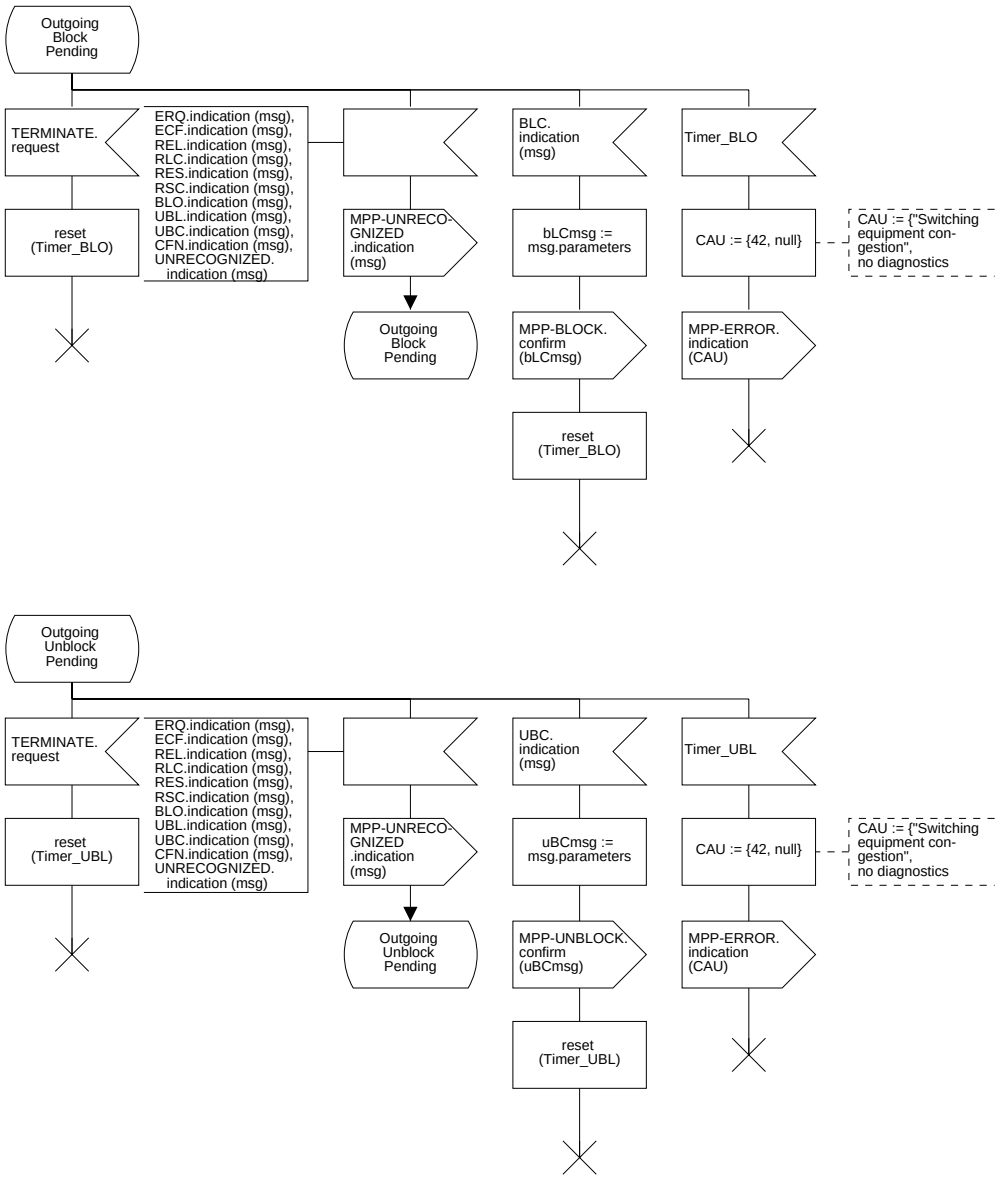


Figure B.6/Q.2630.1 – SDL diagram of the maintenance protocol procedure *(part 5 of 5)*

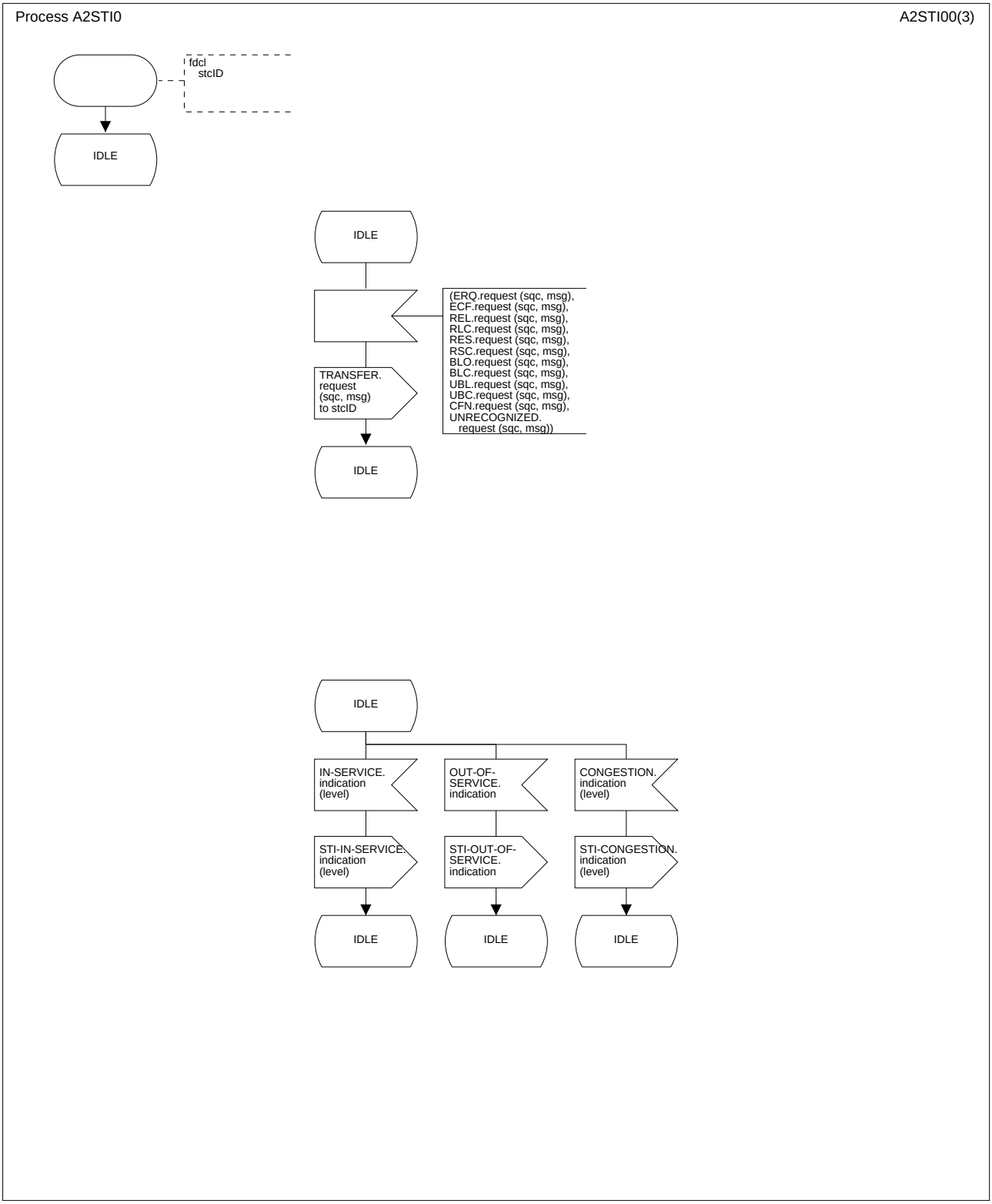


Figure B.7/Q.2630.1 – SDL diagram of the signalling transport interface *(part 1 of 2)*

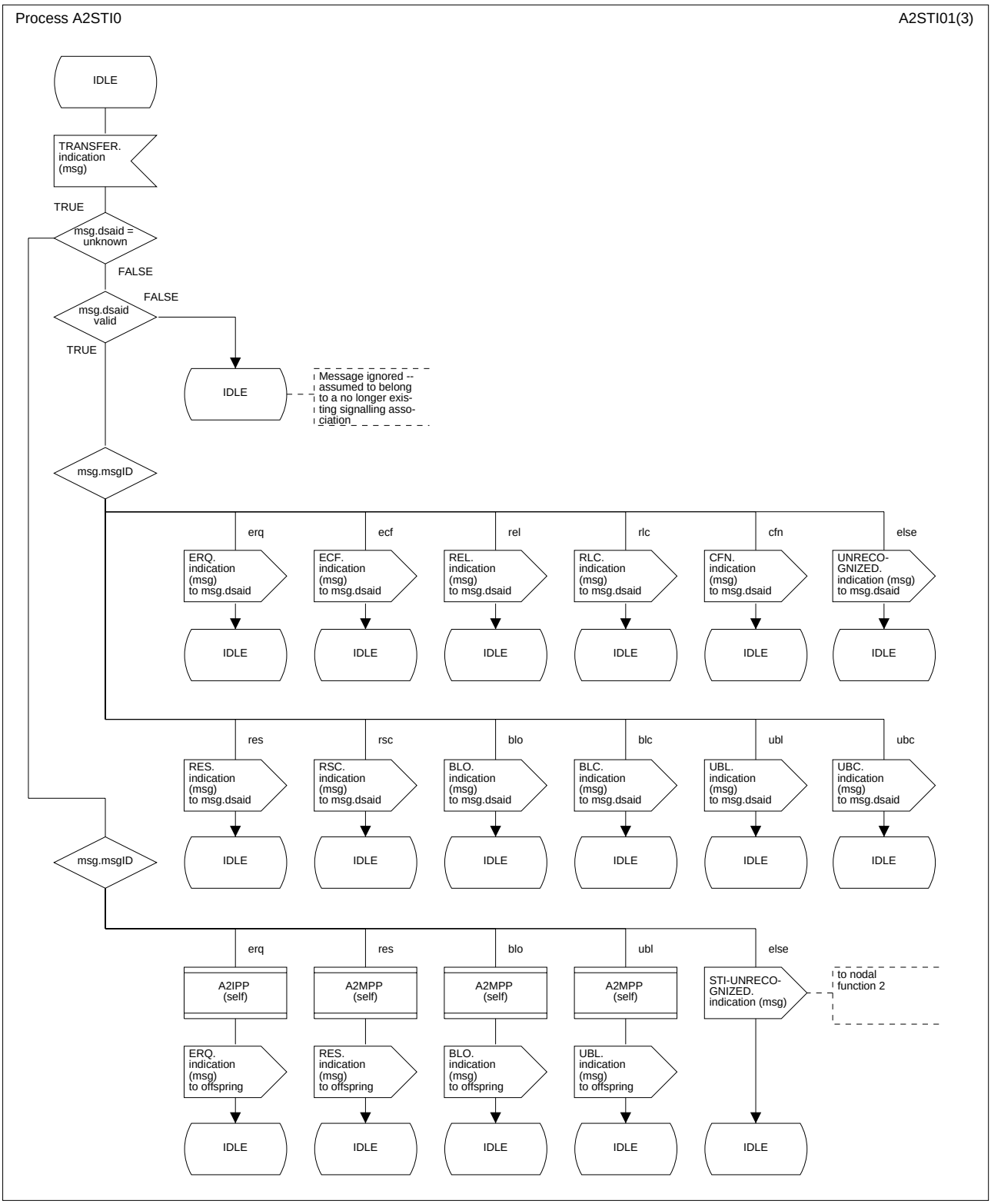


Figure B.7/Q.2630.1 – SDL diagram of the signalling transport interface *(part 2 of 2)*

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

20867

Printed in Switzerland
Geneva, 2001