TSG-RAN Working Group 2 (Radio layer 2 and Radio layer 3) **TSGR2#2(99)081**
Stockholm 8th to 11th March 1999

Agenda Item: 6.1

Source: Nokia

Title: Descriptive SDL and its use in 3GPP standards

Document for: Decision

1. Introduction

SDL included in standards has generally fallen into two categories. The first one includes SDL code which is complex and accurate, but unreadable. This is undesirable, because implementations depend on human understanding of the SDL code even if automatic tools are also available. The second category is readable but syntactically incorrect and/or ambiguous SDL, which cannot be used as a basis of implementations. SDL in protocol standards should be both validatable by automatic tools and easy to read by human beings. However, these two requirements can be mutually exclusive.

ETSI MTS (Methods for Testing and Specification) is currently developing a set of guidelines [1] which, if followed by rapporteurs, would help them to produce SDL that:

- is easy to read
- is syntactically correct
- has correct static semantics
- expresses requirements unambiguously
- can be transformed into a validation model by an expert with the minimum of effort.

In this contribution the main ideas of these guidelines are described, some of them using examples. The guideline document itself is presented in WG2 contribution R2-99082 [2].

2. Descriptive SDL

2.1. Basic principles of descriptive SDL

The basic idea behind descriptive SDL is to be able to produce readable and syntactically correct SDL code. The descriptive SDL guidelines are not intended to be an SDL tutorial – a basic knowledge of SDL is still required. On the other hand, code produced according to the guidelines should be readable even to persons who are not SDL experts.

Descriptive SDL concentrates on the **what** rather than the **how**. Everything the target protocol does must be described, and the different actions must be in the correct order. However, the way in which the details of these actions are coded is not important in descriptive SDL, and indeed often worsens the readability of the code.

To be able to test the syntax and static semantics of the SDL code it would be best to produce descriptive SDL with a proper SDL tool. In addition, this makes it easier to develop the code further into a (part of a) validation model for more thorough testing. However, a specific SDL tool is not mandatory, since descriptive SDL can very well be produced with normal drawing tools as well.

2.2. Structure of the guideline document

The guideline document is grouped into eight categories, and examples of the guidelines in each category are briefly presented below. In the document itself the guidelines are written in ***bold italic***, and they are grouped together at the end of the document for easy reference.

2.2.1. Naming conventions

It is not possible to impose a strict rule on the length of identifiers used in SDL, but as a general guideline names between 6 and 30 characters in length should be preferred over shorter or longer ones. Underscores should be used to separate different parts of a long name, e.g. Deliver_Message. Names should also be meaningful and unambiguous – for instance, a signal named Id_Req could be interpreted as Identity_Request or equally well as Identifier_Required. Process names should be noun phrases which indicate the general function of the process, while procedure names should be verb phrases indicating the specific activity to be carried out by the procedure.

2.2.2. Presentation and layout of diagrams

The control flow of SDL process diagrams should be from the top of the page to the bottom. This enhances the readability of the code greatly when compared to the zig-zag style sometimes used. Also, the symbols that terminate the processing on a particular page should be aligned horizontally. This way it is easy to see at one glance all the possible points where transitions terminate. Also, if a symbol is not large enough to contain all the text necessary to specify the task represented by the symbol, a text extension symbol should be used to carry the additional information.

2.2.3. Structuring

A state, input and the associated transition to the next state should be contained within a single SDL page. Transitions from the next state should not be presented on the same page. In some cases the transitions are so complex that connectors must be used, but in general connectors should be used as little as possible.

2.2.4. The use of procedures and operators

To hide the potentially distracting details of e.g. variable handling and input analysis procedures and operators should be used. However, input and output operations should not be included in procedures. For specification purposes it is often enough to name the procedure in a descriptive manner, include the procedure call in the process code and leave the procedure itself unspecified. The procedure can be specified in more detail when constructing a validation model of the protocol. Macros, however, should be avoided in other cases than timer handling.

2.2.5. The use of decisions

Informal text can be used in decisions, but only in cases where the decision is obviously boolean in nature. A better solution is to use enumerated types. Also, the complete range of possible values must be covered.

2.2.6. System structure, communication and addressing

When coding protocols with SDL complete systems should be specified instead of just specifying the process(es) corresponding to the target protocol(s). This way the interfaces between different parts of the system can be specified. Whenever possible, a minimal number of static process instances should be used instead of dynamically created SDL processes. If the target process of a signal or the signal route the signal can be sent on is ambiguous, either a TO- or a VIA-clause should be used in the output symbol to indicate the recipient clearly.

2.2.7. The specification and use of data

SDL signals should be used to represent normative messages with ASN.1 describing the parameters carried by the messages. Apart from the very simplest of cases one single structured type parameter should be preferred over a list of simple type parameters.

2.2.8. The use of message sequence charts

In the use of MSC charts the same basic principles apply as in the case of SDL code – clear, uncluttered MSC charts should be used. An MSC should concentrate on the description of the message flows and should not be obscured by too many other symbols, such as an extensive amount of conditions.

2.3. Examples

One previous example of using SDL to clarify text standards is an SMG12 contribution [3], which describes a sequence from GSM 03.60 [4]. In the process of converting the textual standards to SDL some cases in which the textual document required clarification were found – textual documents are usually open to some interpretation, which may lead in unresolved issues and implementation-specific solutions in the production phase. However, SDL was used in the contribution mainly as a drawing tool, not a full-fledged specification method. The following list contains some of the most noticeable bad features of the SDL code in [3]. Some of them are syntax errors, others just not-so-good practice.

- One page should contain a state and transition(s) from this state to another (possibly the same) state, but not transitions from the second state. In other words, the control flow on a page should start from a state (or a connector symbol) and end in a state (or a connector symbol) with no nextstate symbols in between.
- The same signal is received in several different signal input symbols representing succesful and unsuccesful cases. The correct way to do this is to use signal parameters to distinguish between different cases.
- Natural language is used in e.g. task symbols. While this can be quite descriptive, it is not the way a formal specification method such as SDL should be used.
- In most cases macros should be replaced with procedures.
- Extensive use of connector symbols makes the control flow difficult to grasp.
- It is possible to reach the decision symbol with the text "Response Expected?" even if the variable Response Expected has not been set. Also, on page 3 of the original code a cause value for a failure is stored, but it is never used.

Annex 1 contains the original SDL code from the SMG12 contribution and a new version of the same code written using the descriptive SDL guidelines. If the new version would have been written from scratch using only the GSM technical specifications as a reference, the code would probably look very much different, but for the purposes of this contribution the method of modifying the existing SDL was the most convenient one. This example shows the importance of having a commonly agreed, syntactically correct way of using SDL.

The example also shows two possible drawbacks of the descriptive SDL guidelines – the increased number of pages used to express a given segment of code and the increased amount of variables. However, having well-structured and readable code should be more important than the number of pages used. The variables are needed e.g. in decisions to make the code more formal – and when the variables are named suitably, they increase readability as well.

2.4. When to use descriptive SDL

Descriptive SDL is perhaps best suited for protocols in which the handling of protocol variables can conveniently be hidden inside procedures. A prime example from the point of view of WG2 is RRC, but the guidelines would be quite suitable for RLC as well. As for other protocols, for example NBAP, RANAP, RNSAP and non-access stratum protocols would benefit from descriptive SDL as well. MAC and especially layer 1 are perhaps a little too fine-grained to take full advantage of descriptive SDL guidelines – if the protocols by nature handle data at high level of detail close to the physical layer, there is little sense in hiding this natural behaviour by layering and structuring the SDL code too extensively. The descriptive SDL guidelines should not be abandoned altogether when coding MAC and layer 1, however. The guidelines on e.g. naming conventions, presentation and layout of diagrams, structuring, use of decisions and system structure, communication and addressing are still valid even in these cases.

3. Proposal

It is proposed that the following paragraph replaces the first paragraph of Chapter 8 in R2.01 V0.0.1:

The groups are encouraged to use SDL where appropriate. The SDL code included in the standards should follow the descriptive SDL guidelines from ETSI TC-MTS (DEG/MTS-00050) as closely as possible.

4. References

[1] ETSI DEG/MTS-00050 V1.5 (Sep-98), Methods for Testing and Specification (MTS); Guidelines for the use of formal SDL as a descriptive tool

[2] 3GPP TSG-RAN WG2, Tdoc TSGR2#2(99)082: Descriptive SDL guidelines. Source: Nokia.

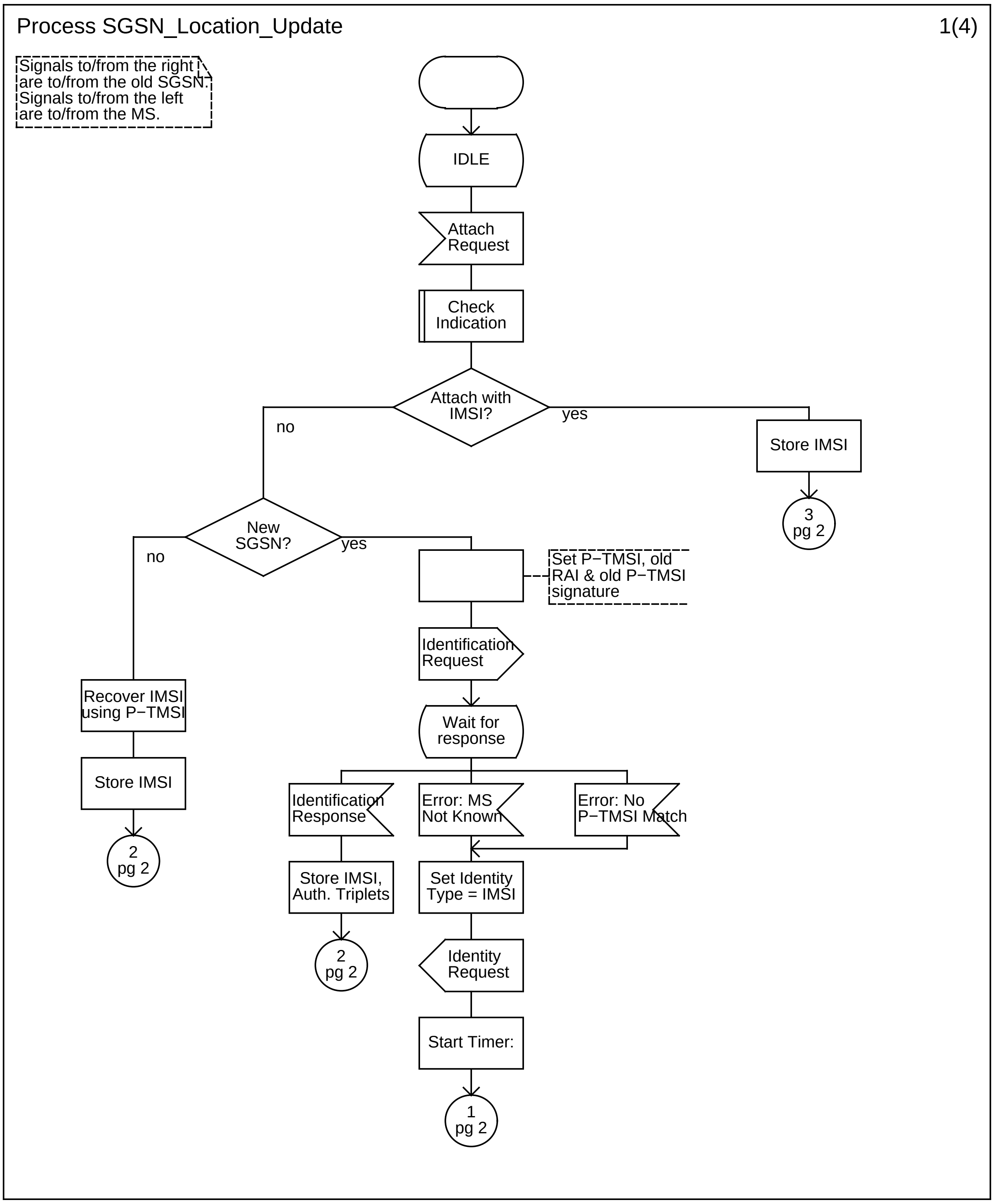
[3] ETSI STC SMG12, Tdoc 98S764: Proposed specification of SDLs for GSM 03.60. Source: Nortel.

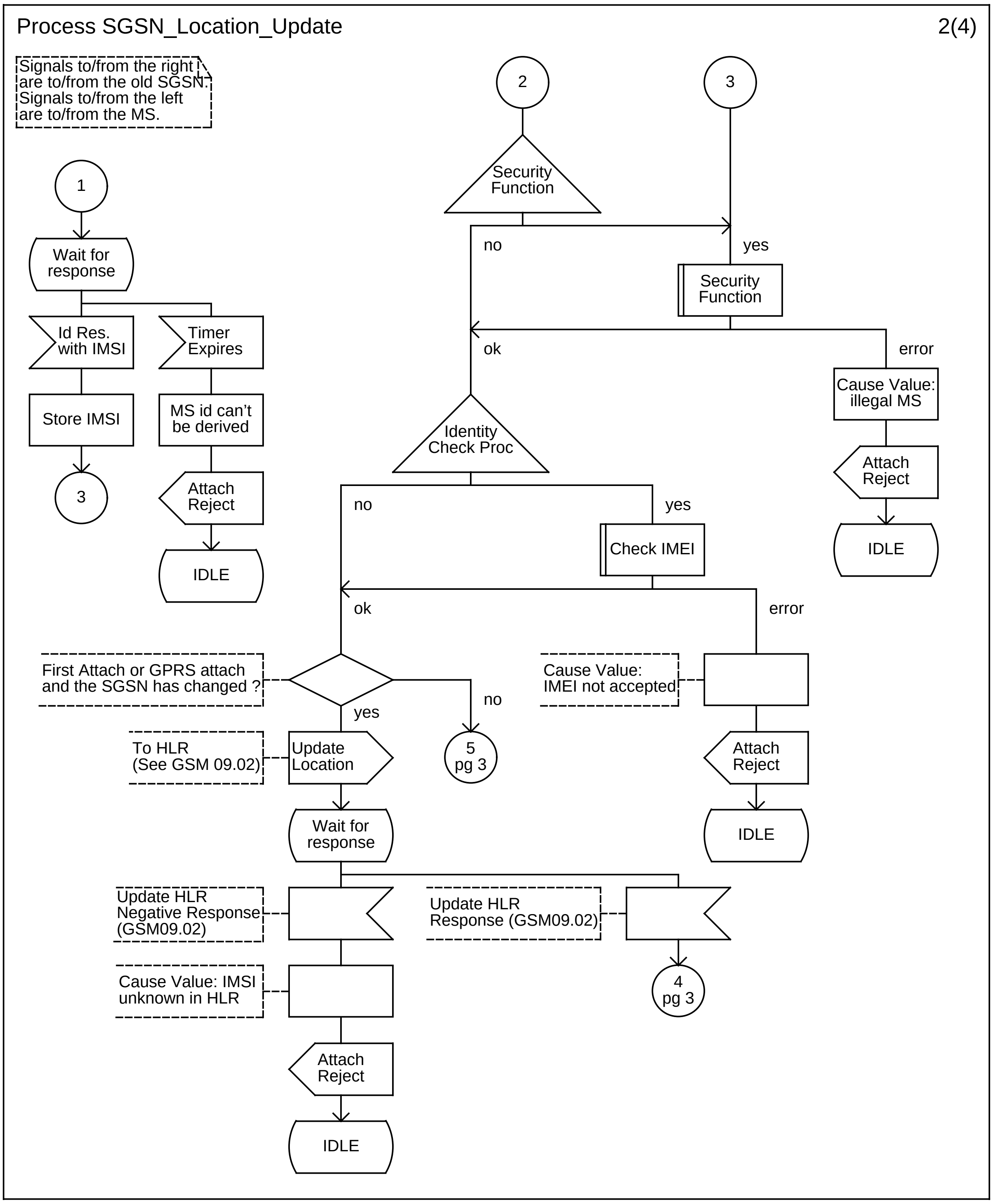
[4] Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Service description; Stage 2 (GSM 03.60 version 6.20 Release 1997)

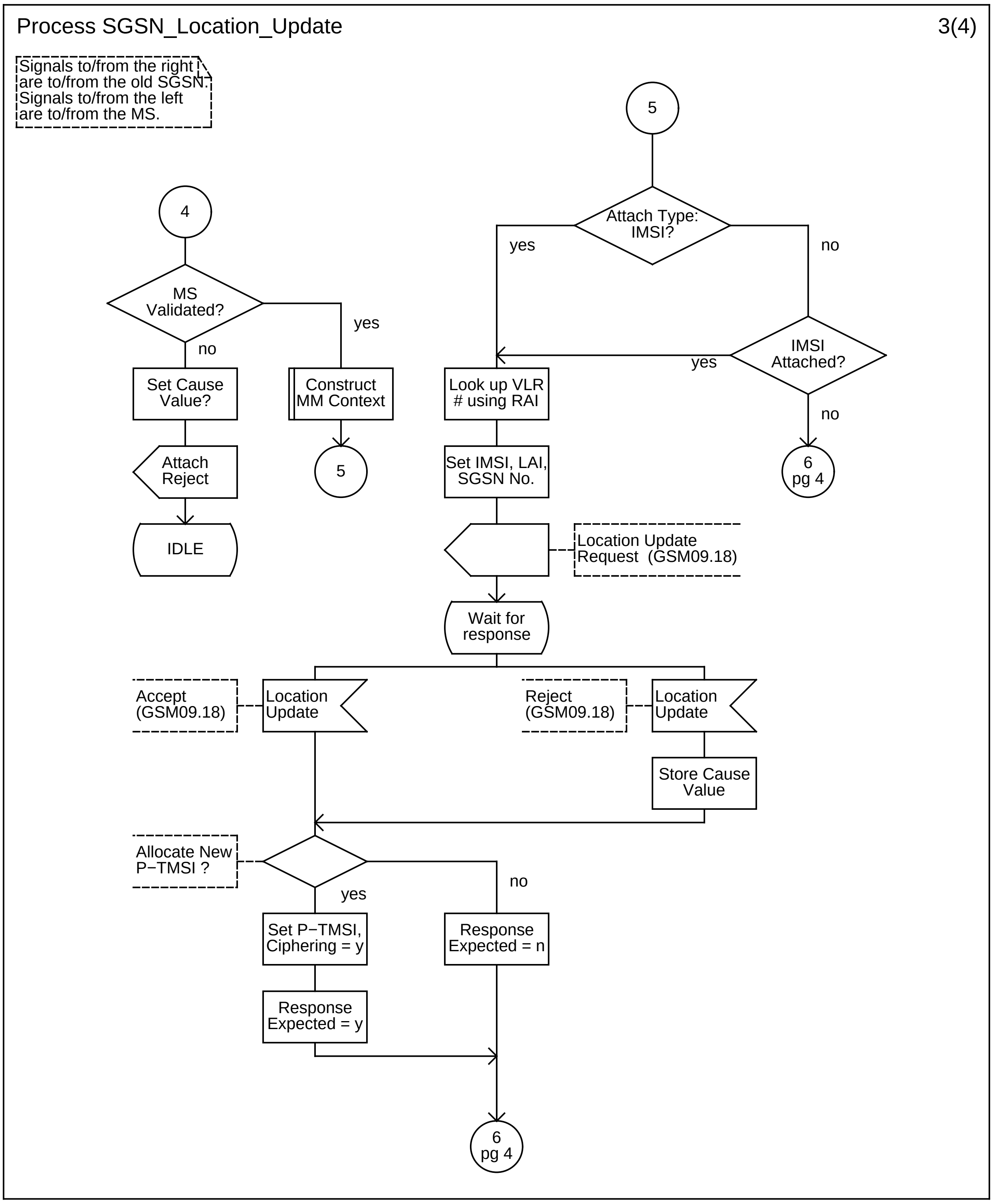
Annex 1: Comparing descriptive SDL code and free-form SDL code

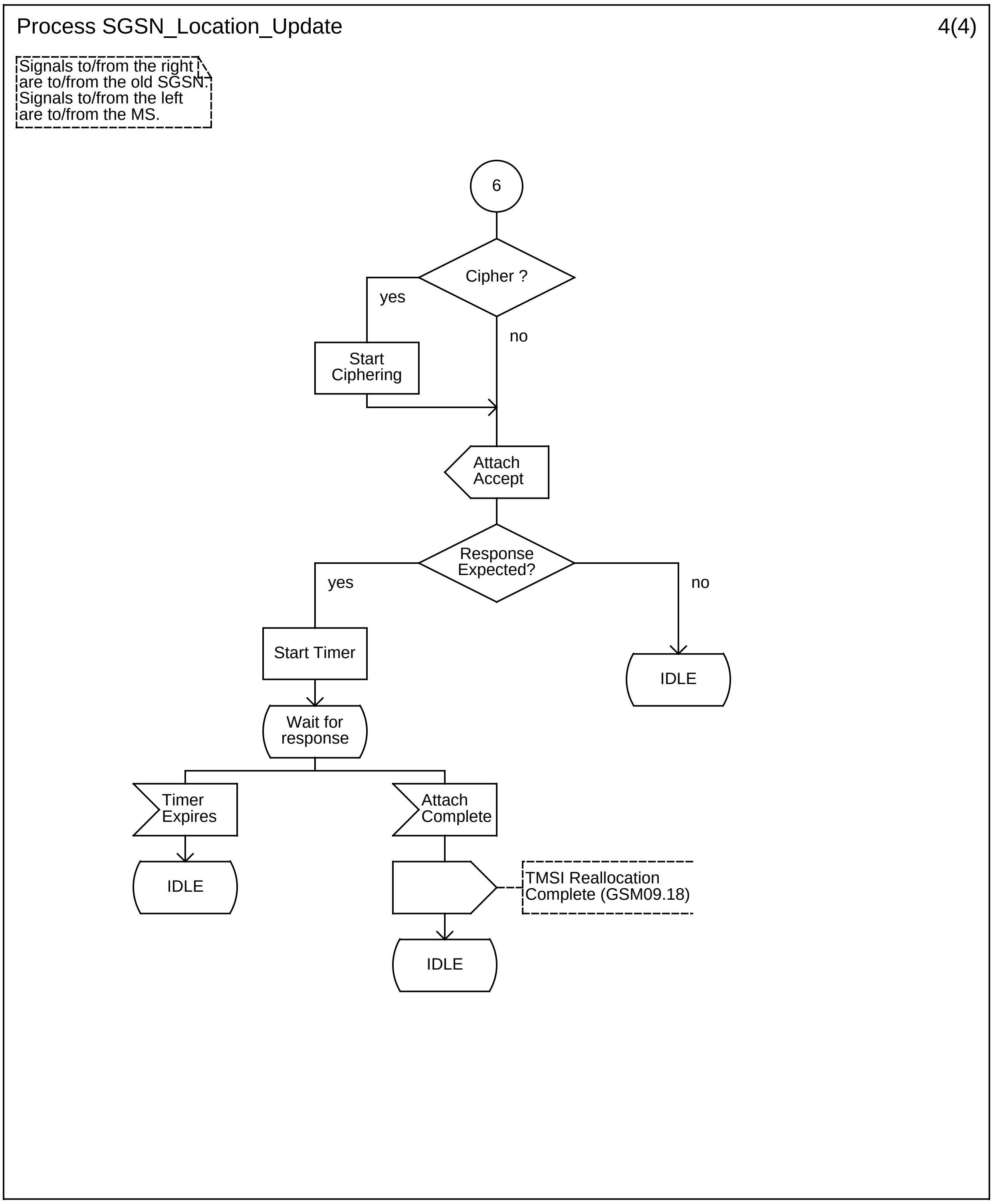
A2.1 Free-form SDL

Unfortunately these SDL diagrams do not have an EPS preview attached to them, so they will not be readable on screen.

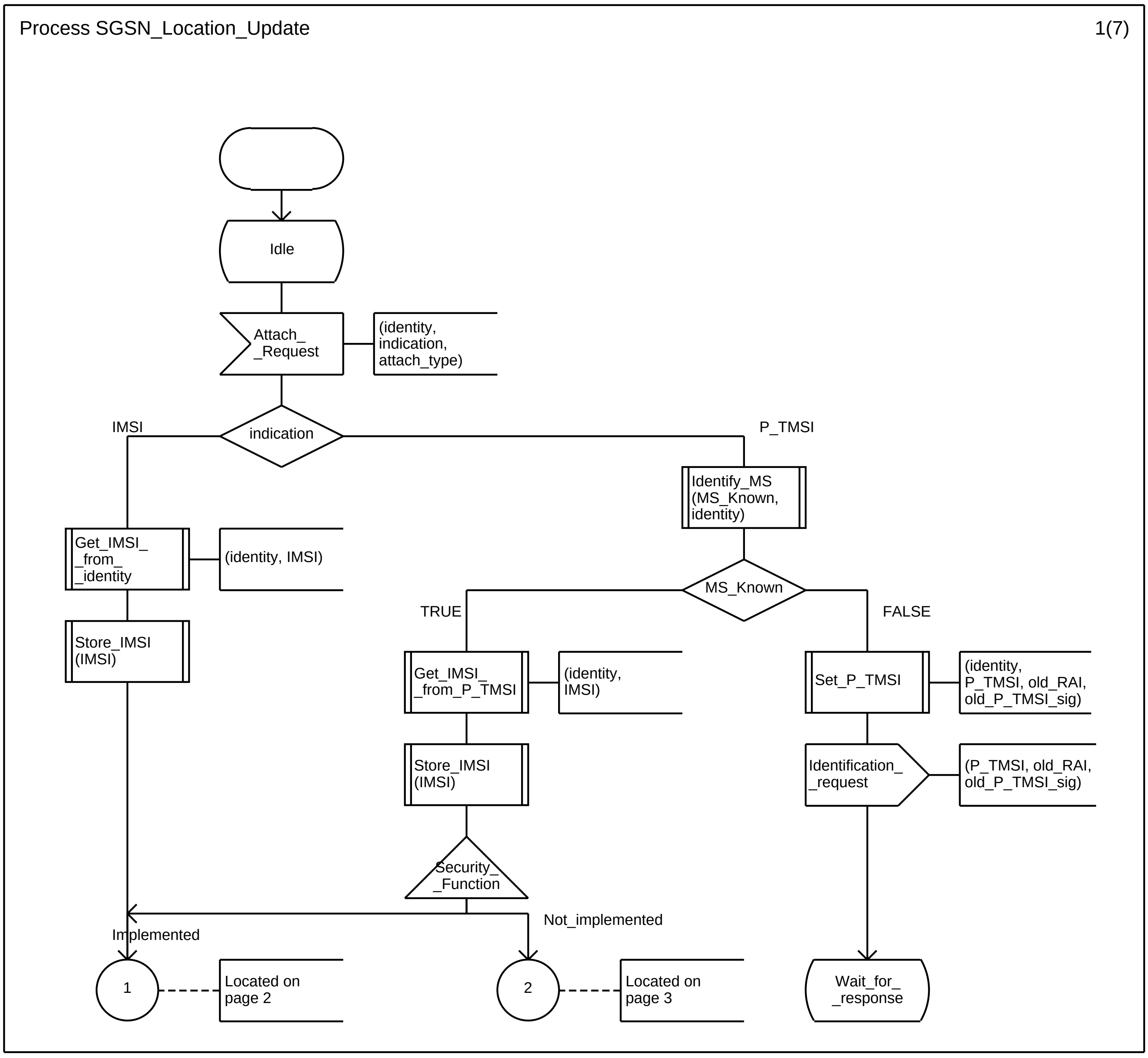


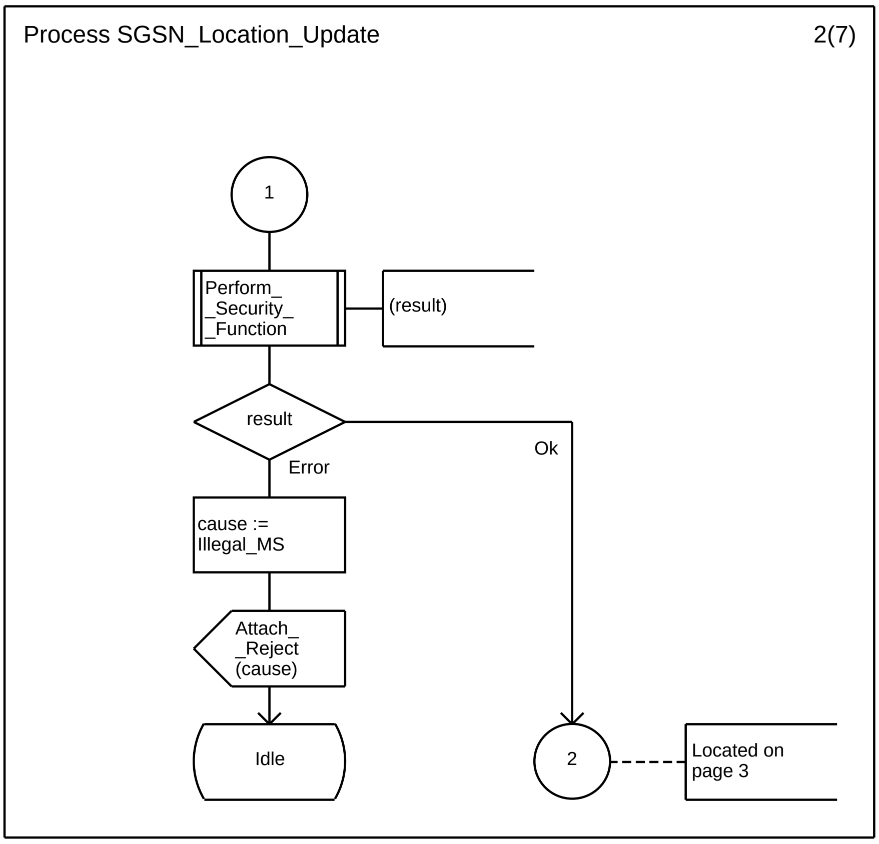

Process SGSN_Location_Update
2(4)
Signals to/from the right are to/from the old SGSN. Signals to/from the left are to/from the MS.
1
Wait for response
Id Res. with IMSI
Timer Expires
Store IMSI
MS id can't be derived
3
Attach Reject
IDLE
2
3
Security Function
no
yes
Security Function
ok
error
Cause Value: illegal MS
Attach Reject
IDLE
Identity Check Proc
no
yes
Check IMEI
ok
error
First Attach or GPRS attach and the SGSN has changed ?
yes
no
Cause Value: IMEI not accepted
To HLR (See GSM 09.02)
Update Location
5 pg 3
Attach Reject
IDLE
Wait for response
Update HLR Negative Response (GSM09.02)
Update HLR Response (GSM09.02)
4 pg 3
Cause Value: IMSI unknown in HLR
Attach Reject
IDLE

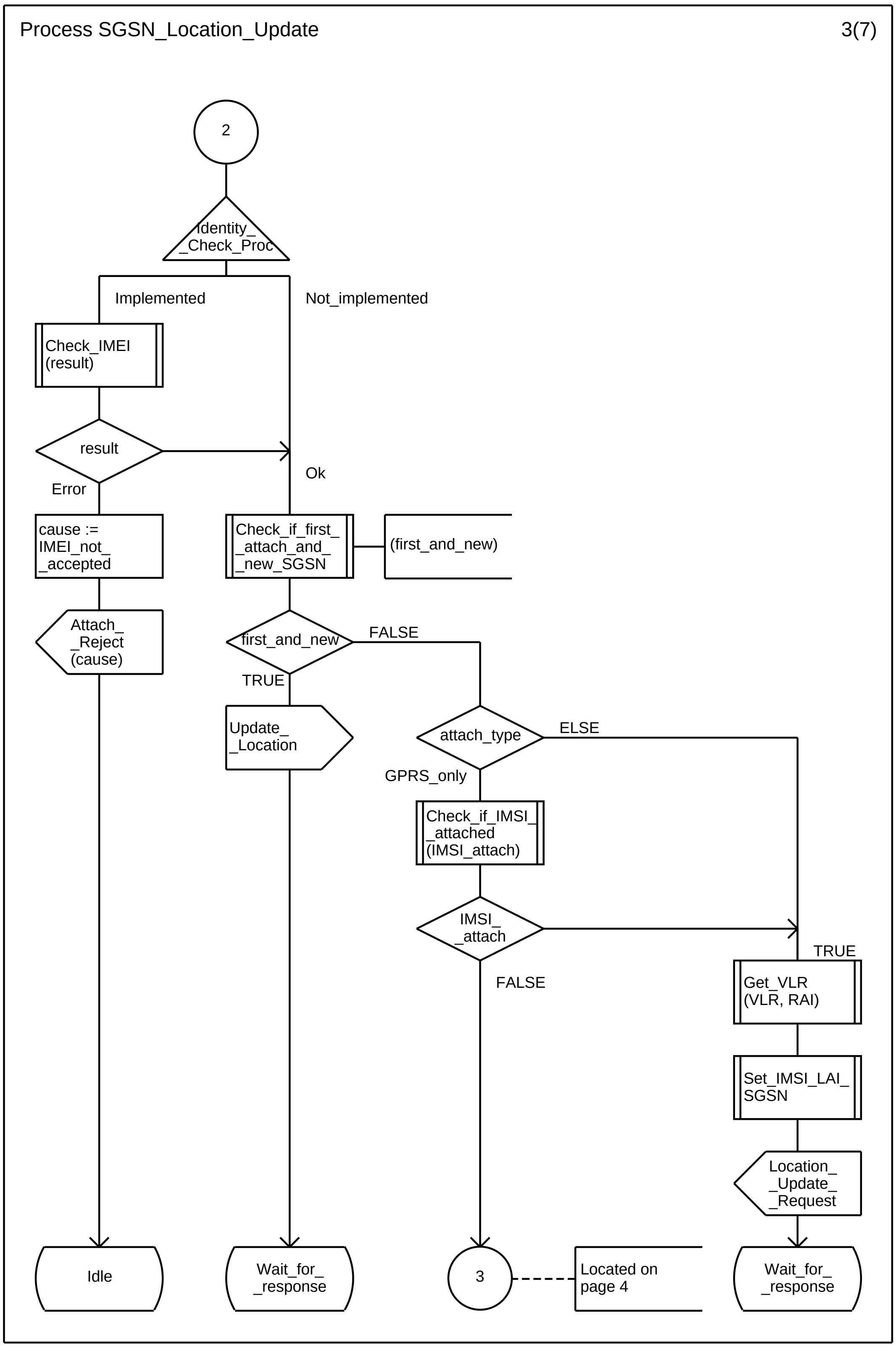

Process SGSN_Location_Update
3(4)
Signals to/from the right are to/from the old SGSN. Signals to/from the left are to/from the MS.
4
MS Validated?
no
yes
Set Cause Value?
Attach Reject
IDLE
Construct MM Context
5
5
Attach Type: IMSI?
yes
no
IMSI Attached?
yes
no
6 pg 4
Look up VLR # using RAI
Set IMSI, LAI, SGSN No.
Location Update Request (GSM09.18)
Wait for response
Accept (GSM09.18)
Location Update
Reject (GSM09.18)
Location Update
Store Cause Value
Allocate New P−TMSI ?
yes
no
Set P−TMSI, Ciphering = y
Response Expected = y
Response Expected = n
6 pg 4


Process SGSN_Location_Update
4(4)
Signals to/from the right are to/from the old SGSN. Signals to/from the left are to/from the MS.
6
Cipher ?
yes
no
Start Ciphering
Attach Accept
Response Expected?
yes
no
Start Timer
IDLE
Wait for response
Timer Expires
Attach Complete
IDLE
TMSI Reallocation Complete (GSM09.18)
IDLE

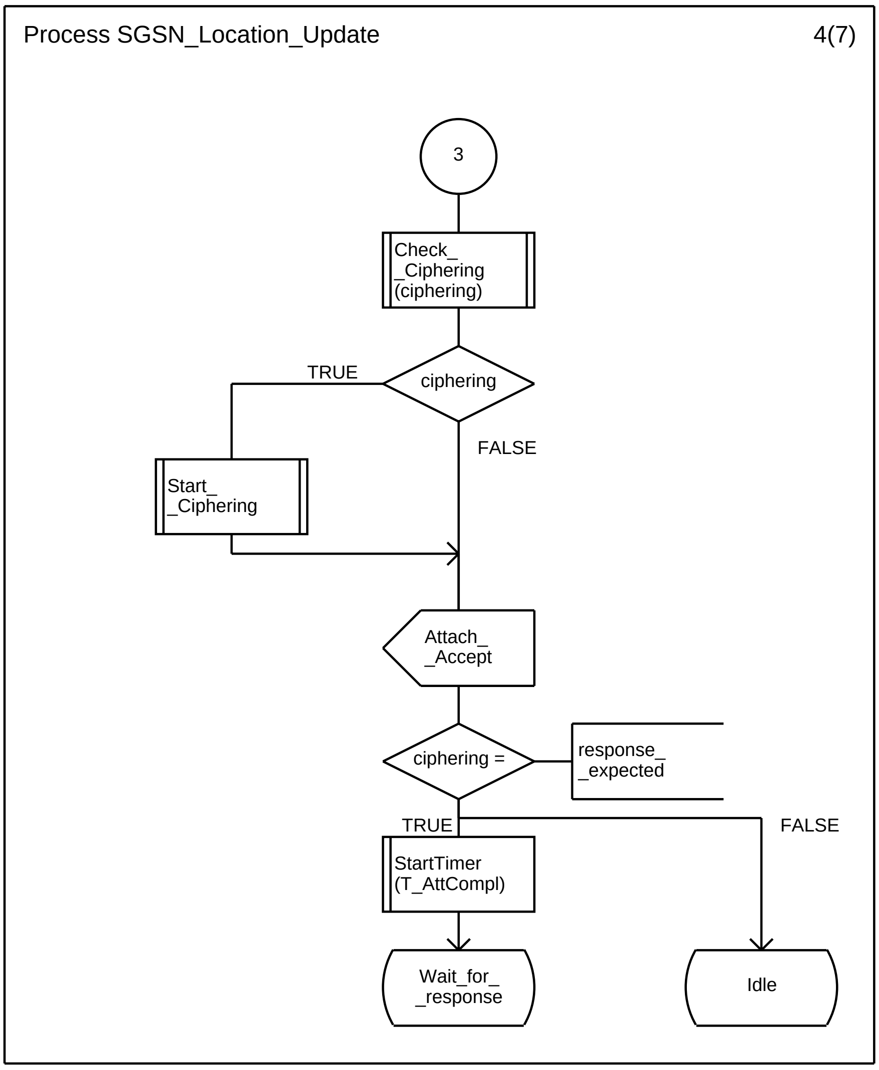
A2.2 Descriptive SDL

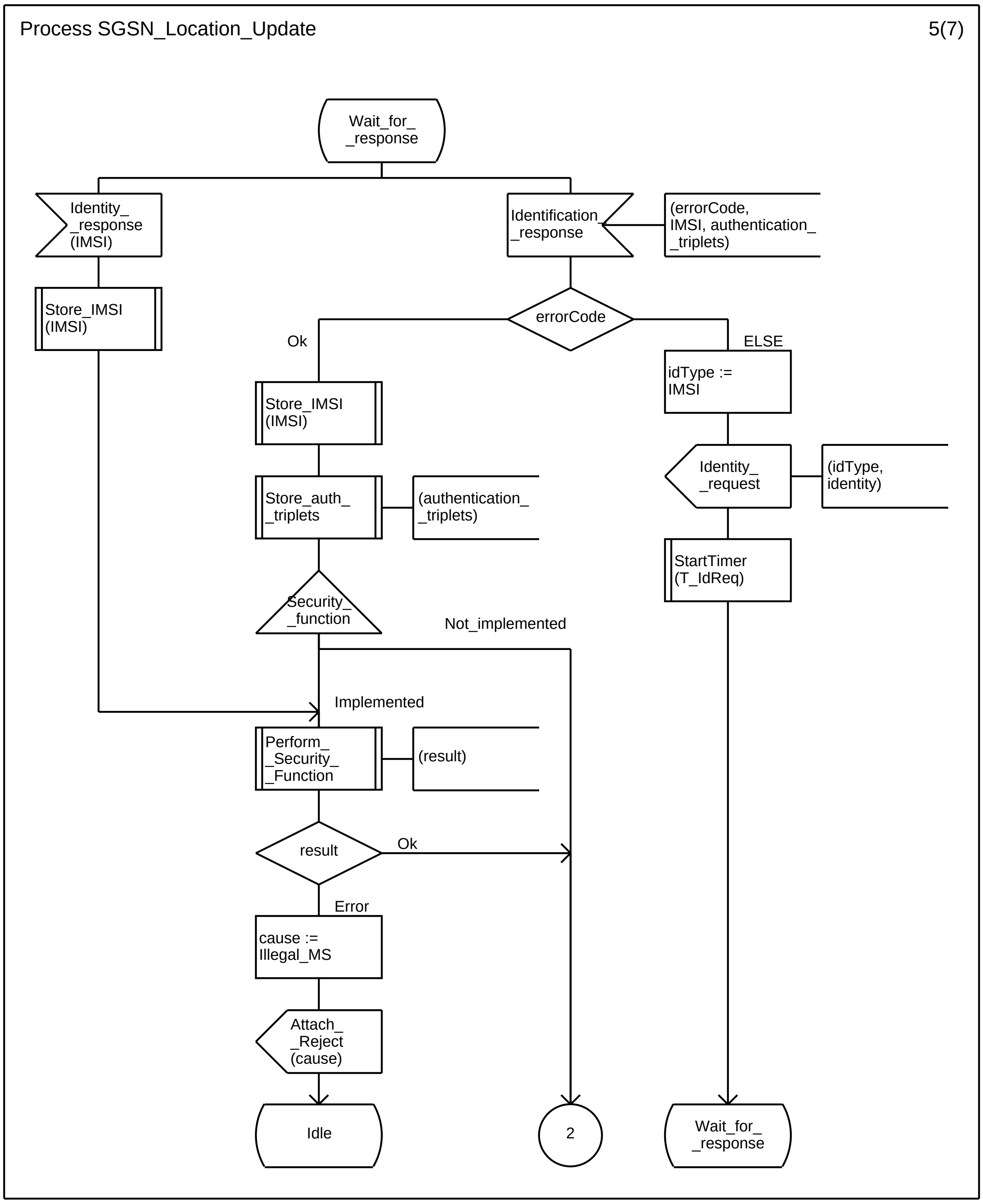


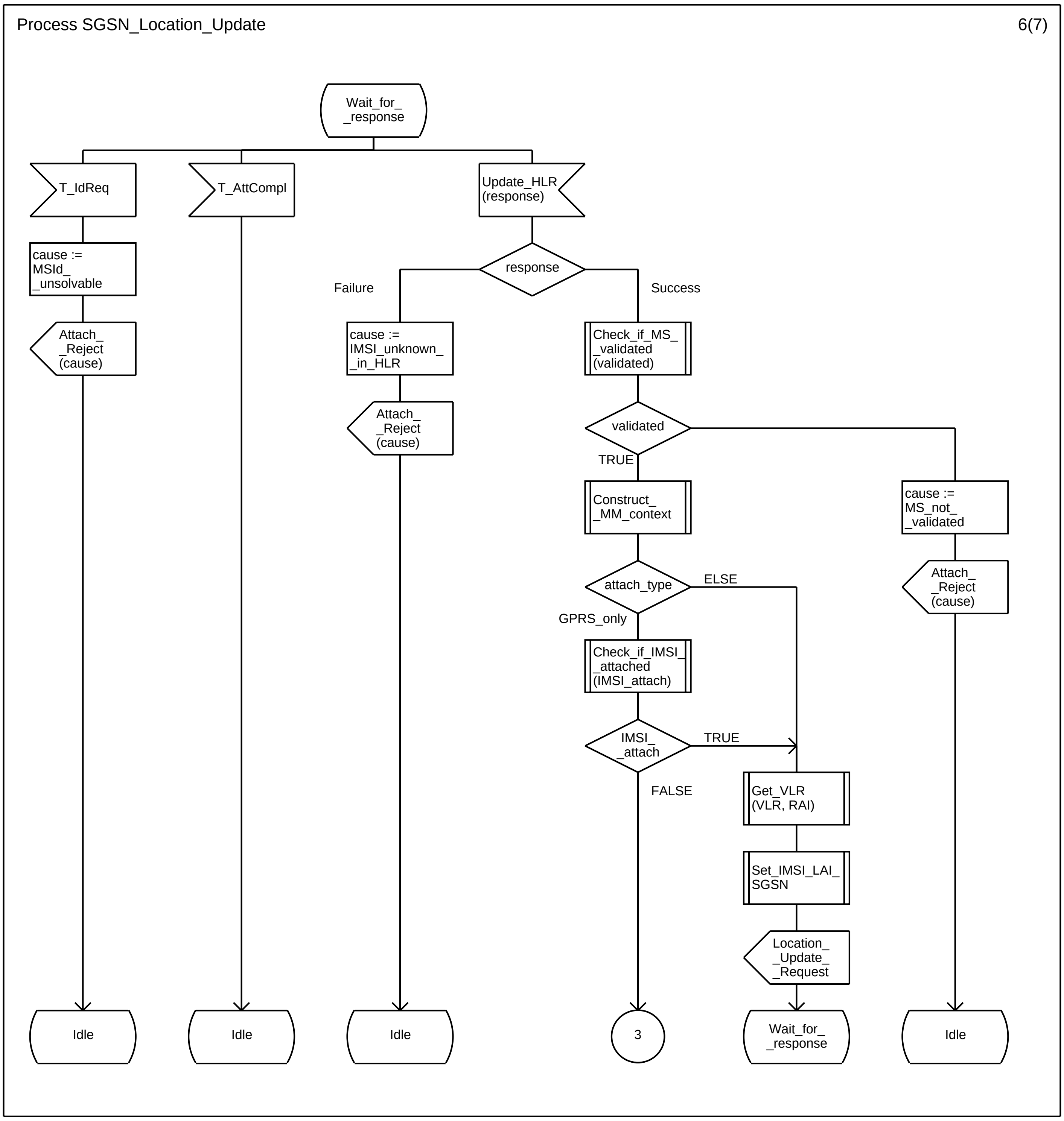

Process SGSN_Location_Update
2(7)
1
Perform_
Security
_Function
(result)
result
Ok
Error
cause :=
Illegal_MS
Attach_
_Reject
(cause)
Idle
2
Located on
page 3

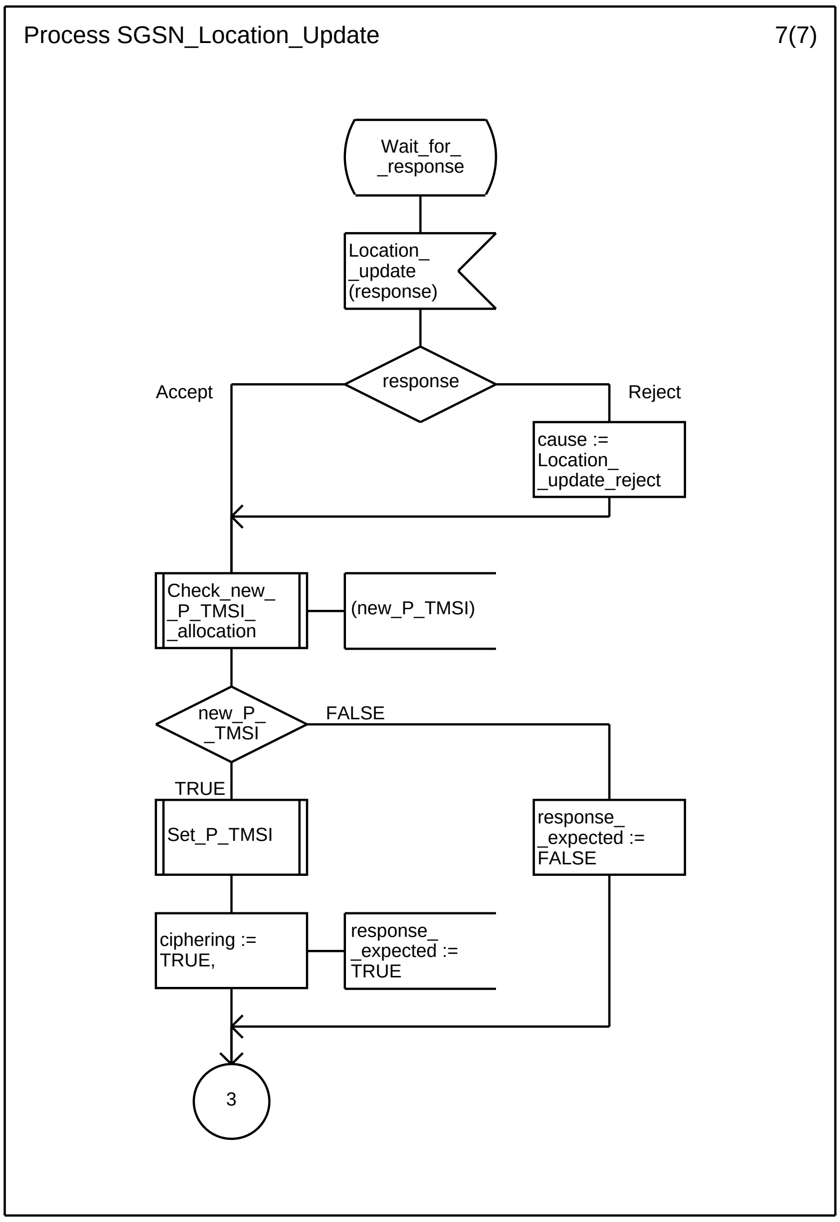


Process SGSN_Location_Update
3(7)
2
Identity_
_Check_Proc
Implemented
Not_implemented
Check_IMEI
(result)
result
Ok
Error
cause :=
IMEI_not_
_accepted
Check_if_first_
_attach_and_
_new_SGSN
(first_and_new)
Attach_
_Reject
(cause)
first_and_new
FALSE
TRUE
Update_
_Location
attach_type
ELSE
GPRS_only
Check_if_IMSI_
_attached
(IMSI_attach)
IMSI_
_attach
TRUE
FALSE
Get_VLR
(VLR, RAI)
Set_IMSI_LAI_
SGSN
Location_
Update
_Request
Idle
Wait_for_
_response
3
Located on
page 4
Wait_for_
_response


Process SGSN_Location_Update
4(7)
3
Check_
_Ciphering
(ciphering)
ciphering
TRUE
FALSE
Start_
_Ciphering
Attach_
_Accept
ciphering =
response_
_expected
TRUE
FALSE
StartTimer
(T_AttCompl)
Wait_for_
_response
Idle


Process SGSN_Location_Update
5(7)
Wait_for_ _response
Identity_ _response (IMSI)
Identification_ _response
(errorCode, IMSI, authentication_ _triplets)
Store_IMSI (IMSI)
errorCode
Ok
ELSE
Store_IMSI (IMSI)
idType := IMSI
Identity_ _request
(idType, identity)
Store_auth_ _triplets
(authentication_ _triplets)
StartTimer (T_IdReq)
Security_ _function
Not_implemented
Implemented
Perform_ _Security_ _Function
(result)
result
Ok
Error
cause := Illegal_MS
Attach_ _Reject (cause)
Idle
2
Wait_for_ _response


Process SGSN_Location_Update
6(7)
Wait_for_ _response
T_IdReq
T_AttCompl
Update_HLR (response)
cause := MSId_ _unsolvable
Attach_ _Reject (cause)
response
Failure
Success
cause := IMSI_unknown_ _in_HLR
Attach_ _Reject (cause)
Check_if_MS_ _validated (validated)
validated
TRUE
Construct_ _MM_context
cause := MS_not_ _validated
attach_type
ELSE
GPRS_only
Attach_ _Reject (cause)
Check_if_IMSI_ _attached (IMSI_attach)
IMSI_ _attach
TRUE
FALSE
Get_VLR (VLR, RAI)
Set_IMSI_LAI_ SGSN
Location_ _Update_ _Request
Idle
Idle
Idle
3
Wait_for_ _response
Idle


Process SGSN_Location_Update
7(7)
Wait_for_ _response
Location_ _update (response)
response
Accept
Reject
cause := Location_ _update_reject
Check_new_ _P_TMSI_ _allocation
(new_P_TMSI)
new_P_ _TMSI
FALSE
TRUE
Set_P_TMSI
response_ _expected := FALSE
ciphering := TRUE,
response_ _expected := TRUE
3