

# **Attachment 4-2-2**

## **WiMAX Forum<sup>®</sup> Network Architecture**

### **Detailed Protocols and Procedures**

[Informative Annex: Hooks and Principles for Evolution]

**WMF-T33-004-R015v01**

**Note:** This Document is reproduced without any modification with the consent of the WiMAX Forum®, which owns the copyright in them.





# **WiMAX Forum<sup>®</sup> Network Architecture**

Detailed Protocols and Procedures

[Informative Annex: Hooks and Principles for Evolution]

**WMF-T33-004-R015v01**

WiMAX Forum<sup>®</sup> Approved

(2009-11-21)

**WiMAX Forum Proprietary**

Copyright © 2007-2009 WiMAX Forum. All Rights Reserved.



## **Copyright Notice, Use Restrictions, Disclaimer, and Limitation of Liability.**

Copyright 2007-2009 WiMAX Forum. All rights reserved.

The WiMAX Forum® owns the copyright in this document and reserves all rights herein. This document is available for download from the WiMAX Forum and may be duplicated for internal use, provided that all copies contain all proprietary notices and disclaimers included herein. Except for the foregoing, this document may not be duplicated, in whole or in part, or distributed without the express written authorization of the WiMAX Forum.

Use of this document is subject to the disclaimers and limitations described below. Use of this document constitutes acceptance of the following terms and conditions:

**THIS DOCUMENT IS PROVIDED “AS IS” AND WITHOUT WARRANTY OF ANY KIND. TO THE GREATEST EXTENT PERMITTED BY LAW, THE WiMAX FORUM DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF TITLE, NONINFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE WiMAX FORUM DOES NOT WARRANT THAT THIS DOCUMENT IS COMPLETE OR WITHOUT ERROR AND DISCLAIMS ANY WARRANTIES TO THE CONTRARY.**

Any products or services provided using technology described in or implemented in connection with this document may be subject to various regulatory controls under the laws and regulations of various governments worldwide. The user is solely responsible for the compliance of its products and/or services with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for its products and/or services as a result of such regulations within the applicable jurisdiction.

**NOTHING IN THIS DOCUMENT CREATES ANY WARRANTIES WHATSOEVER REGARDING THE APPLICABILITY OR NON-APPLICABILITY OF ANY SUCH LAWS OR REGULATIONS OR THE SUITABILITY OR NON-SUITABILITY OF ANY SUCH PRODUCT OR SERVICE FOR USE IN ANY JURISDICTION.**

**NOTHING IN THIS DOCUMENT CREATES ANY WARRANTIES WHATSOEVER REGARDING THE SUITABILITY OR NON-SUITABILITY OF A PRODUCT OR A SERVICE FOR CERTIFICATION UNDER ANY CERTIFICATION PROGRAM OF THE WiMAX FORUM OR ANY THIRD PARTY.**

The WiMAX Forum has not investigated or made an independent determination regarding title or noninfringement of any technologies that may be incorporated, described or referenced in this document. Use of this document or implementation of any technologies described or referenced herein may therefore infringe undisclosed third-party patent rights or other intellectual property rights. The user is solely responsible for making all assessments relating to title and noninfringement of any technology, standard, or specification referenced in this document and for obtaining appropriate authorization to use such technologies, technologies, standards, and specifications, including through the payment of any required license fees.

**NOTHING IN THIS DOCUMENT CREATES ANY WARRANTIES OF TITLE OR NONINFRINGEMENT WITH RESPECT TO ANY TECHNOLOGIES, STANDARDS OR SPECIFICATIONS REFERENCED OR INCORPORATED INTO THIS DOCUMENT.**

**IN NO EVENT SHALL THE WiMAX FORUM OR ANY MEMBER BE LIABLE TO THE USER OR TO A THIRD PARTY FOR ANY CLAIM ARISING FROM OR RELATING TO THE USE OF THIS DOCUMENT, INCLUDING, WITHOUT LIMITATION, A CLAIM THAT SUCH USE INFRINGES A THIRD PARTY’S INTELLECTUAL PROPERTY RIGHTS OR THAT IT FAILS TO COMPLY WITH APPLICABLE LAWS OR REGULATIONS. BY USE OF THIS DOCUMENT, THE USER WAIVES ANY SUCH CLAIM AGAINST THE WiMAX FORUM AND ITS MEMBERS RELATING TO THE USE OF THIS DOCUMENT.**

The WiMAX Forum reserves the right to modify or amend this document without notice and in its sole discretion. The user is solely responsible for determining whether this document has been superseded by a later version or a different document.

“WiMAX,” “Mobile WiMAX,” “Fixed WiMAX,” “WiMAX Forum,” “WiMAX Certified,” “WiMAX Forum Certified,” the WiMAX Forum logo and the WiMAX Forum Certified logo are trademarks of the WiMAX Forum. Third-party trademarks contained in this document are the property of their respective owners.

## TABLE OF CONTENTS

<b>1. SCOPE</b>	<b>1</b>
<b>2. DETAILED DESCRIPTION OF TLV RELATED ERROR HANDLING INCLUDING EXAMPLES</b>	<b>1</b>
2.1 TLV STRUCTURE; REPETITION NUMBERS	1
2.2 TLV ERRORS	2
2.3 EXAMPLES	3
<b>3. USAGE OF THE HOOKS IN THE BASELINE VERSION</b>	<b>12</b>
<b>4. GUIDELINES FOR THE COMPATIBLE EVOLUTION OF THE ASN CONTROL PROTOCOL</b>	<b>12</b>
4.1 CAPABILITY NEGOTIATIONS	12
4.2 CHANGES IN THE OVERALL MESSAGE STRUCTURE	12
<b>5. GUIDELINES FOR HANDLING OF LEGACY NODES</b>	<b>14</b>
5.1 DEFINITION	14
5.2 LEGACY NODE DISCOVERY	14
5.3 GUIDELINES FOR HANDLING LEGACY NODES	14
<b>6. FUTURE USE OF THE HOOKS</b>	<b>18</b>
6.1 TERMS AND ABBREVIATIONS	18
6.2 ELEMENTS OF THE HOOKS	18
6.3 FUTURE COMPATIBLE EVOLUTION	20
6.4 PARSING OF UNKNOWN MESSAGES	20
6.5 CAPABILITY NEGOTIATION/INDICATION	20
6.6 ENHANCEMENTS WITHOUT CAPABILITY NEGOTIATION	20
6.7 COMPATIBLE VERSION MANAGEMENT	21
6.8 LEGACY HANDLING	21
<b>7. FUNDAMENTAL RULES</b>	<b>23</b>
7.1 RULES	23
<b>Table of Figures</b>	
Figure 1 – top level TLV error	4
Figure 2 – second level TLV error	5
Figure 3 – third level TLV error	6
Figure 4 – third level TLV error	7
Figure 5 – third level TLV error	8
Figure 6 – third level TLV error	9
Figure 7 – second level TLV error (repeated TLV)	10
Figure 8 – second level TLV error (repeated TLV)	11

## 1 **Revision History**

November 6, 2009	Initial version of Release 1.5.
------------------	---------------------------------





---

## 1. Scope

This specification contains hooks for the compatible evolution of the ASN control protocol; main hooks are given by the Error Handling and handling of unknown and inopportune control information. This Annex contains

- detailed description of TLV related error handling including examples;
- guidelines for using the hooks in the baseline version;
- guidelines for the compatible evolution of the ASN control protocol;
- guidelines for the ASN control protocol related handling of legacy nodes;
- a description how the hooks for the compatible evolution of the ASN control protocol may be used in future to elaborate a scheme for compatible evolution;
- fundamental rules the hooks and mechanisms for compatible evolution of this specification are built upon.

This Annex is not normative.

---

## 2. Detailed description of TLV related error handling including examples

This section contains more detailed descriptions related to the handling of errors and unknown and inopportune control information.

### 2.1 TLV structure; repetition numbers

A message of the ASN control protocol is structured like an inverted tree, where the message header is the root, and subordinate TLVs are branches.

A message may contain *top level TLVs*:

- a given TLV is a top level TLV of a message if
  - o it is contained in the message;
  - o there is no other TLV containing the given TLV.

If a message contains top level TLVs

$$TLV_0, TLV_1, \dots, TLV_k, \dots, TLV_n$$

in that given order, then for  $0 \leq k \leq n$ ,  $R_k$  is the *repetition number of*  $TLV_k$  *at the message level* if there are exactly  $R_k$  TLVs amongst  $TLV_0, TLV_1, \dots, TLV_{k-1}$  having the same Type as  $TLV_k$ :

- $R_k = 0$  if there is no TLV amongst  $TLV_0, TLV_1, \dots, TLV_{k-1}$  with the same Type as  $TLV_k$ ;
- $R_k = 1$  if there is exactly one TLV amongst  $TLV_0, TLV_1, \dots, TLV_{k-1}$  with the same Type as  $TLV_k$ ;
- $R_k = 2$  if there are exactly two TLV amongst  $TLV_0, TLV_1, \dots, TLV_{k-1}$  with the same Type as  $TLV_k$ ;
- and so on.

Note 1: This means that the repetition number of  $TLV_k$  is the number of TLVs

- with same Type as  $TLV_k$  and
- occurring in the list  $TLV_0, TLV_1, \dots, TLV_{k-1}$  (that is: before and excluding  $TLV_k$ )

Note 2: Given top level TLVs

$$TLV_0, TLV_1, \dots, TLV_k, \dots, TLV_n$$

in that given order, then the repetition number  $R_k$  of  $TLV_k$  at the message level is determined by the following pseudo-code:

$R_k := 0$

1 DO FOR  $i := 0, \dots, k - 1$

2 IF  $\text{Type}(\text{TLV}_i) = \text{Type}(\text{TLV}_k)$  THEN  $R_k := R_k + 1$  FI

3 OD.

4 Example:

5 If a message contains TLVs

6  $\text{TLV}_0, \text{TLV}_1, \text{TLV}_2, \text{TLV}_3, \text{TLV}_4, \text{TLV}_5, \text{TLV}_6$

7 (in that given order) with Types

8 23, 34, 23, 34, 5, 6, 34

9 (in that given order), then  $R_0 = 0, R_1 = 0, R_2 = 1, R_3 = 1, R_4 = 0, R_5 = 0, R_6 = 2$ .

10 For example, for  $k = 0$ , the list  $\text{TLV}_0, \dots, \text{TLV}_{k-1}$  is empty and  $R_0 = 0$ .

11 Each TLV contained in a message may contain further TLVs. A TLV ( $\text{TLV}_1$ ) is the *parent TLV* of a TLV ( $\text{TLV}_2$ ),  
12 if it *directly contains*  $\text{TLV}_2$ , i.e.:

13 -  $\text{TLV}_1$  contains  $\text{TLV}_2$ ;

14 - there is no other TLV ( $\text{TLV}_3$ ) such that  $\text{TLV}_1$  contains  $\text{TLV}_3$  and  $\text{TLV}_3$  contains  $\text{TLV}_2$ .

15 A TLV ( $\text{TLV}_2$ ) is the *child TLV* of a TLV ( $\text{TLV}_1$ ) if  $\text{TLV}_1$  is the parent TLV of  $\text{TLV}_2$ .

16 If a given TLV is the parent TLV of TLVs

17  $\text{TLV}_0, \text{TLV}_1, \dots, \text{TLV}_k, \dots, \text{TLV}_n$

18 in that given order, then  $R_k$  is the *repetition number of*  $\text{TLV}_k$  *at the level of* the given TLV if there are exactly  $R_k$   
19 TLVs amongst  $\text{TLV}_0, \text{TLV}_1, \dots, \text{TLV}_{k-1}$  having the same Type as  $\text{TLV}_k$  :

20 -  $R_k = 0$  if there is no TLV amongst  $\text{TLV}_0, \text{TLV}_1, \dots, \text{TLV}_{k-1}$  with the same Type as  $\text{TLV}_k$  ;

21 -  $R_k = 1$  if there is exactly one TLV amongst  $\text{TLV}_0, \text{TLV}_1, \dots, \text{TLV}_{k-1}$  with the same Type as  $\text{TLV}_k$  ;

22 -  $R_k = 2$  if there are exactly two TLV amongst  $\text{TLV}_0, \text{TLV}_1, \dots, \text{TLV}_{k-1}$  with the same Type as  $\text{TLV}_k$  ;

23 - and so on.

24 Example:

25 If the given contains TLVs

26  $\text{TLV}_0, \text{TLV}_1, \text{TLV}_3, \text{TLV}_3, \text{TLV}_4, \text{TLV}_5, \text{TLV}_6$

27 (in that given order) with Types

28 23, 34, 23, 34, 5, 6, 34

29 (in that given order), then  $R_0 = 0, R_1 = 0, R_2 = 1, R_3 = 1, R_4 = 0, R_5 = 0, R_6 = 2$ .

30 Section 3.4.1.3 of the baseline stage 3 specification defines the notion of *ancestor TLV* of a given TLV. By that  
31 definition, the ancestor TLV of a given TLV is a higher level TLV encapsulating the given TLV, more exactly:

32 - the parent TLV of a given TLV is an ancestor of the given TLV;

33 - the parent TLV of the parent TLV of a given TLV is an ancestor TLV of the given TLV;

34 - the parent TLV of an ancestor TLV of a given TLV is an ancestor TLV of the given TLV;

35 - a top level TLV has no ancestor TLV.

36 A TLV ( $\text{TLV}_2$ ) is a *descendent TLV* of a TLV ( $\text{TLV}_1$ ) if  $\text{TLV}_1$  is an ancestor TLV of  $\text{TLV}_2$ .

## 37 2.2 TLV errors

38 Section 3.4.1.3 of the baseline stage 3 specification defines the notion of a TLV that *surrounds an error* . By that  
39 definition, a TLV surrounds an error if

40 - it directly contains the error or

41 - it is the parent TLV of a TLV surrounding the error.

42 By that definition, each ancestor TLV of a TLV that surrounds an error also surrounds the error.

1 A TLV indicating 'comprehension not required' (TC=1, see section 5.3.1 of the baseline stage 3 specification) is also  
2 known as a *skipable* TLV.

3 Section 3.4.1.3 of the baseline stage 3 specification defines the notion of the *closest skipable* TLV of an error. By  
4 that definition, the closest skipable TLV of an error is the first ancestor TLV encountered tracing back toward the  
5 root from the erroneous TLV, including the erroneous TLV itself, for which comprehension is not required (TC=1):

- 6 - if the TLV directly containing an error indicates 'comprehension required' (TC = 0, see section 5.3.1 of the  
7 baseline stage 3 specification) and all its ancestor TLVs indicate 'comprehension required' as well, then there is  
8 no closest skipable TLV of this error;
- 9 - if the TLV directly containing an error indicates 'comprehension not required', then it is the closest skipable TLV  
10 of the error;
- 11 - if a given TLV surrounds an error and indicates 'comprehension not required' and all its descendent TLVs  
12 surrounding the error indicate 'comprehension required' then the given TLV is the closest skipable TLV of the  
13 error.

## 14 **2.3 Examples**

15 In the examples below, an error is said to occur 'directly' in a TLV if

- 16 - the error was diagnosed in a field of the TLV; or
- 17 - the error consists in the Type of the TLV being not known in the message or in the parent TLV; or
- 18 - the error occurred because the TLV is an unforeseen repetition.

19 In the examples, it is assumed that the indicated error is the first diagnosed error.

20 In Figure 1, an error is diagnosed directly in TLV(n+1). As it is a top level TLV indicating 'comprehension not  
21 required', it is the closest skipable TLV of the error. The message is further processed as if TLV(n+1) was not  
22 contained in the message.

## Top Level error

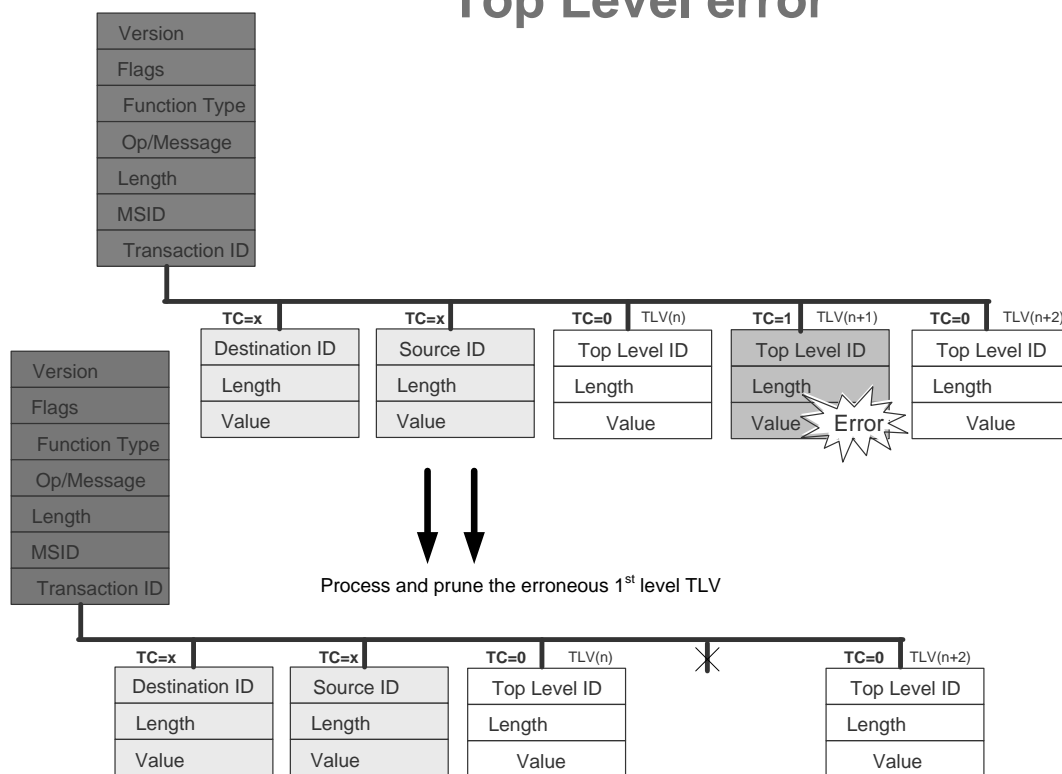
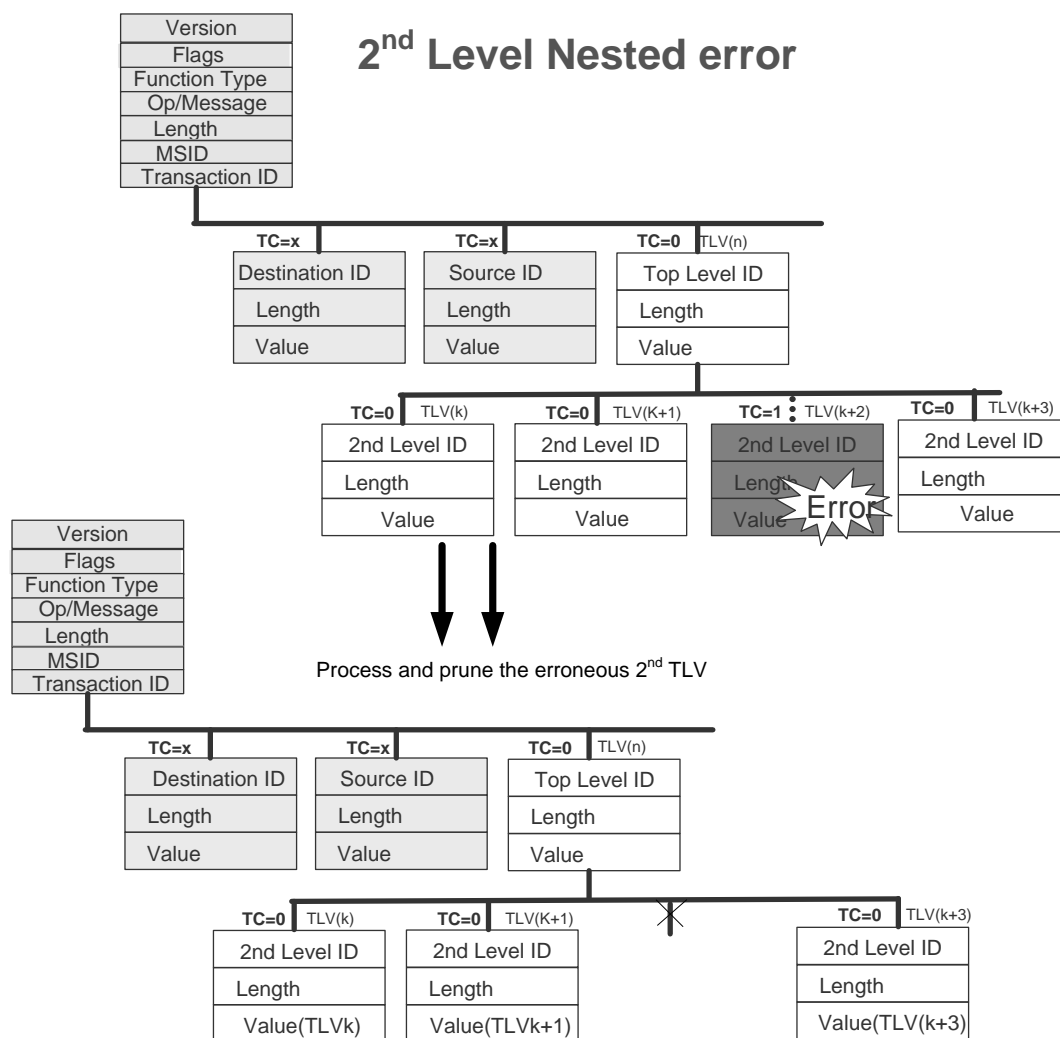


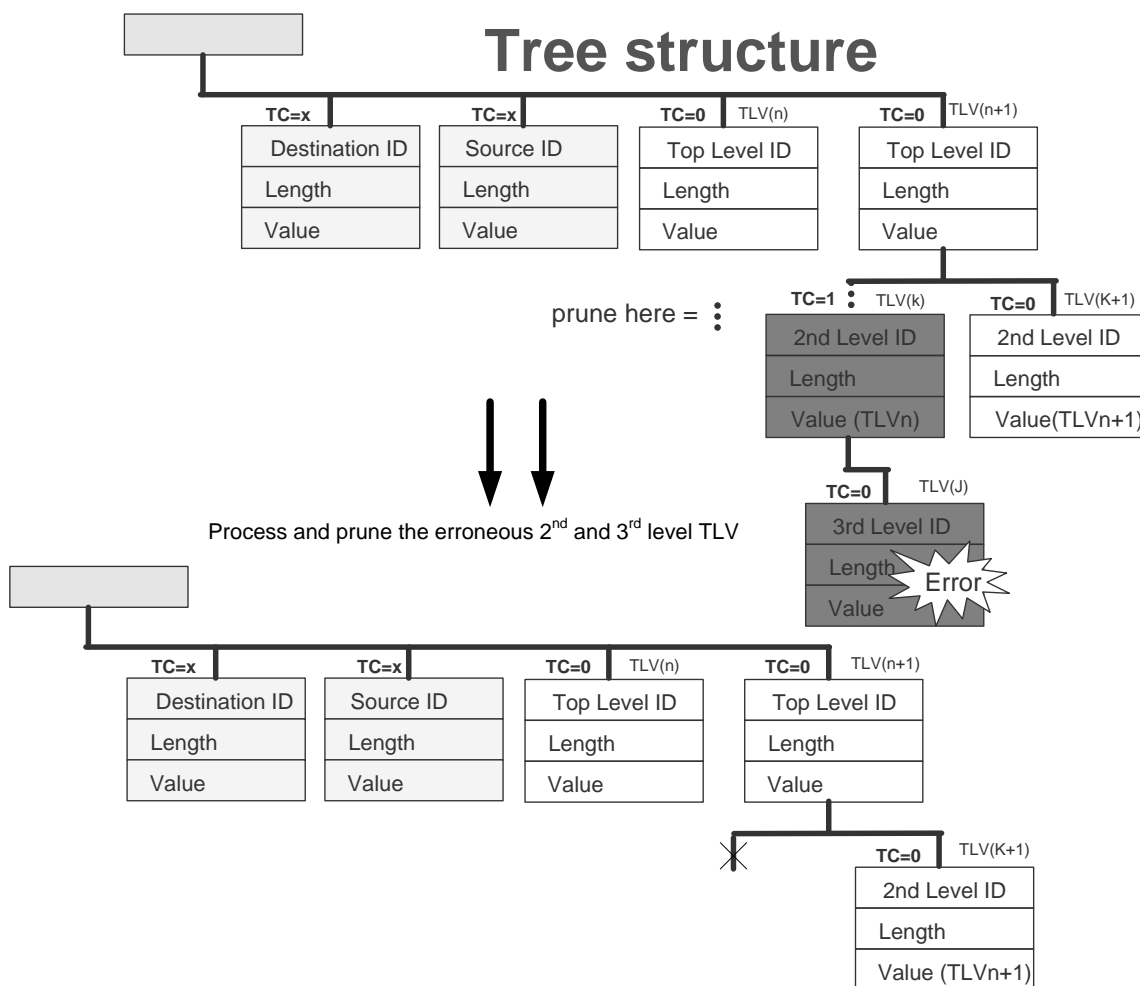
Figure 1 – top level TLV error

- 1 In Figure 2, an error is diagnosed directly in TLV(k+2). As this TLV indicates 'comprehension not required', it is the
- 2 closest skipable TLV of the error. The message is further processed as if TLV(k+2) was not contained in the
- 3 message.



**Figure 2 – second level TLV error**

- 1 In Figure 3, an error is diagnosed directly in TLV(J). This TLV indicates 'comprehension required'; the parent TLV
- 2 is TLV(k) and indicates 'comprehension not required'; therefore TLV(k) is the closest skipable TLV of the error. The
- 3 message is further processed as if TLV(k) was not contained in the message.



**Figure 3 – third level TLV error**

- 1 In Figure 4, an error is diagnosed directly in TLV(J). This TLV indicates 'comprehension not required'; therefore
- 2 TLV(J) is the closest skipable TLV of the error. The message is further processed as if TLV(J) was not contained in
- 3 the message.

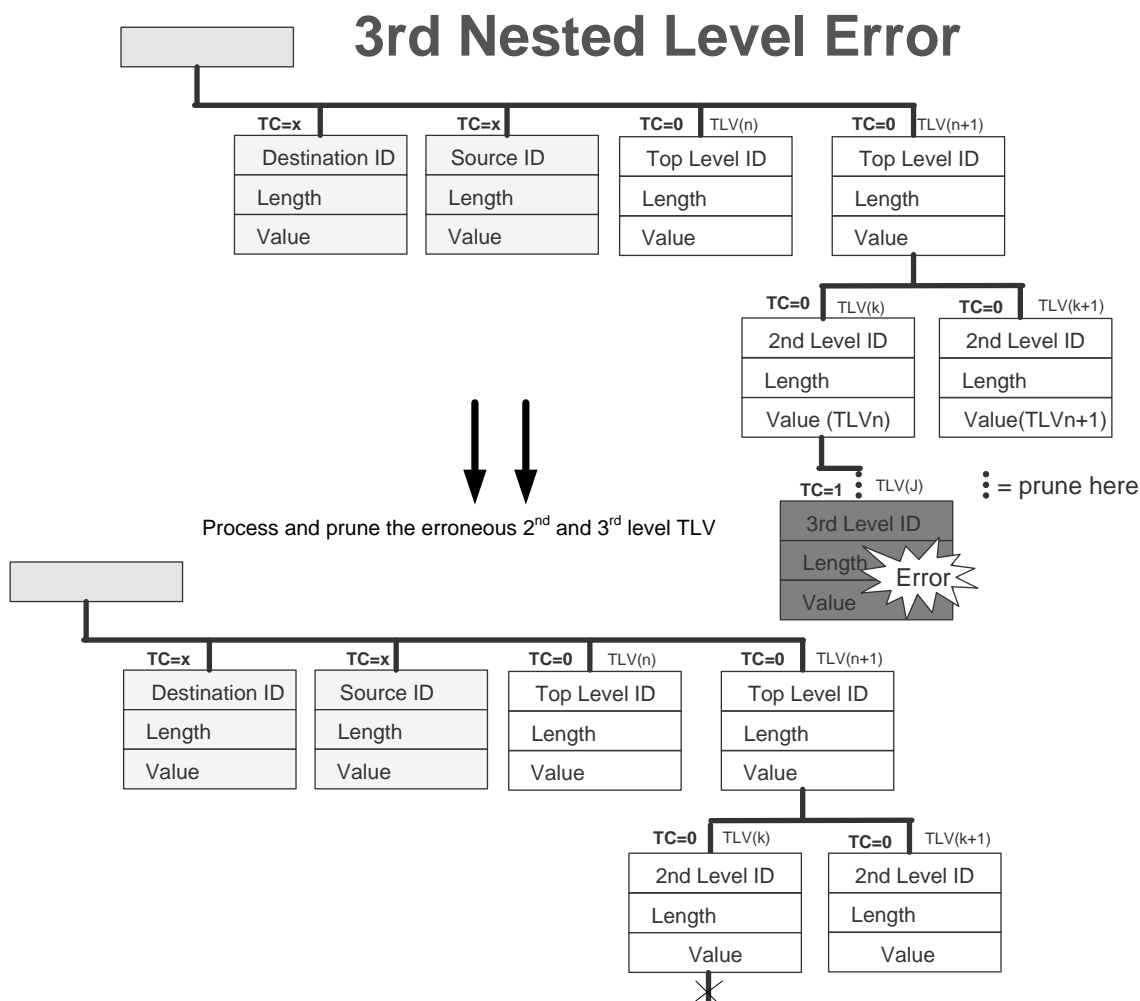


Figure 4 – third level TLV error

In Figure 5, an error is diagnosed directly in TLV(J). This TLV indicates 'comprehension required'; the parent TLV is TLV(k) and indicates 'comprehension required'; these TLVs are not skipable. However the parent TLV of TLV(k) is TLV(n+1) and indicates 'comprehension not required'; TLV(n+1) is an ancestor TLV of TLV(J); it is skipable and evidently it is the closest skipable TLV of the error. The message is further processed as if TLV(n+1) was not contained in the message.

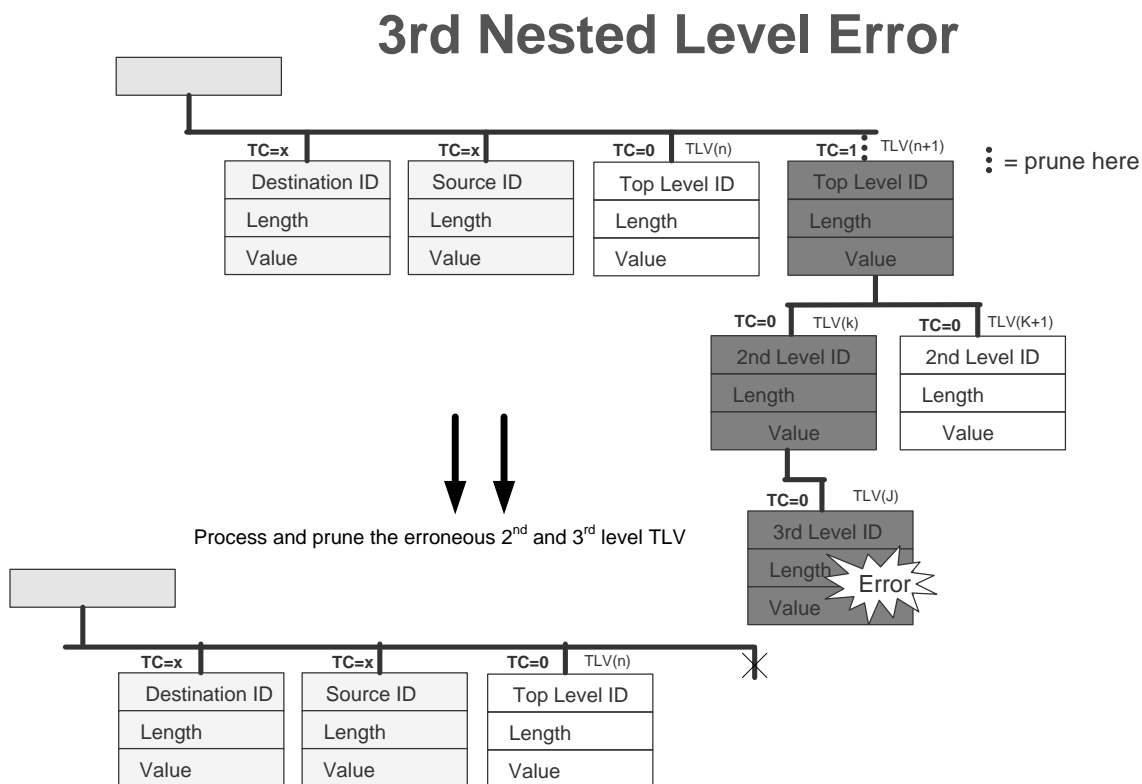


Figure 5 – third level TLV error



- 1 In Figure 6, an error is diagnosed directly in TLV(J). This TLV indicates 'comprehension required'; the parent TLV
- 2 is TLV(k) and indicates 'comprehension required'; the parent TLV of TLV(k) is TLV(n+1) and indicates
- 3 'comprehension required'; there is no ancestor TLV of TLV(J) indicating 'comprehension not required'; therefore
- 4 there is no closest skipable TLV of the error. The message is rejected and an error is reported.

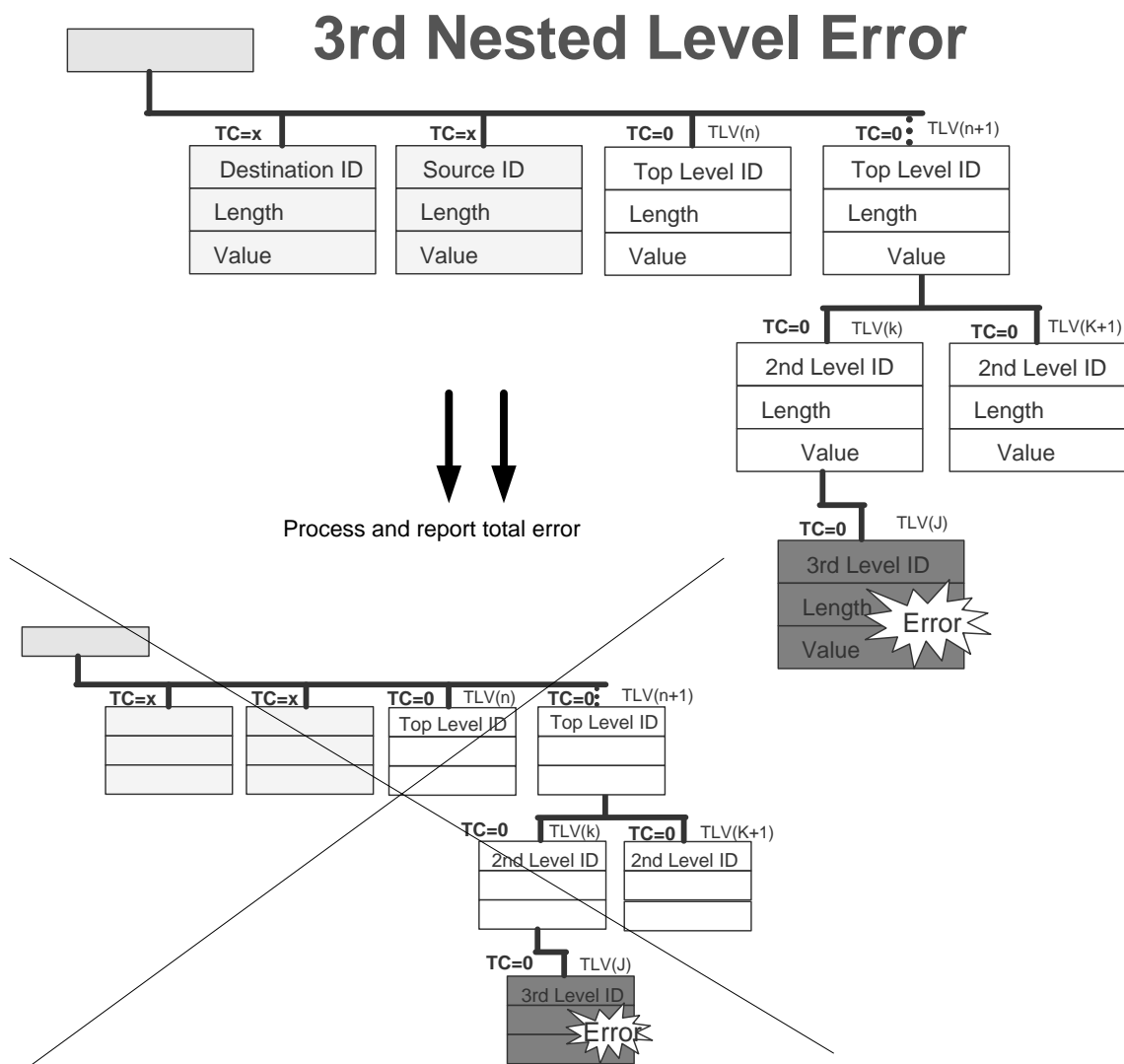
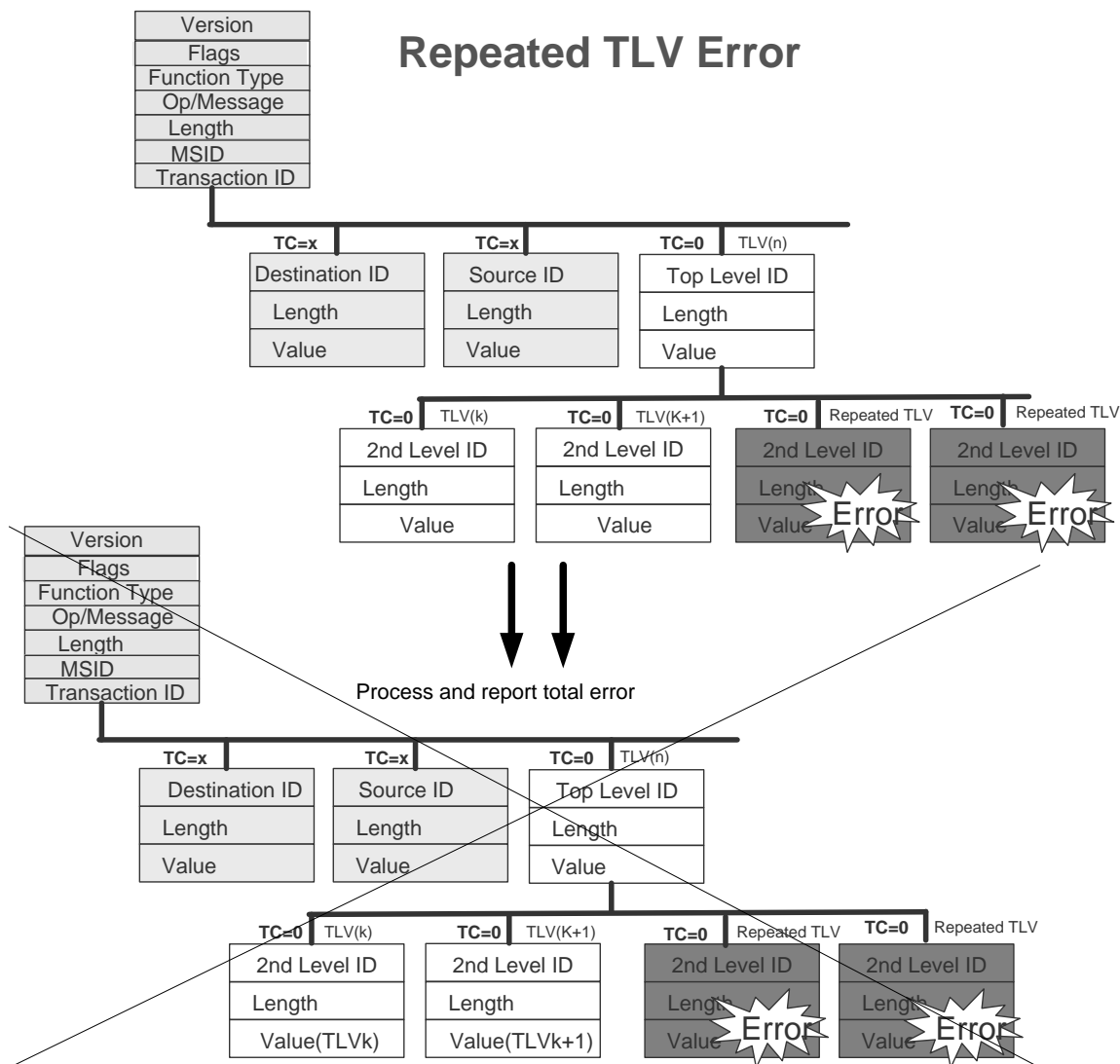


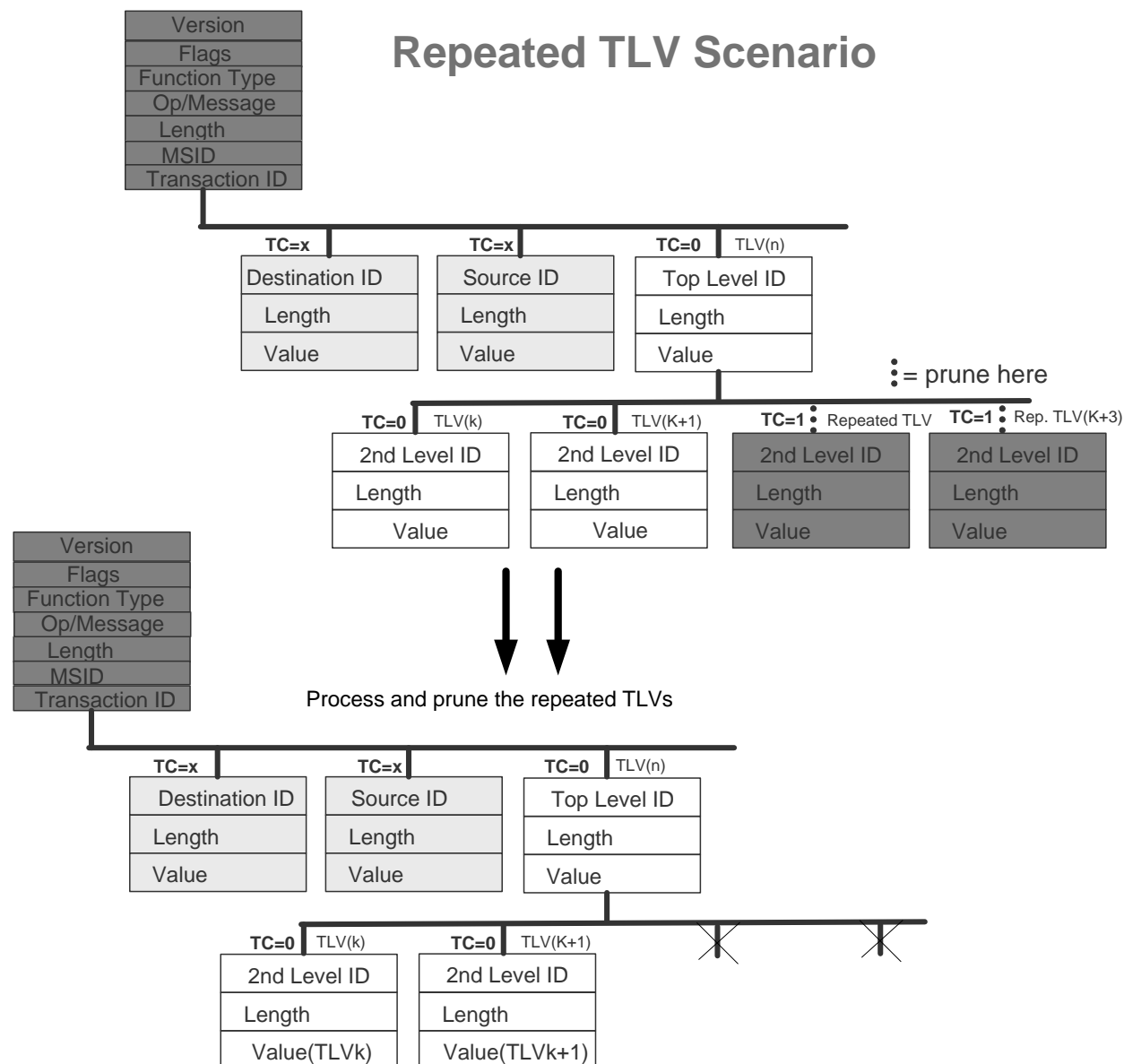
Figure 6 – third level TLV error

In Figure 7, an error directly occurs in both of the TLVs indicated as 'Repeated TLVs'. As the order of processing of TLVs is implementation dependent, an error may be diagnosed in any one of them. This TLV (in which the error is diagnosed) indicates 'comprehension required'; the parent TLV is TLV(n); it is a top level TLV and indicates 'comprehension required'; therefore, there is no ancestor TLV of the Repeated TLV indicating 'comprehension not required'; therefore there is no closest skipable TLV of the error. The message is rejected and an error is reported.



**Figure 7 – second level TLV error (repeated TLV)**

In Figure 8, a message with repeated TLVs is erroneously constructed: the definition of TLV(n) does not specify the repeated occurrences TLV(K+2) and TLV(K+3). As the order of processing is implementation dependent, an error may be diagnosed when any of the duplicated TLVs is processed. As these TLVs indicate “comprehension not required”, they are the closest skipable TLVs of the repetition error. The message is further processed as if the repeated TLV(k+2) and TLV(k+3) were not contained in the message.



**Figure 8 – second level TLV error (repeated TLV)**

---

### 3. Usage of the hooks in the baseline version

The baseline version is Version 1.3 of this specification.

(Bas 1) In the baseline version, the sender should indicate that comprehension is required for function type, message type, and OP ID (C bit = 1).

Note: This setting may also be used when sent to a legacy node, however a legacy node will ignore the indication.

(Bas 2) If the Relay mode of operation is used (see section 3.1.1 of the baseline stage 3 specification), the Destination Identifier TLV and Source Identifier TLV, included as the first TLVs in the message, SHALL indicate that TLV comprehension is required (TC bit = 0).

Note: For the procedures of section 3.4 of the baseline stage 3 specification, setting of the TC bit in the occurrences of Destination Identifier TLV and Source Identifier TLV addressed in (Bas 2) is irrelevant for the receiver. However, requiring the sender to set TC bit = 0

- allows a refined usage of the TC bit in the future evolution of the ASN control protocol (if wanted);

- can also be used when sent to a legacy node, cf. section 5 of this annex. The legacy node would ignore a TLV encoded with TC bit = 1.

(Bas 3) The TC bit sets the requirements for error handling by the receiver. However, in the baseline version, TLVs may be sent as indicating "TLV comprehension required" (TC = 0) even in the case of a TLV where the protocol does not require the receiver to process the TLV, see section 3.4 of the baseline stage 3 specification.

Note: Setting TC bit = 0 should also be used when sent to a legacy node, cf. section 5 of this annex. The legacy node would ignore a TLV encoded with TC bit = 1.

---

### 4. Guidelines for the compatible evolution of the ASN control protocol

This section describes guidelines for the compatible evolution of the ASN control protocol (i.e., protocol on R4, R6 and R8).

#### 4.1 Capability negotiations

Capability negotiation shall be introduced in new versions of the stage 3 specification. It may be specified in a global way (new function type for negotiating capabilities) or per function type (messages within the function type) or in a combination of both. Typical transaction structures can be used: for example capability indication, possibly on request, of both sides.

#### 4.2 Changes in the overall message structure

(MS 1) Future versions of Stage 3 should not change the structure of message header concerning the position and length of the following fields:

- *Version 1 indicator* field of the message header;
- *Flags* field of the message header;
- *Function type* field of the message header;
- *OP ID* field of the message header;
- *Message type* field of the message header;

- 1                   - *Length* field of the message header;
- 2                   - *MSID* field of the message header;
- 3                   - *Transaction ID* field of the message header;
- 4 (MS 2)       Future versions of Stage 3 should not change the meaning and position of the following bits in the *Flags*
- 5                   field of the message header:
- 6                   - R bit, T bit, S bit, and E bit.
- 7 (MS 3)       Future versions of Stage 3 should not change the meaning and position of the following TLVs in the
- 8                   message body:
- 9                   - *Destination Identifier* TLV as first TLV in the message body, if the T bit is set;
- 10                  - *Source Identifier* TLV as second TLV in the message body, if the T bit is set.
- 11

## 5. Guidelines for handling of legacy nodes

### 5.1 Definition

A *legacy node* is a node compliant to an early version of stage 3 (Version 1.2 and earlier). A legacy node hence does not implement the hooks for compatibility handling.

### 5.2 Legacy node discovery

A legacy node is recognized by

- the fact that it sets value 1 in the *Version 1 indicator* field (which was earlier called the *Version* field);
- the fact that it does not set the S bit in the *Flags* field.

### 5.3 Guidelines for handling legacy nodes

(LN 1) Messages of a function type other than the following ones SHOULD not be sent to a legacy node:

- QoS
- HO Control
- Data Path Control
- Context Transfer
- R3 Mobility
- Paging
- RRM
- Authentication Relay
- MS State
- IM Operations
- Accounting

Reason: In pre-baseline versions of Stage 3, an error code 2, 'Invalid Function Type' is specified; on the other hand, section 3.4 specifies:

11	Unexpected message received: Message received in unexpected state, Function or Node	Discard the message, no response generated.
----	--	---

So a legacy node receiving a message with unknown Function type would probably discard the message with or without reporting an error.

(LN 2) Messages with a Message type other than the ones indicated in the table below should not be sent to a legacy node. The table below refers to R1v1.3. A legacy node may not recognize all of these, and react in an unpredictable way, e.g. by ignoring the message or reporting an error. The sender may try to recognize which messages are probably not recognized by the legacy receiver and take suitable action.

Function Type	Message
1 (QoS)	<i>RR_Req</i>
	<i>RR_Rsp</i>
	<i>RR_Ack</i>
2 (HO Control)	<i>HO_Req</i>
	<i>HO_Rsp</i>
	<i>HO_Ack</i>

Function Type	Message
	<i>HO_Cnf</i>
	<i>HO_Complete</i>
	<i>HO_Directive</i>
	<i>HO_Directive_Rsp</i>
3 (Data Path Control)	<i>Path_Dereg_Req</i>
	<i>Path_Dereg_Rsp</i>
	<i>Path_Dereg_Ack</i>
	<i>Path_Modification_Req</i>
	<i>Path_Modification_Rsp</i>
	<i>Path_Modification_Ack</i>
	<i>Path_Prereg_Req</i>
	<i>Path_Prereg_Rsp</i>
	<i>Path_Prereg_Ack</i>
	<i>Path_Reg_Req</i>
	<i>Path_Reg_Rsp</i>
	<i>Path_Reg_Ack</i>
	<i>IM_Exit_State_Ind</i>
	<i>IM_Exit_State_Ind_Ack</i>
4 (Context Transfer)	<i>Context_Req</i>
	<i>Context_Rpt</i>
	<i>Context_Ack</i>
	<i>CMAC_Key_Count_Update</i>
	<i>CMAC_Key_Count_Update_Ack</i>
	Prepaid Request
	Prepaid Notify
5 (R3 Mobility)	<i>Anchor_DPF_HO_Req</i>
	<i>Anchor_DPF_HO_Trigger</i>
	<i>Anchor_DPF_HO_Rsp</i>
	<i>Anchor_DPF_Relocate_Req</i>
	<i>Anchor_DPF_Relocate_Rsp</i>
	<i>FA_Register_Req</i>
	<i>FA_Register_Rsp</i>
	<i>FA_Revoke_Req</i>
	<i>FA_Revoke_Rsp</i>
	<i>Anchor_DPF_Release_Req</i>
	<i>Relocation_Ready_Req</i>
	<i>Relocation_Ready_Rsp</i>
6 (Paging)	<i>Paging_Announce</i>
	<i>Delete_MS_Entry_Req</i>
	<i>PC_Relocation_Ind</i>
	<i>PC_Relocation_Ack</i>
	<i>Delete_MS_Entry_Rsp</i>
	<i>Anchor_PC_Ind</i>
	<i>Anchor_PC_Ack</i>
7 (RRM)	<i>R6_PHY_Parameters_Req</i>
	<i>R6_PHY_Parameters_Rpt</i>
	<i>R4/R6_Spare_Capacity_Req</i>
	<i>R4/R6_Spare_Capacity_Rpt</i>

Function Type	Message
	<i>R6 Neighbor_BS_Resource_Status_Update</i>
	<i>R4/R6 Radio_Config_Update_Req</i>
	<i>R4/R6 Radio_Config_Update_Rpt</i>
	<i>R4/R6 Radio_Config_Update_Ack</i>
8 (Authentication Relay)	<i>AR_EAP_Start</i>
	<i>AR_EAP_Transfer</i>
	Bulk Interim Update
	Bulk Interim Update_Ack
9 (MS State)	<i>MS_PreAttachment_Req</i>
	<i>MS_PreAttachment_Rsp</i>
	<i>MS_PreAttachment_Ack</i>
	<i>MS_Attachment_Req</i>
	<i>MS_Attachment_Rsp</i>
	<i>MS_Attachment_Ack</i>
	<i>Key_Change_Directive</i>
	<i>Key_Change_Cnf</i>
	<i>Key_Change_Ack</i>
	<i>Relocation_Complete_Req</i>
	<i>Relocation_Complete_Rsp</i>
	<i>Relocation_Complete_Ack</i>
	<i>Relocation_Notify</i>
	<i>Relocation_Req</i>
	<i>Relocation_Rsp</i>
	<i>NetExit_MS_State_Change_Req</i>
	<i>NetExit_MS_State_Change_Rsp</i>
	<i>Relocation_Notify_Rsp</i>
10 IM Operations	<i>IM_Entry_State_Change_Req</i>
	<i>IM_Entry_State_Change_Rsp</i>
	<i>IM_Entry_State_Change_Ack</i>
	<i>IM_Exit_State_Change_Req</i>
	<i>IM_Exit_State_Change_Rsp</i>
	<i>Initiate_Paging_Req</i>
	<i>Initiate_Paging_Rsp</i>
	<i>LU_Req</i>
	<i>LU_Rsp</i>
	<i>LU_Cnf</i>
11 Accounting	<i>Hot_lining_Req</i>
	<i>Hot_lining_Rsp</i>

Reason: In pre-baseline versions of Stage 3, an error code 3, 'Invalid Message Type' is specified; on the other hand, section 3.4 specifies in pre-baseline versions of Stage 3:

11	Unexpected message received: Message received in unexpected state, Function or Node	Discard the message, no response generated.
----	--	---

So a legacy node receiving a message with unknown Message type would probably discard the message with or without reporting an error.

(LN 3) Messages with a Version 1 indicator field (earlier called Version field) in the message header other than indicating value "one" SHOULD not be sent to a legacy node.



- Reason: In pre-baseline versions of Stage 3, an error code 1, 'Incompatible Version Number' is specified; on the other hand, an error handling is not specified in the procedural parts. So a legacy node receiving a message with unknown Version number may react in an unpredictable manner.
- (LN 4) When a message with TLV known to the legacy node is sent to the legacy node, the TC bit should be set to 0. Otherwise the legacy node will ignore the TLV. When a message with an unknown TLV is sent to a legacy node, the legacy node will ignore the TLV and process the message. When this behavior is wanted, new TLV may be sent to legacy nodes.
- (LN 5) In a message sent to a legacy node that is known to the legacy node, in a TLV known in the message to the legacy node,
- a field unknown to the legacy node SHOULD not be used;
  - a field known in the TLV to the legacy node, SHOULD not be set to a value that is not known to the legacy node.
- Reason: In pre-baseline versions of Stage 3, the reaction of a node when receiving an unknown field or an unknown value within a field is not clearly defined, so a legacy node may react in an unpredictable manner.
- (LN 6) In a message sent to a legacy node that is known to the legacy node, a TLV encoded as 'TLV comprehension not required' (TC = 1) will be discarded by the legacy node. Comprehension required (TC=0) has no such meaning to a legacy node.
- Reason: In pre-baseline versions of Stage 3, the TC bit was part of the Type field. Setting the bit to 1 indicates to the legacy node a Type with most significant bit equal to one.
- (LN 7) If an extended TLV is sent to a legacy node that is known to the legacy node, the legacy node might react in an unpredictable way.
- Reason: In pre-baseline versions of Stage 3, a corresponding error handling is not specified.

---

## 6. Future use of the hooks

This section discusses the evolution of the ASN control protocol (i.e., protocol on R4, R6 and R8) and introduces the concepts of protocol version and capability negotiation.

The concept of Protocol Version was introduced earlier, but is only suited to handle future incompatible evolution of the ASN control protocol and should not be introduced until absolutely necessary.

The concept of capability negotiation shall be introduced in new versions of the stage 3 specification to handle compatible evolution of the ASN control protocol staying within a given protocol version.

The ASN control protocol contains hooks for a compatible evolution and for forward and backward compatibility.

This section intends to show that the hooks are sufficient for a future compatible evolution of the ASN control protocol. Based on the hooks, a capability negotiation can be introduced at a later stage; this section also describes how legacy node handling will work based on the hooks.

### 6.1 Terms and abbreviations

These terms and abbreviations are only used for this paper; they are not proposed to be used in the standard.

#### 6.1.1 Terms

**Baseline entity** Entity compliant to the ASN control protocol as specified in Stage 3 Release 1, Version 1.3 and later (including the hooks for compatible evolution).

**Legacy Node** A *legacy node* is a node compliant to a Stage 3 specification version before Release 1, Version 1.3.  
Note: For discovery of legacy nodes, cf. section 4.1 of this annex.

**PV** Protocol Version; not to be mixed up with a release or the version of a specification. It is quite possible to evolve the protocol without changing the PV.

**PV1** Protocol Version 1 of this protocol specified in Stage 3 Release 1 Version 1.3 and later.

**PV1 entity** Protocol entity using PV1 (Stage 3 Release 1, Version 1.3 or later).

**PV1 only node** Node within the ASN compliant to PV1 on all interfaces R4, R6 and R8 and not supporting any higher PV on any of the interfaces R4, R6 and R8.

**Future PV1 entity** There will be new versions of Stage 3. The compatibility allows this without introducing a new PV. However in order to distinguish

- an entity compliant to the PV 1 as it is in Stage 3 Release 1, Version 1.3 and
- an entity compliant to PV 1 as it will be in a future Stage 3 V1.x,

the 'Future PV1 entity' is used to denote the latter.

#### 6.1.2 Abbreviations

PV1oN PV1 only node

### 6.2 Elements of the hooks

The hooks are based on an improved error handling and on some new flags in messages. These flags indicate requirements for comprehension and reporting for

- unknown function types and for

- unknown or inopportune message types, TLVs and values.

### 6.2.1 New function types

New function types may be introduced and used in communication with a PV1 only node (PV1oN) in a safe manner: A flag in the message indicates whether comprehension is required for the Function type:

- If comprehension is required for the Function type, the receiver PV1oN reports an error and discards the message.
- If comprehension is not required for the Function type, the receiver PV1oN discards the message without reporting an error.

Legacy handling: See section 6.8.

### 6.2.2 New message types

Within a function type that is already defined in PV1, new message types may be introduced and used in communication with a PV1 only node (PV1oN) in a safe manner: For new message types, a Comprehension Required indication can be set:

- If the Comprehension Required Indication is set, the receiver PV1oN reports an error and discards the message.
- If the Comprehension Required Indication is not set, the receiver PV1oN discards the message without reporting an error.

### 6.2.3 New or inopportune TLVs

Within a Message type that is already defined in PV1, new Information Elements (TLV) may be introduced and used in communication with a PV1 only node (PV1oN) in a safe manner: For new TLV, a Comprehension Required Indication can be set:

- If comprehension is required (TC=0) for a TLV and all of its ancestor TLVs and the TLV is unknown or inopportune in the message, the receiver PV1oN diagnoses and reports an error; the message is otherwise discarded.
- If comprehension is not required for a TLV (TC=1) and the TLV is unknown or inopportune in the message, the receiver PV1oN ignores this TLV and treats the message as if the TLV was not present.
- If comprehension is required for a TLV (TC=0) but not for all of its ancestor TLVs and the TLV is unknown or inopportune in the message, the closest ancestor TLV not requiring comprehension is skipped, i.e., the message is treated as if the TLV and its descendents were not present.

The following design decisions were made: There are in principle two different decisions, which could be made independently, resulting in four ways of reaction:

- to ignore or to reject the TLV:
  - o to ignore the TLV means to treat the message as if the TLV was not present;
  - o to reject the TLV means to consider the surrounding message (in the case of a top level TLV) or the parent TLV as erroneous.
- to trigger an error report or remain silent:
  - o to trigger an error report means
    - to send an error report in the case of a top level TLV;
    - in the case of a non top level TLV to let the parent TLV decide on triggering an error report based on its comprehension settings;
  - o to remain silent means
    - not to send an error report in the case of a top level TLV;
    - in the case of a non top level TLV not to trigger an error report.

Other schemes found in non WiMAX mobile communications protocols have chosen three of the four options: Reject and report, reject and don't report; ignore and don't report.

Here, it was decided to keep the scheme simple and to combine comprehension requirement and reporting requirement. The main reason is that this scheme seems sufficient for compatible evolution.

#### **6.2.4 Reserved bits and reserved fields**

A reserved bit is set to zero by the sender and ignored by the receiver.

A reserved value in a field is rejected by the receiver.

The protocol specification should register value ranges for less specific meaning.

### **6.3 Future compatible evolution**

Due to this compatibility scheme, a compatible evolution of PV1 can be done without danger. A major tool yet to be introduced is the capability negotiation. Another tool is provided by the hooks themselves.

Note that the objective is to leave the Baseline entity compliant to the evolution of PV1: The behavior of a Baseline entity should be a 'legal' behavior in future versions of PV1;

Note also that a Future PV1 entity will be 'almost' compliant to the Baseline version of the specification: This is because it would react to new messages in a specific new way, whereas the Baseline entity would apply error handling or could send new messages if applicable. The same applies to TLV etc.

### **6.4 Parsing of unknown messages**

Guidelines (MS 1), (MS 2) and (MS 3) allow the receiver to decode those parts of an unknown message that are necessary for the error handling.

### **6.5 Capability negotiation/indication**

Capability negotiation / indication shall be introduced in new versions of the stage 3 specification to negotiate / indicate all kinds of capabilities, e.g., support of options. It can take place when a context between nodes is set. There might be updates of capabilities from time to time and when capabilities change. Negotiation is necessary when an agreement between nodes is needed.

Capability indication can for example use a new function type to indicate the function types supported in addition to PV1 of release 1, Version 1.3.

Furthermore detailed information can be indicated using an existing function type with a new message or a new function type; it could go so far to report certification levels. A PV1 entity will react in a defined way.

Capability negotiation is particularly suitable for bigger evolution steps.

### **6.6 Enhancements without capability negotiation**

Smaller enhancing steps do not require capability negotiation/handling.

- 'Nice to have' information can be included by means of new TLVs without requiring comprehension (TC = 1).

The receiver will either understand the additional information and use it or discard the additional information.

- New procedures or procedures with essential new information can be used requiring comprehension (C = 1). If an error is reported, the procedure is abandoned or an alternative / modified procedure is initiated.

- Error codes in Failure Indication TLV: The Failure Indication TLV has a length of 1 byte (value part). Error Code 255 is foreseen for indication of the use of an error extension field. As the error handling specifies that additional bytes in a TLV are ignored by the receiver, this can be done in a safe way. If in future, an additional byte is used for an extended error code, this will cause older nodes to diagnose an "Unspecific Failure". For legacy nodes, see section 5.3 of this Annex, (LN 7).

## 6.7 Compatible version management

The introduction of new protocol versions (PV) should be avoided, if possible. However, a new PV might become necessary at a certain point in time, and the compatibility scheme prepares for that case.

It is assumed that

- (b) a network node should fully support<sup>1</sup> a given PV on all interfaces (R4, R6 and R8) for all function types or not support it at all;
- (c) normally, the protocol version on an interface between two given nodes is not changed, but it can be changed (e.g., when nodes are updated);
- (d) new protocol versions should introduce global procedures<sup>2</sup> by which the protocol version on an interface between two given nodes is negotiated; two nodes negotiate the version when establishing a context between the two nodes and update it from time to time;
- (e) a special case are the transparent messages, e.g., handover related messages in profile C. Here the version management should apply between the two end nodes terminating the transaction;
- (f) the version negotiation procedures must be introduced into the PV1 the latest when a new PV is introduced.

Due to the compatibility scheme, a version negotiation message can be sent to a PV1 entity without danger: For example, a new function type may be used and the PV1 entity will report an error.

The PV 1 entity indicates that it uses PV 1 by

- setting value 1 in the *Version 1 indicator* field which was earlier called the *Version* field
- setting the S bit in the *Flags* field to indicate it is not a legacy entity.

The legacy node is recognized by

- setting value 1 in the *Version 1 indicator* field which was earlier called the *Version* field
- not setting the S bit in the *Flags* field.

However, the receiver *PV1 only node* shall ignore the value of the *Version 1 indicator* field: as it does not support any higher version, it is bound to PV1; a higher PV entity should not send a higher version value to a PV 1 entity. In order to discourage bad version management schemes, the requirement is introduced that the receiver *PV1 only node* shall ignore the value of the *Version 1 indicator* field.

The latest when and if it is decided to introduce a new PV, the PV 1 Specification must be upgraded to include the version negotiation as an optional procedure. From this point in time on, an older PV1 entity will ignore/reject the version negotiation procedure, while a new PV1 entity (PV1+ entity) will run the version negotiation procedure.

A special case are the transparent messages, e.g., handover related messages in profile C. Here the version management should apply between the two end nodes terminating the transaction.

## 6.8 Legacy handling

**Legacy node discovery:** A legacy node is recognized by

- the fact that it sets value 1 in the *Version 1 indicator* field (which was earlier called the *Version* field);
- the fact that it does not set the S bit in the *Flags* field.

**Version negotiation:** A legacy node cannot perform version negotiation. The guidelines for legacy nodes imply that only messages indicating value one in the *Version 1 Indicator* field (earlier called *Version* field) should be sent to a legacy node.

---

<sup>1</sup> 'Full support' means full compliance to the protocol specification.

<sup>2</sup> A 'global' message is a message on R4, R6 or R8 not related to a specific mobile station. A 'global' procedure is a procedure on R4, R6 or R8 not related to a specific mobile station. It uses global messages.

1 **Introduction of new Function types:** The guidelines for legacy nodes imply that only function types defined in the  
2 supported pre-baseline version of stage 3 should be sent to a legacy node.

3 **Introduction of new Message types:** The guidelines for legacy nodes imply that only messages with message type  
4 defined (for the function type) in the supported pre-baseline version of stage 3 should be sent to a  
5 legacy node.

6 **Introduction of new TLV:** Addition of new TLV in a message with known Message type (for the function type) is  
7 possible, the reaction of the legacy node is to ignore the TLV and process the message.

8 **Introduction of new values in fields:** The guidelines for legacy nodes imply that unknown fields or unknown  
9 values within a field should not be set to a legacy node. However new information can be introduced  
10 by means of a new TLV.

11

---

## 7. Fundamental rules

This section describes fundamental rules for the compatible evolution of the ASN control protocol. The hooks and mechanisms for compatible evolution of this specification are based on the assumption that future evolution follows these rules.

### 7.1 Rules

Reference	Rule	Comments
R_VF	An ASN node SHALL support a protocol version of the ASN control protocol either fully - that means it SHALL be fully compliant to the protocol version on all (instances of) interfaces R4, R6 and R8 - or not support it at all.	

Note: It is assumed that when and if a new protocol version of the ASN control protocol is specified, rules on support of earlier protocol version of the ASN control protocol will be added to this annex.

