The ATM Forum

Technical Committee

PNNI 1.1 Errata Version 1.0

af-pnni-0205.000

January 2005

The ATM Forum
Worldwide Headquarters

The address of which can be found at: http://www.atmforum.com/contactfs1.html

Acknowledgements

The Control and Signaling working group was chaired by Mickey Spiegel. Thomas Cornély edited the document. The minutes at related working group meetings were recorded by Peter Roberts.

The following people made significant technical contributions to the PNNI 1.1 Errata Version 1.0 document:

William Cheng

Thomas Cornély

Robert Dianda

Peter Roberts

John Rutemiller

Ethan Mickey Spiegel

Table of Contents

1 INTRODUCTION **5**

1.1 SUMMARY LIST OF CORRECTIONS 5
1.2 SUMMARY LIST OF AMENDMENTS 5

2 CORRECTIONS TO PNNI 1.1 ..... **6**

COR_1 – UPDATED REFERENCES IN 5.14.9.5 6
COR_2 – USE OF CORRECT TERMINOLOGY IN CLEARING PROCEDURES 7
COR_3 – RESTRICTED TRANSIT HANDLING IN DESIGNATED TRANSIT LIST PROCEDURES 8
COR_4 – INCORRECT VARIABLE NAME IN DTL PSEUDOCODE 8
COR_5 – MODIFY PNNI CALL STATE RATHER THAN UNI CALL STATE IN SECTION 18.3.2.1 9

3 AMENDMENTS TO PNNI 1.1 ..... **10**

AM_1 – REVISED CRANKBACK TO NEXT HIGHER LEVEL 10
AM_2 – CLARIFICATION OF SVCC-BASED RCCS ESTABLISHMENT PROCEDURES 11

1 Introduction

This document identifies and specifies resolutions to items that are either considered to be “in error”, or can be “improved” in the ATM Forum Technical Committee PNNI 1.1 specification, af-pnni-0055.002 (April 2002).

Resolutions to these items are referred to as either “corrections” or “amendments”, respectively, in the rest of this document.

1.1 Summary List of Corrections

Item#	Title
Cor_1	Referencing non existent item e) in Sections 5.14.9.5.1 through 5.14.9.5.6
Cor_2	Use of Correct Terminology in Clearing Procedures
Cor_3	Restricted Transit Procedures
Cor_4	Incorrect variable name in DTL Pseudocode
Cor_5	Modify PNNI Call State rather than UNI Call State in Section 18.3.2.1.

1.2 Summary List of Amendments

Item#	Title
Am_1	Revised Crankback to Next Higher Level (Section 8.3.2.2.2).
Am_2	Clarification of SVCC-based RCCs establishment procedures.

2 Corrections to PNNI 1.1

Cor_1 – Updated References in 5.14.9.5

- *Modify Section 5.14.9.5.1 as follows:*

a) not applicable
b) if a nodal info IG appears multiple times in the designated PTSE, two implementations interpreting different of those IGs as valid could lead to scenarios where PGL election does not converge. The solution is to only consider the first appearing nodal info IG as valid and significant for state-significant computation purposes and ignore all following IGs.
c) if a nodal info IG does not appear in the designated PTSE at all, the PTSE and all further PTSEs issued by this node shall be ignored for the state-significant computations.
~~d) n/a.~~
~~e)~~d) is not possible due to the fact that this IG can appear only in a PTSE with ID 1, otherwise it is ignored.

- *Modify Section 5.14.9.5.2 as follows:*

a) if such an IG appears once or multiple times inside of a nodal info IG with cleared `I am leader'-Bit, such a IG must be ignored.

The following cases only apply to IGs where the `I am leader'-Bit is set and additionally the node has been determined as peer group leader:

b) if the binding appears multiple times, only the first thereof must be used for state-significant computation and all following ignored.
c) if no next higher level binding IG appears, then this node behaves as if the `I am leader' bit is not set (i.e., this node's ancestry is unknown at levels higher than that of the advertising node). Specifically, the nodal hierarchy list advertised by this node in Hellos to outside neighbors cannot include any information about the advertising node's parent node. This node also cannot originate calls on uplinks to upnodes at levels higher than that of the advertising node (see Section 5.13.1).
~~d) n/a.~~
~~e)~~d) equivalent to 5.14.9.5.1 ~~e~~d).

- *Modify Section 5.14.9.5.3 as follows:*

The following cases only apply to IGs where the `Nodal Representation'-Bit is set (i.e., the node is complex).

a) not applicable
b) if a nodal state parameters IG appears mutliple times inside of a single PTSE only the first one is used for any state-significant computations (even if the service category sets defined do not intersect). This includes the default spoke.
c) default spoke does not appear. Spokes of this node may not be used in state-significant computations unless exceptions are advertised for both directions, except for vi)/5.14.9.4. This does not affect iv)/5.14.9.4 (PGL election), v)/5.14.9.4 (computation of hierarchy to be advertised in outside hellos), or vi)/5.14.9.4 (reachability computation).
~~d) n/a.~~
~~e)~~d) if a nodal state parameters IG appears multiple times in different PTSEs (even if the service category sets defined do not intersect), only the one appearing in the PTSE with the lowest ID is used for state-significant computations.

If the node does not have the `Nodal Representation'-Bit set (i.e., the node is complex).

a) if nodal state IGs appear they are ignored for the sake of any state-significant computations
b) not applicable.
c) not applicable.
~~d) n/a.~~
~~e)~~d) analog a).

- *Modify Section 5.14.9.5.4 as follows:*

a) not applicable
b) if such IG appears mutliple times inside of a single PTSE only the first one is used for any state-significant computations (even if the service category sets defined do not intersect).
c) not applicable
~~d) n/a.~~
~~e)~~d) if a IG appears multiple times in different PTSEs (even if the service category sets defined do not intersect), only the one appearing in the PTSE with the lowest ID is used for state-significant computations. This rule also applies when a horizontal link IG and an uplink IG appear (necessarily in different PTSEs) with the same value for the local port ID.

- *Modify Section 5.14.9.5.6 as follows:*

a) not applicable
b) if multiple ULIAs appear in an uplink IG, only the first one is used for state-significant computations.
c) if an uplink IE does not have a ULIA associated with it, then the link can not be used for state-significant computations (e.g. link aggregation) and route computation.
~~d) n/a.~~
~~e)~~d) not applicable

Cor_2 – Use of Correct Terminology in Clearing Procedures

- *Modify the second and third paragraph of Section 6.5.3.3 "Clearing" as follows:*

The preceding side shall initiate clearing by: sending a RELEASE message, starting timer T308; ~~release~~ disconnecting the virtual channel and entering the Release Request state.

The succeeding side of the PNNI interface shall enter the Release Indication state upon receipt of a RELEASE message. On receipt of this message, the succeeding side shall ~~release~~ disconnect the virtual channel, and initiate procedures for clearing the network connection by informing the PNNI Call control entity. Once the virtual channel used for the call has been ~~released~~disconnected, the succeeding side shall: send a RELEASE COMPLETE message to the preceding side; release both the call reference and virtual channel (i.e. Connection identifier); and enter the Null state.

Cor_3 – Restricted Transit Handling in Designated Transit List Procedures

- *Modify the second paragraph of Section 7.2.2 "Procedures at an entry border node", clause c) as follows:*

  When the target is a node, if any of the border node's ancestors up to but not including the level of the target is a restricted transit node, then the call shall be cleared with cause #2 "no route to specified transit network" or cause #3 "no route to destination" and cranked back with a blocked transit type of "call or party has been blocked at the succeeding end of this interface" , with a crankback level subfield set as specified in Section 8.3.1.3, and crankback cause #129 "node is a restricted transit node".

- *Modify step 5)a in Section 7.5 "Pseudocode of DTL procedures" as follows:*

  5) determine a route across the current node (from 3) and through the current port (from 3 or 4) to the target (from 4).
     is the target a node?
     a. yes: is any of the border node's ancestors up to but not including the level of the target a restricted transit node ?
        a.1: yes: abort, an error has occurred.
        a.2: no: goto c.

Cor_4 – Incorrect Variable Name in DTL Pseudocode

- *Update step 2) in Section 7.5 "Pseudocode of DTL procedures" as follows:*

  The operations required to process the received stack of DTLs and to generate the new stack of DTLs are as follows:
  1) receive a PNNI SETUP or ADD PARTY message.
  2) extract a DTL stack from the SETUP or ADD PARTY message and initialize both OneAncestorIsRestrictedTransit and LowestLevelExitBorderIsRestrictedTransit flags to FALSE.

- *Update Annex S "Explicitly Routed Calls" in step 2) of the restated pseudocode of Section 7.5. Note: This change bar indicates a correction of the variable name. This should not be read as a change bar required by the Annex S pseudocode, rather an alignment with the corrected pseudocode of Section 7.5.*

  The operations required to process the received stack of DTLs and to generate the new stack of DTLs are as follows:
  1) receive a PNNI SETUP or ADD PARTY message.
  2) extract a DTL stack from the SETUP or ADD PARTY message and initialize both OneAncestorIsRestrictedTransit and LowestLevelExitBorderIsRestrictedTransit flags to FALSE.

Cor_5 – Modify PNNI Call State rather than UNI Call State in Section 18.3.2.1

- *Update Section 18.3.2.1 "Call State" as follows:*

18.3.2.1 Call State

~~*See Section A12.3.2.1 of the UNI Signalling 4.1 specification.*~~

Section 6.4.5.14 applies with the following changes:

- *The maximum length is modified from 5 to 3008 octets.*
- *Modify coding of call state information element by adding the following new octets:*

Bits

8	7	6	5	4	3	2	1	Octets
Call Reference List identifier								
0	0	0	0	0	0	0	1	6*
Call Reference List Length								6.1*
Call Reference List Length (continued)								6.2*
0/1 Call Reference Flag	Call Reference Value							6.3.1* (Note)
								6.3.2* (Note)
								6.3.3* (Note)

Note - Octet group 6.3 may be repeated up to 1000 times. Each occurrence indicates a call reference for which the call is in the state indicated in the call state value in octet 5.

Call Reference List Length (octets 6.1 and 6.2)

Length of the call reference list contents in octets, i.e. excluding the octets used for the call reference list length and the identifier.

Call Reference Flag (octet 6.3.1)

Bit 8	Meaning
0	The message is sent from the side that originated the call reference
1	The message is sent to the side that originated the call reference

Call Reference Value (octets 6.3.1, 6.3.2, and 6.3.3)

The Call reference value is a 23-bit integer (coded in binary) to uniquely identify a call.

3 Amendments to PNNI 1.1

Am_1 – Revised Crankback to Next Higher Level

- *Replace the second paragraph of Section 8.3.2.2.2 "Crankback to next higher level":*

If only nodes and/or links internal to the peer group were returned as blocked nodes or links in the Crankback information element of RELEASE, or ADD PARTY REJECT messages, then the logical node corresponding to the peer group should be listed as blocked. In this case, the blocked node ID should match the node ID indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message.

With:

If only nodes and/or links internal to the peer group were returned as blocked nodes or links in the Crankback information element of RELEASE, or ADD PARTY REJECT messages, then one of the following sets of actions should be taken:

a) If the entry border node does not attempt to determine if an alternate route exists using a different entry border node or egress link of the peer group then:

List the logical node corresponding to the peer group as blocked using blocked transit type #3 "blocked node". The blocked node ID shall be the node ID indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message.

This method minimizes call attempts and may result in more blocked calls. This method should not be used when there is sparse connectivity outside the peer group.

b) If the entry border node does attempt to determine if an alternate route exists using a different entry border node or egress link of the peer group then there are three options:

1. If the entry border node determines (e.g. by performing a path computation) that the call is likely to succeed if routed over the same ingress link, but over a different egress link from the peer group, then:

   List the logical link corresponding to the egress link from the peer group as blocked using blocked transit type #4 "blocked link". Specifically, the blocked link's preceding node ID and port ID should be set to the values indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message, and the blocked link's succeeding node ID should be set to the node ID of the target of the path calculation, as determined from the received DTL stack in the procedures of Section 7.2.2. However, if the target of the path calculation is not a node, then the blocked link's succeeding node ID is set to all zeros.

2. If the entry border node determines that the call is likely to succeed if routed over a different entry border node of the peer group or if the entry border node does not determine whether the call is likely to succeed if routed over a different entry border node of the peer group, then:

   List the logical link corresponding to the ingress link to the peer group as blocked using blocked transit type #2 "call or party has been blocked at the succeeding end of this interface".

3. If the entry border node determines that the call is unlikely to succeed if a different entry border node or different egress link of the peer group is used, then:

   List the logical node corresponding to the peer group as blocked using blocked transit type #3 "blocked node". The blocked node ID shall be the node ID indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message.

~~a)List the logical node corresponding to the peer group as blocked using blocked transit type #3 "blocked node". In this case, the blocked node ID shall be the node ID indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message.~~

~~This case should be used when the border node does not attempt or cannot determine if an alternate route exists using a different entry border node or egress link. The implication is that even if an alternate path across this peer group does exist, it will not be considered in subsequent alternate routing attempts for this connection.~~

~~b)List the logical link corresponding to the egress link from the peer group as blocked using blocked transit type #4 "blocked link". Specifically, the blocked link's preceding node ID and port ID should be set to the values indicated by the current transit pointer in the top DTL on the stack in the received SETUP or ADD PARTY message, and the blocked link's succeeding node ID should be set to the node ID of the target of the path calculation, as determined from the received DTL stack in the procedures of Section 7.2.2. However, if the target of the path calculation is not a node, then the blocked link's succeeding node ID is set to all zeros.~~

~~This case could be used when the entry border node can determine that the call can be successfully routed (e.g., by performing a path computation to the destination) over the same ingress link, but over a different egress link from the peer group. The implication is that even if an alternate path does exist using the egress link, it will not be considered in subsequent alternate routing attempts for this connection.~~

~~c)List the logical link corresponding to the ingress link to the peer group as blocked using blocked transit type #2 "call or party has been blocked at the succeeding end of this interface".~~

~~This case could be used when the entry border node can determine that the call cannot be successfully routed (e.g., by performing a path computation to the destination) over the same ingress link.~~

Am_2 – Clarification of SVCC-based RCCs establishment procedures

- *Modify the third paragraph of Section 5.5.6.1 "SVCC Establishment" as follows:*

Each LGN finds out about the address of neighboring LGNs from the uplinks announced by border nodes in the peer groups that the LGN represents. Specifically, the called party address used must be the ATM End System Address for the neighboring LGN at the other end of the uplink which the SVCC is intended to cross, as advertised by the border node. As well, the calling party address used must be the ATM End System Address of the SVCC's originating LGN as advertised in the that LGN's Nodal IG. For future flexibility, note that there is no requirement that different border nodes report the same ATM End System Address for the neighboring LGN.

- *Modify paragraph 3 of Section 5.5.6.3 "Detailed Mechanisms for Establishment and Maintenance of SVCCs" as follows:*

One of the following two sets of procedures shall apply in order to prevent multiple active (in the 2-Way Inside state) SVCC-based RCCs to the same neighboring LGN:

i) If a SETUP is received in which the called party address matches the neighbor's address of an existing SVCC-based RCC, then the SETUP is accepted and the existing SVCC-based RCC is cleared with cause number 16 "normal call clearing".

ii) ThisLGN must accept all calls as the called party for a SVCC-based RCC, since the identity of the neighboring LGN is not yet known. When ThisLGN is the called party, the identity of the neighboring LGN is determined from SVCC-based RCC Hellos received over that SVCC. When ThisLGN is the calling party for a SVCC, the neighboring LGN's identity is determined from the uplink advertisement that triggered establishment of the SVCC.

If ThisLGN detects the presence of two or more SVCCs to the same neighboring LGN then:

B.1 If ThisLGN's node ID is numerically larger than the neighboring LGNs node ID, then choose one SVCC to leave open. Close the other SVCC(s) with cause number 16 "normal call clearing".

B.2 If ThisLGN's node ID is numerically smaller than the neighboring LGN's node ID, then ThisLGN shall continue to use all SVCCs. Hello packets must be transmitted on all SVCCs. Only one of the SVCC-based RCC Hello state machines to the neighboring LGN should be allowed to reach the 2-Way Inside state (see Section 5.6.3.1 item 3 for details).

- *Modify the list item 3 of Section 5.6.3.1 "SVCC-Based RCC Hello Protocol" as follows:*

3 For the LGN that is the calling party for the SVCC-based RCC, t~~T~~he node ID in received Hellos must be equal to the value in the corresponding uplink PTSE. If it is not equal the event HelloMismatchReceived is triggered.

~~For the LGN that is the calling party for the SVCC-based RCC, that uplink PTSE is necessarily received before the SVCC is initiated.~~ For the LGN that is the called party, ~~there is a race condition between the arrival of the call setup from the neighbor and the uplink PTSE. If the called-party LGN receives a SETUP from a node which it has yet to recognize as a neighbor, the called-party LGN must accept the call, but ignore any hellos until an uplink PTSE is received indicating that node as a neighbor. The~~ the binding between the uplink PTSE and the SVCC-based RCC is the node ID in Hellos received over the SVCC-based RCC and the Upnode ID in the uplink PTSE. If the Hellos indicate a neighbor node ID that does not match the Upnode ID in any received uplink PTSE, then the Hellos shall be ignored. If step ii of section 5.5.6.3 is used then, if

- the Hellos indicate a neighbor node ID that is the same as the neighbor node ID received in Hellos from another SVCC-based RCC,
- that SVCC-based RCC has reached the 2-Way Inside state, and
- the conditions for the event 2-WayInsideReceived as defined in Section 5.6.2.1.3 are met,

then the event 1-WayInsideReceived is triggered instead (ensuring that only one of the SVCC-based RCCs to that neighbor reaches the 2-Way Inside state).