The ATM Forum

Technical Committee

PNNI Routing Resynchronization Control, Version 1.0

af-cs-0201.000

June 2004

© 2004 by The ATM Forum. This specification/document may be reproduced and distributed in whole, but (except as provided in the next sentence) not in part, for internal and informational use only and not for commercial distribution. Notwithstanding the foregoing sentence, any protocol implementation conformance statements (PICS) or implementation conformance statements (ICS) contained in this specification/document may be separately reproduced and distributed provided that it is reproduced and distributed in whole, but not in part, for uses other than commercial distribution. All other rights reserved. Except as expressly stated in this notice, no part of this specification/document may be reproduced or transmitted in any form or by any means, or stored in any information storage and retrieval system, without the prior written permission of The ATM Forum.

The information in this publication is believed to be accurate as of its publication date. Such information is subject to change without notice and The ATM Forum is not responsible for any errors. The ATM Forum does not assume any responsibility to update or correct any information in this publication. Notwithstanding anything to the contrary, neither The ATM Forum nor the publisher make any representation or warranty, expressed or implied, concerning the completeness, accuracy, or applicability of any information contained in this publication. No liability of any kind shall be assumed by The ATM Forum or the publisher as a result of reliance upon any information contained in this publication.
The receipt or any use of this document or its contents does not in any way create by implication or otherwise:

- Any express or implied license or right to or under any ATM Forum member company's patent, copyright, trademark or trade secret rights which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor
- Any warranty or representation that any ATM Forum member companies will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor
- Any form of relationship between any ATM Forum member companies and the recipient or user of this document.

Implementation or use of specific ATM standards or recommendations and ATM Forum specifications will be voluntary, and no company shall agree or be obliged to implement them by virtue of participation in The ATM Forum.

The ATM Forum is a non-profit international organization accelerating industry cooperation on ATM technology. The ATM Forum does not, expressly or otherwise, endorse or promote any specific products or services.

NOTE: The user's attention is called to the possibility that implementation of the ATM interoperability specification contained herein may require use of an invention covered by patent rights held by ATM Forum Member companies or others. By publication of this ATM interoperability specification, no position is taken by The ATM Forum with respect to validity of any patent claims or of any patent rights related thereto or the ability to obtain the license to use such rights. ATM Forum Member companies agree to grant licenses under the relevant patents they own on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. For additional information contact:
The ATM Forum
Worldwide Headquarters
2570 West El Camino Real, Suite 304
Mountain View, CA 94040-1313
Tel: +1-650-949-6700
Fax: +1-650-949-6705

Preface

During preparation of this addendum, the Control Signaling Working Group was chaired by Gert Oster and Mickey Spiegel. The minutes at related working group meetings were recorded by Dave Paw and Peter Roberts. The editors of this addendum were Gerald Ash and Mostafa Hashem Sherif. The editors would like to thank the following contributors for their help with this addendum as well as all participants of the Control Signaling working group and OSPF/ISIS email list:

Sirak Bahlbi

Gagan Choudhury

Thomas Cornely

Bob Dianda

Jeff Han

Dave Katz

Ghassem Koleyni

Vishwas Manral

Shawn McAllister

Mahmood Noorchashm

Jeff Parker

Carl Rajsic

Peter Roberts

John Rutemiller

Vera Sapozhnikova

Mickey Spiegel

Russ White

Alex Zinin

The editors would also like to thank the many additional contributors who made indirect but significant contributions to this addendum: Margaret Chiosi, Ragu Aswatnarayan, Enrique Cuevas, Tom Helstern, Steve Holmgren, Mike Shapiro, Anurag Maunder, Marty Albright, Sohel Khan, and Greg Wetzel.

This specification uses three levels for indicating the degree of compliance necessary for specific functions, procedures, or coding. They are indicated by the use of key words as follows:

- **Requirement:** "Shall" indicates a required function, procedure, or coding necessary for compliance. The word "shall" used in text indicates a conditional requirement when the operation described is dependent on whether or not an objective or option is chosen.
- **Objective:** "Should" indicates an objective which is not required for compliance, but which is considered desirable.
- **Option:** "May" indicates an optional operation without implying a desirability of one operation over another. That is, it identifies an operation that is allowed while still maintaining compliance.

Table of Contents

1. Introduction ***6***

1.1 SCOPE [NORMATIVE] **ERROR! BOOKMARK NOT DEFINED.**

1.2 BACKGROUND AND OVERVIEW [INFORMATIVE] 6

1.3 TERMINOLOGY [NORMATIVE] 7

2. Mechanisms for PNNI Routing Resynchronization Control [Informative] ***8***

2.1 DATABASE BACKUP AND PNNI GRACEFUL RESTART 8

2.2 DATABASE RESYNCHRONIZATION 9

3. Procedures [Normative] ***11***

3.1 PNNI PROTOCOL CHANGES REQUIRED FOR DATABASE BACKUP AND PNNI GRACEFUL RESTART 11

3.1.1 Procedures for Database Backup *11*

3.1.2 Procedures for PNNI Graceful Restart *11*

3.1.2.1 Entering PNNI Graceful Restart 11

3.1.2.2 Exiting PNNI Graceful Restart 12

3.2 PNNI PROTOCOL CHANGES REQUIRED FOR DATABASE RESYNCHRONIZATION 12

3.3 ARCHITECTURAL VARIABLES 30

4. References ***31***

4.1 REFERENCES [NORMATIVE] 31

4.2 REFERENCES [INFORMATIVE] 31

Annex A. Protocol Implementation Conformance Statement (PICS) for PNNI Routing Resynchronization Control [Normative] ***32***

A.1 INTRODUCTION 32

A.1.1 Scope *32*

A.1.2 Normative References *32*

A.1.3 Definitions *32*

A.1.4 Acronyms *32*

A.1.5 Conformance *33*

A.2 IDENTIFICATION OF THE IMPLEMENTATION 33

A.2.1 Date of Statement *33*

A.2.2 Implementation Under Test (IUT) Identification *33*

A.2.3 System Under Test (SUT) Identification *33*

A.2.4 Product Supplier *34*

A.2.5 Client *34*

A.2.6 PICS Contact Person *34*

A.3 PICS PROFORMA 35

A.3.1 Global statement of conformance *35*

A.3.2 Instructions for Completing the PICS Proforma *35*

A.4 PICS FOR THE SUPPORT OF PNNI ROUTING RESYNCHRONIZATION CONTROL AT THE PNNI INTERFACE 36

A.4.1 Major Capability at PNNI (MCP) *36*

A.4.2 Routing Procedures at the PNNI (RPP) *36*

1. Introduction

1.1 Scope [Normative]

This document is an optional Addendum to PNNI 1.1 [af-pnni-0055.002]. A device supporting PNNI 1.0 [af-pnni-0055.000] may implement the functionality defined in this addendum by treating this addendum as if it were an optional addendum to PNNI 1.0 and PNNI 1.0 Errata and PICS [af-pnni-0081.000].

The document contains the specification for the support of PNNI Routing Resynchronization Control. Based on overload and failure experience with link-state protocols, this addendum identifies means to enable PNNI routing protocols to gracefully recover from loss of topology database information.

The proposed mechanisms in this specification include the following:
- database backup and PNNI graceful restart
- database resynchronization

Unless otherwise specified by SHALL terminology, the congestion mechanisms described are suggested guidelines that can be adjusted as needed by specific implementations. This specification provides guidelines for when to initiate these mechanisms, and procedures for database backup, PNNI graceful restart and database resynchronization.

1.2 Background and Overview [Informative]

There is evidence based on previous failures that link-state protocols such as PNNI cannot recover from large failures that can result in widespread loss of topology database information. The problem is aggravated when the number of nodes in a peer group is large, which can then lead to an overload in the flooding of topology database information.

Link-state (LS) protocols typically use topology-state update (TSU) mechanisms to build the topology database at each node, typically conveying the topology status through flooding of TSU messages containing link, node, and reachable-address information between nodes. In PNNI, such mechanisms use the PNNI topology state element (PTSE); in OSPF, they use the link state advertisement (LSA); in frame-relay and proprietary-routing networks, they may use other TSU mechanisms to exchange topology status information to build the topology database at each node. Topology information in PNNI is distributed in PTSEs, which are encapsulated in PNNI topology state packets (PTSPs) and periodically flooded to other nodes in the domain through all available links. In some instances of network overload, failure, and/or congestion, redundancy of the flooding mechanism can overwhelm the routing control processors and bring the network down. In this Section a generic term – TSU – is used to cover LS protocols in general. PTSE and PTSP are specifically referred to in Section 2 and thereafter.

Earlier papers identified issues of congestion control and failure recovery for link-state protocol networks, such as PNNI [choudhury1, choudhury2, choudhury3, pappalardo1, pappalardo2]. The capabilities presented in this specification allow PNNI to gracefully recovery from loss of topology database information. The automatic mechanisms proposed in this specification include:

- database backup and PNNI graceful restart for recovery from loss of topology database information and maintaining call establishment capabilities, and
- database resynchronization

In Section 2, protocol mechanisms for database backup, PNNI graceful restart, and database resynchronization are described. In Section 3 procedures for implementing these mechanisms are discussed.

Congestion control mechanisms are defined in [af-cs-0200.000], which detect and notify congestion states between nodes, reduce the rate of flooding of PTSEs, maintain link adjacencies, and prioritize treatment of Hello and PTSE Acknowledgement packets.

The following examples illustrate the need for graceful recovery mechanisms, where we discuss failure experience in which all topology database information has been lost in all nodes in a large network, and the link-state protocol has been unable to recover. In the failure of a large frame relay network [att], an initial procedural error triggered two undetected software bugs, leading to a huge overload of control messages in the network. The result of this control overload was the loss of all topology information, which the LS protocol then attempted to recover using the usual Hello and LS updates. However, the LS protocol was overwhelmed and unable to recover, and manual means had to be used to restart the network after a long outage.

A more recent occurrence in a large ATM network resulted in an overload of TSUs, and a lengthy network outage [att, pappalardo1, pappalardo2]. Manual procedures were put in place to reduce TSU flooding, which worked to stabilize the network. It is desirable that such TSU flooding reduction be automatic under overload. In general, there have been a number of major outages reported by most major carriers, and routing protocol issues have generally been involved. Other relevant LS-network failures are reported in [cholewka, jander].

Various networks employing LS protocols use various control messages and mechanisms to update the LS database, not necessarily PTSEs, LSAs, or flooding mechanisms. Based on experience, however, the LS protocols including PNNI are found to be vulnerable to loss of topology information, such as occurred in the scenario described above [att], control overload to resynchronize databases, and other failure/overload scenarios which make such networks more vulnerable in the absence of adequate protection mechanisms. Hence we are addressing a generic problem of LS protocols across a variety of implementations, and the basic problem is prevalent in LS protocol networks employing frame-relay, ATM, and IP based technologies. However the solutions given are for PNNI networks alone.

In summary, service providers have suffered a few massive failures of operational networks due to control overloads of LS protocols ('PNNI', ‘OSPF’, etc.). In the instances cited, the LS protocol overwhelmed the network with a control load 'storm' ('PTSE/LSA overload'), which brought the network down, and then prevented its recovery. Fortunately such failures are very rare; however, 'rare' for such events is unacceptable, 'never' is the goal. Such failures are not always the fault of the service provider operation or the vendor/equipment implementation. They are in some cases due to shortcomings in the link-state protocols themselves. The proposals in the specification will provide recovery mechanisms in case such events occur.

1.3 Terminology [Normative]

CPU	Central Processing Unit
CSI	Congestion State Indication
FSM	Finite State Machine
IG	Information Group
IGP	Interior Gateway Protocol
IGR	Initiate Graceful Restart
LGN	Logical Group Node
LS	Link State
LSA	Link State Advertisement
OSPF	Open Shortest Path First
PGL	Peer Group Leader
PNNI	Private Network-Network Interface
PTSE	PNNI Topology State Element
PTSP	PNNI Topology State Packet
RCC	Routing Control Channel
RCI	Resynch Congestion Indication
TSU	Topology State Update
VC	Virtual Circuit
VCI	Virtual Circuit Identifier
VPI	Virtual Path Identifier

2. Mechanisms for PNNI Routing Resynchronization Control [Informative]

In the following section, we describe at a general level the following mechanisms for inclusion into PNNI:

a. database backup and PNNI graceful restart which allow a topology database to be automatically recovered from loss based on local backup mechanisms, so that calls continue to originate, terminate and transit a node uninterrupted during a failure on the node, and
b. database resynchronization, which allows a node to recover database information gracefully from local faults on the node.

Procedures for these mechanisms are described in Section 3.

2.1 Database Backup and PNNI Graceful Restart

There is a need to gracefully recover from loss of topology database information while maintaining the ability to establish new calls during a failure. This need is illustrated in Section 1, where we discuss failure experience in which all topology database information is lost in all nodes in a large network, the link-state protocol is unable to recover, and calls are unable to be established. Database backup and PNNI graceful restart procedures address these problems.

Prior to database backup and PNNI graceful restart, LS protocols intentionally route around a restarting node while it re-creates local reachability with its neighboring nodes and relearns the topology database from those neighbors. Avoiding the restarting node during call establishment is accomplished by either:
a. having the node's neighbors reissue their PTSEs, omitting links to the restarting node, or
b. by sending calls towards the restarting node and allowing PNNI crankback to determine alternate paths around the node.

These actions add to routing control plane instability since unnecessary topology changes are incurred in the network. In addition, crankback processing may become a burden on nodes attempting to re-establish calls. However, if the node is able to:
a. keep its local state information during the restart (so that RCCs and associated FSMs states do not change, new PGLs are not elected, and self-originated information and associated states are not lost),
b. keep its routing tables consistent during the restart, and
c. recover its topology database from a database backup or retrieve it by performing re-synchronizations with neighbors (without dropping links),

then, it would be possible to maintain new call establishment without crankbacks, while not causing unnecessary topology changes.

To aid PNNI graceful restart procedures, database backup and database resynchronization are applied, wherein the database is either:
a. backed up on the local node, recovered following a database failure, and subsequently resynchronized with topology information throughout the network, or
b. retrieved from local neighbors during a database resynchronization.

It is common practice in large networks to provide a method of locally backing up the LS database information. Then in the case of a failure condition in which database information is lost, the backup database is accessed and used to reinitialize the LS database. The concept of a database backup is not new, however, it is new in the context of using database backup together with LS protocols. Database backup has worked in practice for many years in large-scale distributed, dynamic routing networks [ash]. The intuition behind the database backup concept is simple: for example, if a node loses its whole database, it is better to reinitialize with a local copy, which can be expected to have a large percentage of correct, up-to-date information.

With local database backup, a node provides a local, primary, nonvolatile memory backup of all PTSE/PTSPs, routing tables, and states of interfaces. Restoration of data from the local backup memory, once initiated, is completed quickly. A

mechanism to recover any lost updates of database information, that is, those updates lost during the interval from the time of failure to the recovery time, is needed and is provided by database resynchronization procedures as outlined in Section 2.2. Database backup and resynchronization provide a definite advantage for improving convergence times and reducing control messages between neighbors during recovery, and is the cornerstone to PNNI graceful restart.

Mechanisms have been implemented for graceful restart within LS protocols [moy1, moy2, zinin1], however, these mechanisms cannot directly be applied to PNNI without complicated procedures being added to handle PNNI specific capabilities. A specific mechanism for PNNI is needed, although the motivation to maintain routing stability is the same. The intent of "PNNI graceful restart" is to maintain call establishment capability of the node while the node's routing control software is restarted and to insure the node can gracefully retrieve necessary topology database information during the restart.

PNNI graceful restart is triggered by local node failure, or other causes. The actions taken by the restarting node during the restart are a) restores database information from a locally stored backup of protocol states and topology information required for its protocol functions to remain unchanged during the restart, b) starts a graceful restart timer, c) resynchronizes database information from neighboring peers, and d) recomputes routing tables after the database has been resynchronized. As a result of restoring its database from the database backup, the restarting node's state as viewed by all other nodes in the network is unchanged, and the restarting node and other nodes in the network continue to operate as before the restart. In particular, calls continue to be established across the restarting node. During the graceful restart interval, the neighbor nodes, neighbor LGNs, entry border nodes, and PGLs of the restarting node are unaware that the node is restarting since nothing is flooded to indicate that. All protocol actions proceed as normal during the restart, such as link states (2-way-Inside, Common-Outside, 2-Way, Full, etc.), PGL FSM states, self-originated PTSE information and other required information needed by the restarting node to allow it to maintain all states as before the failure. This allows other nodes in the network to continue operating as if the node was not restarting, even though it is.

The restarting node then fully recovers its topology database information by performing database resynchronizations with all neighbor nodes and associated neighbor LGNs (if the restarting node was the PGL of a PG). To perform database resynchronizations, the restarting node requires some help from its neighbors (also known as helper nodes) in that these helper nodes must allow the restarting node to resynchronize its databases with them. The helper nodes need not know anything more about why it is performing a database resynchronization with the restarting node. The restarting node must perform a database resynchronization with all neighbor nodes and LGNs within a specified graceful restart time interval. If problems are encountered such that the graceful restart is not completed within the timeout period, the process automatically reverts back to a standard PNNI restart.

A general mechanism for database backup and PNNI graceful restart is described in Section 3.1.

2.2 Database Resynchronization

PNNI topology database synchronization is achieved via two methods,

a. initial topology database synchronization when a node is first connected to the network, and
b. asynchronous flooding, which ensures continuous topology database synchronization in presence of topology changes after the initial procedure is completed.

It is sometimes necessary for PNNI nodes to resynchronize their topology databases, such as under failure when database information is lost. One of the reasons for the addition of the database resynchronization procedures is to allow for the recovery of the topology information when a failure in the transfer of PTSEs is detected due to flooding congestion. These failures can be caused by flooding storms during network duress. For example, a limit on the size of the retransmission list can cause the loss of PTSEs that need to be flooded to a neighbor when those lists become congested.
Since these conditions also affect and potentially amplify a node's inability to successfully complete a database resynchronization, the resynchronization should only be attempted when it determined "safe to do so". For this reason, the database resynchronization should be delayed to give the node or the neighboring peer time to reduce congestion levels first without adding to the problem. Delaying the database resynchronization requires the monitoring of all congestion conditions and then notifying this condition and the node's ability to participate in a database resynchronization with its neighboring peers. When a node detects congestion that could affect the successful completion

of a database resynchronization, it indicates this congestion to its neighboring peers. When a node receives an indication of congestion from its neighboring peer or is actively indicating congestion to its neighboring peer, it does not initiate a database resynchronization until the congestion has abated.

"OSPF Out of band topology database resynchronization" [zinin2] describes how routers can reestablish adjacencies after a reload, such as with database backup, described above. Database resynchronization between neighbors is also helpful during PNNI graceful restart. See Section 2.1 for more information about PNNI graceful restart. When a node fails and attempts to recover, it needs to relearn the topology state database. If the database was backed up on the node, then some information may be stale and needs to be updated. If the database was not backed up on the node, then it needs to relearn the topology to build new routing tables. To update or relearn the topology, the node and possibly LGNs (if this node is elected PGL) performs a database resynchronization with each neighbor it is adjacent to.

A general mechanism for database resynchronization used by a node to recover from local faults on the node and during PNNI graceful restart is described in Section 3.2.

3. Procedures [Normative]

3.1 PNNI Protocol Changes Required for Database Backup and PNNI Graceful Restart

3.1.1 Procedures for Database Backup

Prior to restarting, a node SHALL provide a local, non-volatile memory backup as follows:

1. the database backup SHALL meet requirements specified in [GR-472-CORE]
2. the database backup SHALL contain:
   a. up to date routing tables,
   b. SVCC-based RCC binding information, if any,
   c. Hello FSM data structures on the node and associated LGNs running on the node,
   d. neighboring peer FSM data structures on the node and associated LGNs running on the node,
   e. self-originated PTSE information,
   f. PGL election FSM data structure on the node and associated LGNs running on the node,,
   g. any other relevant state information (e.g. peer group reachability information, aggregation information, nodal state parameters for complex node implementations, etc.) such that the state of the node and other nodes in the network are not affected during the restart
3. the database backup SHOULD contain all non-self-originated PTSEs.
4. all items SHALL be backed up at least every Tbackup seconds.

3.1.2 Procedures for PNNI Graceful Restart

Local node failure is defined as the state where the database and routing control plane states and processing is lost. The procedures defined in this Section enable the locally failed node to recover through a) database backup, which restores all the significant state information and PTSE database information required for the node to function as before the local node failure, and b) PTSE database resynchronization occurs with each neighboring peer to retrieve PTSEs that may have been lost during the failure. During the recovery, the restarting node can establish and transit new calls, and other nodes in the network can route calls through and to the restarting node.

The graceful restart mechanisms SHOULD be used for unplanned outages, such as node failure, control overload, or crash of a node's control software. The processes of entering PNNI graceful restart, and of exiting graceful restart are covered in the following sections.

3.1.2.1 Entering PNNI Graceful Restart

A node (call it Node X) SHALL initiate PNNI graceful restart automatically after failure and recovery by starting the GracefulRestartTimer timer with GracefulRestartInterval and perform the steps outlined below. During the PNNI graceful restart period, the restarting node:

1. SHALL hold off on sending PNNI routing packets until the corresponding FSM states are restored.
2. SHALL restore data from the local backup memory. This restores all database information, including:
   a. up to date routing tables that can be reused to establish calls during the restart,
   b. SVCC-based RCC binding information, if any, such that the SVCC-based RCC is maintained across the restart.
   c. Hello FSM data structures on the node and associated LGNs running on the node,
   d. neighboring peer FSM data structures on the node and associated LGNs running on the node, adjusting the states as follows:
      - Negotiating, Exchanging, Loading SHALL become Negotiating
      - Full, Full(Resynch Allowed), Negotiating in Full, Exchanging in Full, Loading in Full SHALL become Full, even though some PTSE information in the database may be missing.
   e. self-originated PTSE information, such that self-originated PTSEs can be retained during the restart,

f. PGL election FSM data structure on the node and associated LGNs running on the node, such that elected PGL information is not lost,
g. any other relevant state information (e.g. peer group reachability information, aggregation information, nodal state parameters for complex node implementations, etc.) such that the state of the node and other nodes in the network are not affected during the restart

This step SHALL be completed within the hello inactivity interval seconds. Note that the Hello related variables should be configured to sufficiently large values so that 75% of the inactivity times (for lowest level nodes, HelloInterval times InactivityFactor) is more than the amount of time it takes to restore this backed up information.

3. SHALL restore any non-self-originated PTSEs contained in the local backup memory.
4. SHALL take the restored state information into account when sending PNNI routing packets.
5. MAY originate a new Nodal IG with the CSI bits indicating either low or high congestion. This can be used if the restarting node is not as capable at processing PNNI protocol events while performing PNNI graceful restart and would like the other nodes in the network to react as if it were congested, thereby providing additional processing for database resynchronizations during graceful restart.
6. SHALL follow the procedures in Section 3.2 to recover any missing PTSE information by performing a resynchronization with each neighboring peer. In order to avoid causing congestion on the node, the number of adjacencies resynchronizing simultaneously SHALL be limited to a maximum of Nmaxresync, as defined in Section 3.1 of [af-cs-0200.000].
7. SHALL set the hello interval advertised to neighbors in Hello packets to StressInactivityFactorRestart times HelloInterval (which causes the neighbor's inactivity time to change), set the inactivity factor to StressInactivityFactorRestart times InactivityFactor (which causes this node's inactivity time to change), and set the horizontal link inactivity time to StressInactivityFactorRestart times HorizontalLinkInactivityTime. The restarting node SHALL NOT change its HelloInterval in the hello data structure (the frequency of sending hellos to neighbors does not change).

3.1.2.2 Exiting PNNI Graceful Restart

A restarting node SHALL exit PNNI graceful restart when any of the following occurs:

1. The restarting node has performed a database resynchronization with all of its neighbor nodes and neighbor LGNs.
2. The GracefulRestartTimer expires.

The following actions SHALL be performed by the restarting node when exiting PNNI graceful restart:

1. recompute routing tables using the information currently in the PTSE topology database.
2. stop the GracefulRestartTimer if it is still running.
3. perform database synchronizations starting from the Negotiating state on any neighbor nodes and neighbor LGNs that were not resynchronized as described in the steps above.

3.2 PNNI Protocol Changes Required for Database Resynchronization

The following Sections in the PNNI specification need to be modified as follows (note that the change marks in the following Sections are with respect to the PNNI 1.1 specification):

Section 5.6.3.2, second paragraph

On receipt of Hellos received from the neighboring peer LGN, the LGN Horizontal Link Extension information group is only processed if the SVCC-based RCC Hello State Machine is in 2-Way Inside and the corresponding neighboring peer state machine is in Full, Full (Resynch Allowed), Negotiating In Full, Exchanging In Full, or Loading In Full state~~s~~.

Section 5.6.3.2.4,

Hlhp2

Action: Set the Remote Port ID in the LGN horizontal link hello data structure to the Port ID listed in the entry for the Aggregation Token in the received LGN Horizontal Link Extension IG. Trigger an advertisement of the horizontal link in a new instance of a horizontal link PTSE originated by this node, provided the corresponding Neighboring Peer State machine is in state~~s~~ Full, Full (Resynch Allowed), Negotiating In Full, Exchanging In Full, or Loading In Full.

Hlhp3

Action: Trigger an advertisement of the horizontal link in a new instance of a horizontal link PTSE originated by this node, provided the corresponding Neighboring Peer State machine is in state~~s~~ Full, Full (Resynch Allowed), Negotiating In Full, Exchanging In Full, or Loading In Full.

Section 5.6.3.3, paragraph 3

Additionally, there are two inactivity timers. The Inactivity timer associated with the SVCC operates exactly as in the normal Hello protocol. The Horizontal Link Inactivity Timer is either started if it is not already active or otherwise restarted each time a non-empty LGN Horizontal Link Extension IG is processed, since this IG describes all horizontal links to this neighbor. Note that LGN Horizontal Link Extension IGs are processed only when the SVCC-based RCC Hello protocol is in the 2-WayInside state and the corresponding neighboring peer state machine is in the Full, Full (Resynch Allowed), Negotiating In Full, Exchanging In Full, or Loading In Full state~~s~~.

Section 5.7, fourth paragraph

When PTSPs, PTSE Acknowledgment packets, Database Summary packets, or PTSE Request packets are transmitted, any of the links between the neighboring peers that is in the Hello state 2-WayInside may be used. Successive packets may be sent on different links, without any harmful effects on the distribution and maintenance of PNNI routing information. Links between lowest-level neighboring peers may only be advertised in PTSEs when the neighboring peer state machine is in Full, Full (Resynch Allowed), Negotiating In Full, Exchange In Full or Loading In Full state~~s~~. ~~For the case of neighboring lowest level peers connected by physical links and VPCs, c~~Changes into any of these states from other states or out of these states to other states ~~the Full state~~ will cause new instances of one or more of this node's PTSEs to be originated or flushed.

..... and add after last paragraph:

In the case of a node wishing to resynchronize with a neighboring peer to recover from a failure local to the node, the node sends and receives Database Summary, PTSE Request and PTSP packets while the peer data structure is in the Full(Resynch Allowed), Negotiating In Full, Exchange In Full and Loading In Full states and the PTSE advertising the horizontal link to the neighboring peer is maintained. Database resynchronization procedures are very similar to database synchronization procedures when the neighboring peer is exchanging databases for the first time.

The types of local failure that could trigger a database resynchronization can include conditions where data are known to be lost due to congestion. As this congestion may potentially amplify a node's inability to successfully complete a database resynchronization, the resynchronization should only be attempted when it is determined that it is "safe to do so". When a node detects congestion is occurring that could affect the successful completion of a database resynchronization, it shall indicate this congestion to some or all of its neighboring peers by setting a flag in the PNNI Packet Header of all subsequent PTSPs or PTSE Acknowledgement packets sent to those neighbors. The flag is only a local flag used between two neighboring peers to indicate congestion and in the case of PTSPs is not flooded to the entire network. When a node receives an indication that its neighboring peer is experiencing congestion or is actively indicating to its neighboring peer that it itself is currently congested, it will not initiate a database resynchronization until the neighboring peer indicates the congestion has abated and it itself stops indicating congestion to that neighbor. A timer is used to perform periodic checks of the congestion state if the database resynchronization is delayed due to congestion. The value used to indicate congestion is the same value used for reserved bits. In this way, a node that does not support the database resynchronization functionality will always maintain this value as 'congested' and the database resynchronization shall never be requested by the neighboring peer.

Section 5.7.1, add new neighboring peer data structure items and modify last one

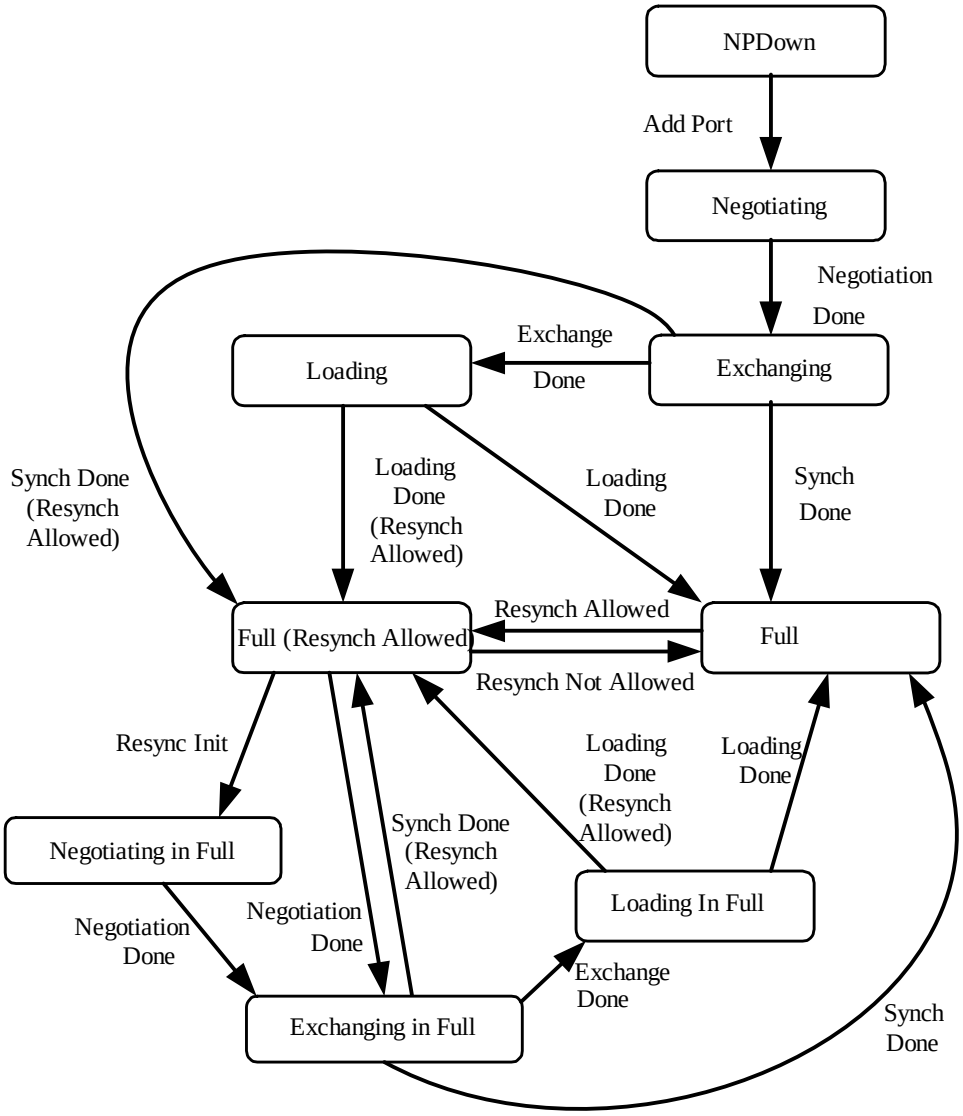
ResynchRetryInterval
The amount of time, in seconds, to delay when a database resynchronization is requested but cannot be attempted due to congestion as indicated by the Neighboring Peer Congestion Status
Resynch Retry Timer
When this timer expires, the node shall either initiate a Database Resynchronization or reschedule this timer depending on the local and neighboring peer congestion state.
Neighboring Peer Congestion Status
This parameter tracks the congestion state of this node and the neighboring. Its value can be either Congested or NotCongested.

The Neighboring Peer Congestion Status is set to Congested if either the node locally becomes congested or if a PTSP or PTSE Acknowledgement packet is received with the Congestion bit set.

The Neighboring Peer Congestion Status is set to NotCongested if both the node locally is not congested and a PTSP or PTSE Acknowledgement packet is received with the Congestion bit not set.
Resynch Retry Count
The number of times the database resynchronization has been held off by congestion.
MaxResynchRetries
The maximum number of times the database resynchronization is expected to be held off by congestion. If the database resynchronization is delayed by more than this number of retries, management should be notified.
ResynchInactivityInterval
The amount of time, in seconds, before a node declares a database resynchronization with a neighbor has failed and that it shall start a database synchronization in the Negotiating state.
Resynch Inactivity Timer
When this timer expires, the node shall withdraw the link PTSE advertisements to the neighboring peer, move to the Negotiating state and restart the database synchronization process over again.

"Last Received Database Summary Packet's Identifying Information
The Database Summary packet flags (including the Initialize, More, Master, Synch In Full bit, and reserved bits) and DS sequence number contained in the last Database Summary packet received from the neighboring peer. This information is used to determine whether the next Database Summary packet received from the neighboring peer is a duplicate."

Section 5.7.2, add the following new states



Exchanging

In this state the node describes its topology database by sending Database Summary packets with the Synch In Full bit not set to the neighboring peer. Following as a result of processing Database Summary packets, required PTSEs can be requested.

Full

In this state, this node has received all PTSEs known to be available from the neighboring peer. Links to the neighboring peer can now be advertised in PTSEs. In this mode, database resynchronizations cannot be attempted.

Full (Resynch Allowed)

This state is equivalent to the Full state in all respects except that a database resynchronization may be initiated.

Negotiating In Full

In this state, the node has initiated or has been requested by its neighboring peer to initiate database resynchronization procedures. The goal of this step is to decide which node is the master, and to decide upon the initial DS sequence number used during the resynchronization. Links to the neighboring peer remain advertised in PTSEs as in the Full state.

Exchanging In Full

In this state the node describes its topology database by sending Database Summary packets to the neighboring peer with the Synch In Full bit set. Following as a result of processing Database Summary packets with the Synch In Full bit set, required PTSEs can be requested. Links to the neighboring peer remain advertised in PTSEs as in the Full state.

Loading In Full

In this state, a full sequence of Database Summary packets has been exchanged with the neighboring peer, and the required PTSEs are requested and at least one has not yet been received. Links to the neighboring peer remain advertised in PTSEs as in the Full state.

Section 5.7.3, modify DSMismatch and add new event causing neighboring peer state change

DSMismatch

This event is generated if one of the following occurs:

1) A Database Summary packet has been received that:

a) has an unexpected DS sequence number, or
b) unexpectedly has the Initialize bit set, or
c) has an unexpected setting of the Master bit, or
d) has an unexpected setting of the Synch In Full bit, except when the conditions for DSResynchMismatch are met.

2) The Resynch Inactivity Timer has expired.

Any of these conditions indicates that an error has occurred in database synchronization.

DSResynchMismatch

This event restarts the database resynchronization procedures without dropping to the Negotiating state, in the event a node experiences an internal error. It is generated when the neighboring peer state is Negotiating In Full, Exchanging In Full, or Loading In Full and a Database Summary packet has been received with the Synch In Full bit set and the Initialize bit set, unexpectedly. Note that the setting of the Master bit and DS sequence number can differ from what was expected.

DSResynchInit

This event indicates that a resynchronization must be initiated. This is generated when a node requires to resynchronize with a neighboring peer as described in Section 5.7.

SynchDone

The neighboring peer's last Database Summary packet has been received, this node's last Database Summary packet has been sent, ~~and~~ the PTSE request list is empty, and the Neighboring Peer Congestion Status is Congested.

LoadingDone

The last PTSE on the PTSE Request List has been received and the Neighboring Peer Congestion Status is Congested.

SynchDone(ResynchAllowed)

The neighboring peer's last Database Summary packet has been received, this node's last Database Summary packet has been sent, the PTSE request list is empty, and Neighboring Peer Congestion Status is NotCongested.

LoadingDone(ResynchAllowed)

The last PTSE on the PTSE Request List has been received, and the Neighboring Peer Congestion Status is NotCongested.

ResynchAllowed

The Neighboring Peer Congestion Status changed to NotCongested, implying that a neighboring peer that previously was restricted from performing a resynchronization is now able to.

ResynchNotAllowed

The Neighboring Peer Congestion Status changed to Congested, implying that a neighboring peer that previously was not restricted from performing a resynchronization is now unable to.

Section 5.7.4:

Table 5-12: Neighboring Peer FSM

	States								
Events	**NPDown**	**Negotiating**	**Exchanging**	**Loading**	**Full**	**Full(Resynch Allowed)**	**Negotiating In Full**	**Exchanging In Full**	**Loading In Full**
Add Port	Ds1 Negotiating	Ds7 Negotiating	Ds7 Exchanging	Ds7 Loading	Ds8 Full	Ds8 Full(Resynch Allowed)	Ds8 Negotiating In Full	Ds8 Exchanging In Full	Ds8 Loading In Full
Negotiation Done	FSM_ERR	Ds2 Exchanging	FSM_ERR	FSM_ERR	FSM_ERR	Ds16 Exchanging In Full	Ds16 Exchanging In Full	FSM_ERR	FSM_ERR
Exchange Done	FSM_ERR	FSM_ERR	Ds3 Loading	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds3 Loading In Full	FSM_ERR
Synch Done	FSM_ERR	FSM_ERR	Ds4 Full	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds17 Full	FSM_ERR
Loading Done	FSM_ERR	FSM_ERR	FSM_ERR	Ds4 Full	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds17 Full
Synch Done(ResynchAllowed)	FSM_ERR	FSM_ERR	Ds4 Full(Resynch Allowed)	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds17 Full(Resynch Allowed)	FSM_ERR
Loading Done(ResynchAllowed)	FSM_ERR	FSM_ERR	FSM_ERR	Ds4 Full(Resynch Allowed)	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds17 Full(Resynch Allowed)
DS Mismatch	FSM_ERR	FSM_ERR	Ds5 Negotiating	Ds5 Negotiating	Ds6 Negotiating	Ds6 Negotiating	Ds6 Negotiating	Ds6 Negotiating	Ds6 Negotiating
DS Resynch Init	FSM_ERR	FSM_ERR	FSM_ERR	Ds19 Loading	Ds19 Full	Ds15 Negotiating in Full	Ds18 Negotiating in Full	Ds18 Negotiating in Full	Ds18 Negotiating in Full
DS Resynch Mismatch	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds18 Negotiating in Full	Ds18 Negotiating in Full	Ds18 Negotiating In Full
ResynchNot Allowed	FSM_ERR	Ds0 Negotiating	Ds0 Exchanging	Ds0 Loading	FSM_ERR	Ds0 Full	Ds0 Negotiating in Full	Ds0 Exchanging In Full	Ds0 Loading In Full

ResynchAllowed	FSM_ERR	Ds0 Negotiating	Ds0 Exchanging	Ds0 Loading	Ds0 Full (Resynch Allowed)	FSM_ERR	Ds0 Negotiating in Full	Ds0 Exchanging In Full	Ds0 Loading In Full
BadPTSE Request	FSM_ERR	FSM_ERR	Ds5 Negotiating	Ds5 Negotiating	Ds6 Negotiating	Ds6 Negotiating	FSM_ERR	Ds6 Negotiating	Ds6 Negotiating
DropPort	FSM_ERR	Ds9 Negotiating	Ds9 Exchanging	Ds9 Loading	Ds9 Full	Ds9 Full(Resynch Allowed)	Ds9 Negotiating In Full	Ds9 Exchanging In Full	Ds9 Loading In Full
DropPort Last	FSM_ERR	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown	Ds10 NPDown
DS Rxmt Timer Expired	FSM_ERR	Ds11 Negotiating	Ds11 Exchanging	FSM_ERR	FSM_ERR	FSM_ERR	Ds11 Negotiating In Full	Ds11 Exchanging In Full	FSM_ERR
Request Rxmt Timer Expired	FSM_ERR	FSM_ERR	Ds12 Exchanging	Ds12 Loading	FSM_ERR	FSM_ERR	FSM_ERR	Ds12 Exchanging In Full	Ds12 Loading In Full
PTSE Rxmt Timer Expired (1)	FSM_ERR	FSM_ERR	Ds13 Exchanging	Ds13 Loading	Ds13 Full	Ds13 Full(Resynch Allowed)	Ds13 Negotiating In Full	Ds13 Exchanging In Full	Ds13 Loading In Full
Peer Delayed Ack Timer Expired	FSM_ERR	FSM_ERR	Ds14 Exchanging	Ds14 Loading	Ds14 Full	Ds14 Full(Resynch Allowed)	Ds14 Negotiating In Full	Ds14 Exchanging In Full	Ds14 Loading In Full
Resynch Inactivity Timer Expired	FSM_ERR	FSM_ERR	FSM_ERR	FSM_ERR	Ds6 Negotiating	Ds6 Negotiating	Ds6 Negotiating	Ds6 Negotiating	Ds6 Negotiating
Resynch Retry Timer Expired	FSM_ERR	FSM_ERR	FSM_ERR	Ds19 Loading	Ds19 Full	Ds15 Negotiating In Full	FSM_ERR	FSM_ERR	FSM_ERR

Ds1

Action: For the case of lowest-level nodes, which are connected by physical links and/or VPCs, the port ID is added to the Port ID List in the neighboring peer data structure.

"Upon entering this state, if this is the first time that an adjacency has been attempted, the DS sequence number should be assigned some unique value (like the time of day clock). Otherwise, the node increments the DS sequence number saved from the previous time this adjacency was active for this neighboring peer, if that information is still available. Upon entering this state, the node increments the DS sequence number for this neighboring peer. If this is the first time that an adjacency has been attempted, the DS sequence number should be assigned some unique value (like the time of day clock). It then declares itself master (sets the Master bit to one), and starts sending Database Summary packets, with the Initialize, More, and Master bits set and the Synch In Full bit not set. No PTSE Summaries are included in this packet. This Database Summary packet is retransmitted at intervals of DSRxmtInterval until the next state is entered (See Section 5.7.5).

Ds5

Action: The Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer and Resynch Inactivity Timer are stopped if not previously stopped. The Peer Retransmission List, Peer Delayed Acks List, PTSE Request List and all related timers are cleared. The exchange of database summaries must start over again. The node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, and Master bits set and the Synch In Full bit not set. No PTSE summaries are included in this packet. The DS Rxmt Timer is started and the Database Summary packet is retransmitted each DSRxmtInterval time.

Ds6

Action: Same as Ds5, except if there are any PTSEs advertising links to that neighbor, those PTSEs must be modified to remove the links. Such PTSEs must be re-originated or if necessary, flushed. The Resynch Retry Timer is stopped if not previously stopped.

Ds10

Action: The Peer Delayed Ack Timer, DS Rxmt Timer, Resynch Retry Timer, Resynch Inactivity Timer, and Request Rxmt Timer are stopped, if not previously stopped. The Peer Retransmission List, Peer Delayed Acks List, PTSE Request List and all related timers are cleared.

Ds15

Action: The Resynch Retry Timer is stopped if not previously stopped. The node initiates the database resynchronization. It starts sending Database Summary packets, with the Initialize, More, Master and Synch In Full bits set and the DS sequence number set to the value stored in the Last Received Database Summary Packet's Identifying Information of the neighboring peer data structure. No PTSE Summaries are included in this packet. The DS Rxmt Timer is started and the Database Summary packet is retransmitted each DSRxmtInterval time (see Sections 5.7.5). Start the Resynch Inactivity Timer.

Ds16

Action: Same as Ds2, except database summaries sent have the Synch In Full bit set. The Resynch Retry Timer is stopped if not previously stopped. Start the Resynch Inactivity Timer if the timer is not already running.

Ds17

Action: Stop the DS Rxmt Timer and Resynch Inactivity Timer. The databases are now resynchronized.

Ds18

Action: The Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer are stopped if not previously stopped. The PTSE Request List and all related timers are cleared. The exchange of database summaries with the Synch In Full bit set must start over again.

The node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master and Synch In Full bits set. No PTSE summaries are included in this packet. The DS Rxmt Timer is started and the Database Summary packet is retransmitted each DSRxmtInterval time.

Ds19

Action: Restart the Resynch Retry Timer.

Section 5.7.5 Sending Database Summary Packets (new text)

This section describes how Database Summary packets are sent to a neighboring peer.

Only one Database Summary packet is allowed outstanding at any one time.

The sending of Database Summary packets depends on the neighboring peer state.

In the Negotiating state, the node sends empty Database Summary packets, with the Initialize, More and Master bits set and the Synch In Full bit not set. When sending such Database Summary packets, the DS Rxmt Timer must be restarted. These packets are retransmitted every DSRxmtInterval seconds, when the DS Rxmt Timer fires.

In the Exchanging state, including when sending the Database Summary packet in response to the event NegotiationDone, the Database Summary packets contain summaries of the topology state information contained in the node's database. In the case of Logical Group Nodes, those portions of the topology database that were originated or received at the level of the Logical Group Node or at higher levels are included in the database summary (lower level PTSEs may belong to the switching system's topology database for one or more of its incarnations as a lower level node, but do not belong to the logical group node's topology database). The PTSP and PTSE header information of each such PTSE is listed in one of the node's Database Summary packets. PTSEs for which new instances are received after the Exchanging state has been entered need not be included in any Database Summary packet, since they will be handled by the normal flooding procedures. It is recommended but not required that each PTSE be included at most once in the entire sequence of Database Summary packets sent to the neighboring peer.

In the Exchanging state, the determination of when to send a Database Summary packet depends on whether the node is master or slave. When a new Database Summary packet is to be sent, the packet's DS sequence number is set as described in Section 5.7.6 and a new set of PTSEs from the node's topology database is described. Each item is considered to have been received by the neighboring peer when the Database Summary packet in which it was included is acknowledged. Note that the More bit is set asymmetrically, with different rules used depending on whether the node is master or slave:

Master

Database Summary packets are sent when either (i) the slave acknowledges the previous Database Summary packet by echoing the DS sequence number, or (ii) DSRxmtInterval seconds elapse without an acknowledgment, in which case the previous Database Summary packet is retransmitted. The DS Rxmt Timer must be restarted whenever a Database Summary packet is transmitted. "If the node has already sent its entire sequence of

Database Summary packets, then the More bit must be set to zero. If this packet includes the last portions of the database summary to be sent to the slave, then the More bit may optionally be set to zero.

Slave

Database Summary packets are sent only in response to Database Summary packets received from the master. If the packet received from the master is new, a new Database Summary packet is sent; otherwise the previous Database Summary packet is retransmitted. If the node has already sent its entire sequence of Database Summary packets (i.e., the contents of this Database Summary packet are empty), then the More bit must be set to zero.

In states Loading and Full the slave must resend its last Database Summary packet in response to duplicate Database Summary packets received from the master. Note that in the Loading or Full state, the last packet that the slave had sent must have been empty, with the Initialize, More, and Master bits set to zero and with the same DS sequence number as in the current neighboring peer data structure.

In the Full(Resynch Allowed) state, the node may initiate a database resynchronization with a particular neighbor (events triggering a database resynchronization are implementation specific, however may include things like error recovery) by injecting a DSResynchInit event into the corresponding neighboring peer FSM. As part of the DSResynchInit event, the node sends empty database summary packets with the Initialize, More, Master and Synch In Full bits set and the DS sequence number set to the value stored in the Last Received Database Summary Packet's Identifying Information of the neighboring peer data structure. The Resynch Inactivity timer is started when the first such database summary packet is set. When sending such database summary packets, the DS Rxmt Timer must be restarted. These packets are re-transmitted by the node initiating a database resynchronization every DS Rxmt Interval seconds, when the DS Rxmt Timer fires. Also in the Full(Resynch Allowed) state, a node sends database summary packets in response to received database summary packets containing the Initialize, More, Master and Synch in Full bits set from a neighboring peer that is requesting a database resynchronization. When a node responds to a neighboring peer requesting a database resynchronization, it sets the Synch In Full, Initialize, More and Master bits as described in Section 5.7.6. When a node responds to the first such database summary packet to a neighboring peer requesting a database resynchronization, it starts the Resynch Inactivity Timer.

In the Exchanging in Full state, the node sends database summary packets identical to the Exchanging state described above, except the Synch in Full bit is set in these database summary packets.

Section 5.7.6, Receiving Database Summary Packets

This section explains the detailed processing of a received Database Summary packet. The incoming Database Summary packet is associated with a neighboring peer by the interface over which it was received. Each Database Summary packet has a DS sequence number, and is implicitly acknowledged. The further processing of the Database Summary packet depends on the state of the neighboring peer data structure associated with the Remote Node ID.

If a Database Summary packet is accepted, the following packet fields are saved in the corresponding neighboring peer data structure as the "Last Received Database Summary Packet's Identifying Information": the Database Summary packet flags (including the Initialize, More, Master, Synch in Full and reserved bits), and the DS sequence number. If these fields are set identically in two consecutive Database Summary packets received from the neighboring peer, the second Database Summary packet is considered to be a "duplicate" in the processing described below.

If the neighboring peer state is NPDown the packet must be ignored.

Otherwise, if the state is:

Negotiating

If the received packet matches one of the following cases, then the neighboring peer state machine must be executed with the event NegotiationDone (causing the state to transition to Exchanging) and the packet must be accepted as next in sequence and processed further. Otherwise, the packet must be ignored.

- The Initialize, More and Master bits are one, the Synch In Full bit is set to zero, the contents of the packet are empty, and the neighboring peer's Node ID is larger than this node's own node ID. In this case this node is now a Slave. Upon generating the event NegotiationDone, the slave must take the following actions:

  - Stop the DS Rxmt Timer,
  - Set the Master bit to zero (indicating slave), set the Initialize bit to zero, set the DS sequence number to that specified by the master, and send a Database Summary packet to the master including the first portion of this node's database summary (see Section 5.7.5).

- The Initialize, ~~and~~ Master, and Synch In Full bits are zero, the packet's DS sequence number equals the node's own DS sequence number (indicating acknowledgment), and the neighboring peer's node ID is smaller than the node's own node ID. In this case this node is Master. Upon generating the event NegotiationDone, the master must take the following actions (the last two actions need not be taken in this order):

  - Stop the DS Rxmt Timer,
  - Process the contents of the received Database Summary packet (see the last paragraph describing actions to be taken under the Exchanging state below),
  - Increment the DS sequence number by one, set the Initialize bit to zero, send a Database Summary packet to the slave including the first portion of this node's database summary (see Section 5.7.5) and restart the DS Rxmt Timer.

Exchanging

Execute the following steps in order:

- If the node is master and the received Database Summary packet is a duplicate, stop processing the packet.
- If the node is slave and the received Database Summary packet is a duplicate, respond by retransmitting the last Database Summary packet sent to the master and stop processing the received Database Summary packet.
- If the state of the Master bit is inconsistent with the master/slave state of the connection, generate the event DSMismatch and stop processing the packet. Otherwise:
- If the Synch In Full bit is set, generate the event DSMismatch and stop processing the packet.
- If the Initialize bit is set, generate the event DSMismatch and stop processing the packet.
- If the node is master and the packet's DS sequence number equals the node's own DS sequence number (this packet is the next in sequence), the packet must be accepted and processed as follows (the last two actions need not be taken in this order):

  - Stop the DS Rxmt Timer,
  - Process the contents of the received Database Summary packet (see below),
  - In the following order:
    A) Increment the DS sequence number by one,
    B) If the node has already sent its entire sequence of Database Summary packets (i.e., the previous Database Summary packet that the node sent had the More bit set to zero), and the received packet also has the More bit set to zero, generate the event ExchangeDone if the PTSE Request List is not empty, or the event SynchDone if the PTSE Request List is empty.
    C) Otherwise, send a new Database Summary packet to the slave and restart the DS Rxmt Timer (see Section 5.7.5).
- If the node is slave and the packet's DS sequence number is one more than the node's own DS sequence number (this packet is the next in sequence), the packet must be accepted and processed as follows (in no particular order):
  - Process the contents of the received Database Summary packet (see below),
  - In the following order:
    D) Set the DS sequence number to the DS sequence number appearing in the received packet,
    E) Send a Database Summary packet to the master (see Section 5.7.5),

F) If the received packet has the More bit set to zero, and the just transmitted Database Summary packet also had its More bit set to zero (i.e., the contents of the just transmitted Database Summary packet were empty), then generate the event ExchangeDone if the PTSE Request List is not empty, or the event SynchDone if the PTSE Request List is empty.

- Else, generate the event DSMismatch and stop processing the packet.

Processing the contents of a received Database Summary packet:

When the node accepts a received Database Summary packet as the next in sequence, the contents of the most recently transmitted Database Summary packet are acknowledged as having been received and the contents of the received Database Summary packet are processed as follows.

For each PTSE listed, the node looks up the PTSE in its database to see whether it also has an instance of the PTSE. If it does not, or if the database copy is less recent (see Section 5.7.5), one of the following actions is taken:

- If the listed PTSE is one of this node's self-originated PTSEs, the node must either:
  - Re-originate a newer instance of the PTSE with a larger sequence number, if the node has a valid instance of the PTSE (see Section 5.7.5), or
  - Flush the PTSE from the routing domain after installing it in the topology database with the remaining lifetime set to ExpiredAge (see Section 5.7.5).
- Otherwise, if the listed PTSE has PTSE Remaining Lifetime ExpiredAge, the PTSP and PTSE header contents in the PTSE summary must be accepted as a new or updated PTSE with empty contents. Follow the procedures for receiving a PTSE in Section 5.7.5 to determine whether or not the PTSE must be flooded to other neighboring peers, after installing the PTSE in the topology database.
- Otherwise, the PTSE is put on the PTSE request list, so that it can be requested from a neighboring peer (immediately or at some later time) in PTSE Request packets.

Loading, ~~or~~ Full, or Full(Resynch Allowed) and the received database summary packet has Synch In Full bit not set

In these states, and for this type of Database Summary packet received with the Synch In Full bit not set, the packet must be a duplicate. Any other Database Summary packet with the Synch In Full bit not set must generate the event DSMismatch, causing the adjacency to revert to the Negotiating state and the two neighboring peers to resynchronize their databases.

The procedures followed when receiving a Database Summary packet with the Synch In Full bit not set in these states are the same as those followed in the Exchanging state, except that packets accepted as the next in sequence must generate the event DSMismatch instead, and further processing of such packets must be stopped. Note that receipt of packets with an inconsistent Master bit or with the Initialize bit set to one must also generate the event DSMismatch.

Loading or Full and the received database summary packet has Synch In Full bit set

If the received packet has the Initialize, More, Master and the Synch In Full bits set to one, and the contents of the packet are empty, then the neighboring peer is requesting a database resynchronization. The node must generate the event DSResynchInit. Otherwise, the packet must be checked for duplicate. If the packet is a duplicate, then it is ignored. Otherwise it is not a duplicate and an error occurred during resynchronization. The state machine must be executed with the event DSMismatch.

Full(Resynch Allowed) and the received database summary packet has Synch in Full bit set

If the received packet matches one of the following cases, then the neighboring peer state machine must be executed with the specified event. Otherwise, the packet must be checked for duplicate. If the packet is a duplicate, then it is ignored. Otherwise it is not a duplicate and an error occurred during resynchronization. The state machine must be executed with the event DSMismatch.

- The Initialize, More, Master and the Synch In Full bits are one, the contents of the packet are empty, and the neighboring peer's Node ID is smaller than this node's own node ID. In this case this node is the master and the neighboring peer is requesting a database resynchronization. The master must generate the event DS Resynch Init (causing the state to transition to Negotiating in Full) and take the following actions:

  - Stop the DS Rxmt Timer, if it was still running
  - Start the Resynch Inactivity Timer if the timer is not already running
  - Set the Initialize, More, Master and Synch In Full bits to one, increment the DS sequence number by one, and send an empty Database Summary packet to the slave (see Section 5.7.5).
  - Restart the DS Rxmt Timer.

- The Initialize, More, Master and the Synch In Full bits are one, the contents of the packet are empty, and the neighboring peer's Node ID is larger than this node's own node ID. In this case this node is now a slave. The slave must generate the event NegotiationDone (causing the state to transition to Exchanging In Full), and take the following actions:

  - Stop the DS Rxmt Timer,
  - Start the Resynch Inactivity Timer if the timer is not already running
  - Set the Master bit to zero (indicating slave), set the Initialize bit to zero, set the Synch In Full bit to one, set the DS sequence number to that specified by the master, and send a Database Summary packet to the master including the first portion of this node's database summary (see Section 5.7.5).

Negotiating In Full

If the received packet matches the following cases, then the neighboring peer state machine must be executed with the event Negotiation Done (causing the state to transition to Exchanging In Full) and the packet must be accepted as next in sequence and processed further. Otherwise, the packet must be checked for duplicate. If the packet is a duplicate, then it is ignored. Otherwise it is not a duplicate and an error occurred during resynchronization. If the packet has the Initialize and Synch In Full bits set, then the state machine must be executed with the event DS Resynch Mismatch. Otherwise, the state machine must be executed with the event DSMismatch.

- The Initialize and Master bits are zero and the Synch In Full bit is set, the packet's DS sequence number equals the node's own DS sequence number (indicating acknowledgment), and the neighboring peer's node ID is smaller than the node's own node ID. In this case this node is acknowledged as the master. Upon generating the event Negotiating Done, the master must take the following actions (the last two actions need not be taken in this order):

  - Stop the DS Rxmt Timer, if it was running
  - Process the contents of the received Database Summary packet (see Exchanging state for processing contents of received database summary packet),
  - Increment the DS sequence number by one, set the Initialize bit to zero, send a Database Summary packet to the slave including the first portion of this node's database summary (see Section 5.7.5) and restart the DS Rxmt Timer.

- The Initialize, More, Master and the Synch In Full bits are one, the contents of the packet are empty, and the neighboring peer's Node ID is larger than this node's own node ID. In this case this node is being acknowledged as a slave. Upon generating the event Negotiation Done, the slave must take the following actions:

  - Stop the DS Rxmt Timer, if it was still running

- Set the Master bit to zero (indicating slave), set the Initialize bit to zero, set the DS sequence number to that specified by the master, and send a Database Summary packet to the master including the first portion of this node's database summary (see Section 5.7.5).

Exchanging In Full

Execute the following steps in order:

- If the node is master and the received Database Summary packet is a duplicate, stop processing the packet.
- If the node is slave and the received Database Summary packet is a duplicate, respond by re-transmitting the last Database Summary packet sent to the master and stop processing the received Database Summary packet.
- If the state of the Master bit is inconsistent with the master/slave state of the connection, generate the event DSMismatch and stop processing the packet. Otherwise:
- If the Synch In Full bit is not set, generate the event DSMismatch and stop processing the packet.
- If the Initialize bit is set, generate the event DS-Resynch-Mismatch and stop processing the packet.
- If the node is master and the packet's DS sequence number equals the node's own DS sequence number (this packet is the next in sequence), the packet must be accepted and processed as follows (the last two actions need not be taken in this order): <See Exchanging State when identical scenario occurs for details about actions taken>
- If the node is slave and the packet's DS sequence number is one more than the node's own DS sequence number (this packet is the next in sequence), the packet must be accepted and processed as follows (in no particular order): <See Exchanging State when identical scenario occurs for details about actions taken>
- Else, generate the event DSMismatch and stop processing the packet.

Processing the contents of a received Database Summary packet:

<See Exchanging state for processing contents of received database summary packet>

Loading In Full

In this state, the node has sent and received an entire sequence of Database Summary packets with the Synch In Full bit set. The only packets received should be duplicates. Any other Database Summary packets received that are not requesting a resynchronization must generate the event DSMismatch, causing the adjacency to revert to the Negotiating state and the two neighboring peers to synchronize their databases.

The procedures followed when receiving a Database Summary packet in the Loading In Full state are the same as those followed in the Exchanging In Full state, except that packets accepted as the next in sequence must generate the event DSMismatch instead, and further processing of such packets must be stopped. Note that receipt of packets with an inconsistent Master bit must also generate the event DSMismatch. Any packets with only an inconsistent Initialize bit must generate the event DS-Resynch-Mismatch.

Section 5.7.7, Modify the following paragraphs

Replace the wording "states Exchanging and Loading" to "states Exchanging, Loading, Exchanging In Full and Loading In Full" in every paragraph except the last.

Modify the last paragraph as follows:

When the PTSE request list becomes empty, and the neighboring peer state is Loading or Loading In Full (i.e., a complete sequence of Database Summary packets has been both sent to and received from the neighboring peer), the LoadingDone event is generated.

Section 5.7.8, Change the first paragraph as follows

Received PTSE Request packets specify a list of PTSEs that the neighboring peer wishes to receive. PTSE Request packets must be accepted when the corresponding neighboring peer state is Exchanging, Loading, Exchanging In Full, Loading In Full, Full(Resynch Allowed) or Full. In all other states PTSE Request packets must be ignored.

Add a new Section, 5.7.9 Resynch Inactivity Timer

When a node wishes to resynchronize its database with its neighboring peer (for instance, as the result of the node experiencing a local fault), the node must inject the DS-Resynch-Init event into the neighboring peer FSM. If database resynchronization is allowed, then processing this event causes the neighboring peer FSM to start the Resynch Inactivity Timer, if the timer was not already running. When a node receives a Database Summary packet in the Full(Resynch Allowed) state with the Initialize, More, Master, and Synch In Full bits set, it must start the Resynch Inactivity Timer if the timer is not already running.

In the event the Resynch Timer expires, the neighboring peer state machine must be executed with the event DS Mismatch. This forces the node to withdraw all links to that neighbor currently being advertised in PTSEs and forces the neighboring peer state machine to the Negotiating State where it attempts to synchronize databases starting from the Negotiating State.

In the event of a successful database resynchronization, the Resynch Inactivity Timer must be stopped.

Add Section 5.7.10 Resynch Congestion Indication

When a node wishes to resynchronize its database with its neighboring peer (for instance, as the result of the node experiencing a local fault or PTSEs that were lost due to congestion), the node must first check the Neighboring Peer Congestion Status.

The Resynch Congestion Indication (RCI) should be included in PTSPs and PTSE Acknowledgements transmitted to a neighboring peer, if the node's internal resources (memory, buffer management, CPU utilization, etc.) are unable to handle a database resynchronization between the neighboring peer and the node. For this case, RCI should be set to inhibit that neighbor from attempting a database resynchronization. The RCI should be set to not congested in PTSPs and PTSE Acknowledgements transmitted to a neighboring peer when the node's internal resources become available and a database resynchronization can be performed safely between the neighboring peer and the node.

If the Neighboring Peer Congestion Status is set to Congested, then the database resynchronization is not attempted but the Resynch Retry Timer is started, if it is not already running. This is used so that the neighboring peer FSM continually retries to perform a database resynchronization in the event of congestion and allows some time for the congestion condition to clear.

In the event the Resynch Retry Timer expires, the neighboring peer state machine must be executed with the event ResynchRetryTimerExpired. If the Neighboring Peer Congestion Status is set to NotCongested, this will cause the neighboring peer state machine to transition into the Negotiating In Full state. If the Neighboring Peer Congestion Status is set to Congested, the Resynch Retry Timer is restarted. A count of the number of times the database resynchronization attempt was delayed due to congestion should be kept. When this count exceeds the MaxResynchRetries value, management should be notified. This count is cleared when the state changes from Loading to any state other than Full or from Full to any other state.

Add Section 5.7.11 Limiting Simultaneous Resynchronizations

In order to throttle the number of adjacencies brought up simultaneously, the node SHALL limit the number of adjacencies resynchronizing to no more than Nmaxresynch adjacencies being brought up simultaneously. Once a subset of adjacencies have been brought up successfully, newer adjacencies may be brought up as long as the number of simultaneous adjacencies being brought up does not exceed Nmaxresync. The appropriate value of Nmaxresynch would depend on the node processing power, link bandwidth and propagation delay. The value of Nmaxresynch is an architectural variable.

Section 5.8.3.1, modify text as follows

Replace the wording "Exchanging, Loading, or Full" to "Exchanging, Loading, Full, Full(Resynch Allowed), Negotiating In Full, Exchanging In Full or Loading In Full" in Section 5.8.3.1.

Section 5.8.3.3, modify text as follows

Replace the wording "Exchanging or Loading" to "Exchanging, Loading, Exchanging In Full or Loading In Full" in steps (2) and (6) of Section 5.8.3.3.

Replace the wording "Exchanging or Loading" to "Exchanging, Loading, Negotiating In Full, Exchanging In Full or Loading In Full" in step (5) of Section 5.8.3.3.

Change Table 5-20 as follows:

Table 5-20: The PNNI Packet Header

Offset	Size (Octets)	Name	Function/Description
0	2	Packet Type	See Table 5-21 PNNI Packet Types.
2	2	Packet length	
4	1	Protocol version	Specifies the version according to which this packet was formatted. This specification defines version one of the PNNI routing protocol packet formats.
5	1	Newest version supported	The newest version supported and oldest version supported fields are included in order for nodes to negotiate the most recent protocol version that can be understood by both nodes exchanging a particular type of packet.
6	1	Oldest version supported	See Above.
7	1	~~*Reserved*~~*Flags*	See Table 5-21a PNNI Packet Flags

Table 5-21a: PNNI Packet Flags

Bit ID:	bit 8 (MSB)	bits 7..1 (LSB)
Bit Name:	'Resynch Congestion Indication' (RCI) bit	*Reserved*
Description:	When the packet is type PTSP or PTSE Acknowledgement, set to zero when congestion is detected within the node and one when there is no congestion. Set to zero for other packet types.	

The following table illustrates the information used in a Database Summary packet.

Table 5-42: The Database Summary Packet

Offset	Size (Octets)	Name	Function/Description
0	8	PNNI Header	A PNNI packet header structure with Packet type = 4 (Database summary packet). See Table 5-20.
8	2	Flags	See Table 5-43.
10	2	*Reserved*	
12	4	DS sequence number	
Repeat for each set of PTSEs in the topology database:			
	2	Type	Type = 512 (Nodal PTSE summaries)
	2	Length	
	22	Originating node ID	
	14	Originating node's Peer Group ID	
	2	*Reserved*	
	2	PTSE Summary Count	The number of PTSE summaries for this originating node ID.
Repeat the following structure PTSE Summary Count times:			
	2	PTSEType	
	2	*Reserved*	
	4	PTSE Identifier	
	4	PTSE Sequence Number	
	2	PTSE Checksum	
	2	PTSE Remaining Lifetime	

Table 5-43: Database Summary Packet Flags

Bit ID:	bit 16 (MSB)	bit 15	bit 14	bit 13	bits 1~~3~~2..1 (LSB)
Bit Name:	'Initialize' (I) bit	'More'(M) bit	'Master' (MS) bit	'Synch In Full' (SF) bit	*Reserved*
Description:	Set to one during initialization of the database synchronization process. Otherwise set to zero.	Set to one if the transmitting node has additional PTSEs to summarize, or zero if all PTSEs have been summarized (see Section 5.7.5).	Set to one initially by both nodes. Later set to zero by the node that assumes the slave role in the database synchronization process.	Set to one when a node wishes to perform a database resynchronization.	

3.3 Architectural Variables

These are the architectural variables used in the PNNI specification.

GracefulRestartInterval: default value 300 seconds.
: The period of time that a node which initiates graceful restart performs graceful restart procedures identified in Section 21.5.

Tbackup: default value 300 seconds.
: The maximum period of time between backup database updates at a node.

MaxResynchRetries: default value 10
: The maximum number of times the database resynchronization is expected to be held off by congestion. If the database resynchronization is delayed by more than this number of retries, management should be notified.

Nmaxresync: default value implementation dependent
: Maximum number of adjacencies to be resynchronized simultaneously at a node.

ResynchInactivityInterval: default value 180 seconds
: The amount of time, in seconds, before a node declares a database resynchronization with a neighbor has failed and that it shall start a database synchronization in the Negotiating state.

ResynchRetryInterval: default value 20 seconds
: The amount of time, in seconds, to delay when a database resynchronization is requested but cannot be attempted due to congestion as indicated by the Neighboring Peer Congestion Status.

StressInactivityFactorRestart: default value 4.
: The multiplier to be used to increase the Hello Inactivity time and Horizontal Link Inactivity time during PNNI Graceful Restart.

4. References

4.1 References [Normative]

[af-pnni-0055.000] "Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)," March 1996.

[af-pnni-0055.002] "Private Network-Network Interface Specification Version 1.1 (PNNI 1.1)," April 2002.

[af-pnni-0081.000] "PNNI V1.0 Errata and PICS," May 1997.

[af-cs-0200.000] "PNNI Routing Congestion Control, Version 1," June 2004.

4.2 References [Informative]

[ash] Ash, G. R., "Dynamic Routing in Telecommunications Networks," McGraw Hill.

[att] "AT&T announces cause of frame-relay network outage," AT&T Press Release, April 22, 1998.

[cholewka] Cholewka, K., "MCI Outage Has Domino Effect," Inter@ctive Week, August 20, 1999.

[choudhury1] Choudhury, G., Maunder, A. S., Sapozhnikova, V., "Faster Link-State Convergence and Improving Network Scalability and Stability," submitted for presentation at LCN 2001.

[choudhury2] Choudhury, G., Ash, G., Manral, V., Maunder, A., Sapozhnikova, V., "Prioritized Treatment of Specific OSPF Packets and Congestion Avoidance: Algorithms and Simulations," AT&T Technical Report, August 2003.

[choudhury3] Choudhury, G., Ash, G., Manral, V., Maunder, A., Sapozhnikova, V., "Prioritized Treatment of Specific OSPF Packets and Congestion Avoidance," Internet Draft, work in progress.

[GR-472-CORE] Bellcore, General Requirements Document 472-CORE.

[jander] Jander, M., "In Qwest Outage, ATM Takes Some Heat," Light Reading, April 6, 2001.

[mummert] Mummert, V. S., "Network Management and its Implementation on the No. 4ESS," International Switching Symposium, Japan, 1976.

[moy1] Moy, J., et. al., "Graceful OSPF Restart", RFC 3623, November 2003.

[moy2] Moy, J., "OSPFD 2.0 Patch listing, Patch 2.2 and 2.4, Hitless Restart," http://ospf.org/ospfd2_0/patches.html, September 2001.

[pappalardo1] Pappalardo, D., "Can one rogue switch buckle AT&T's network?," Network World Fusion, February 23, 2001.

[pappalardo2] Pappalardo, D., "AT&T, customers grapple with ATM net outage," Network World, February 26, 2001.

[zinen1] Zinin, A., et. al., "OSPF Restart Signaling," work in progress.

[zinin2] Zinin, A., et. al., "OSPF Out-of-band LSDB Resynchronization, " work in progress.

Annex A. Protocol Implementation Conformance Statement (PICS) for PNNI Routing Resynchronization Control [Normative]

A.1 Introduction

To evaluate conformance of a particular implementation, it is necessary to have a statement of which capabilities and options have been implemented. Such a statement is called a Protocol Implementation Conformance Statement (PICS).

A.1.1 Scope

This document provides the PICS proforma for PNNI Routing Resynchronization Control, Version 1, defined in [1], in compliance with the relevant requirements, and in accordance with the relevant guidelines, given in ISO/IEC 9646-7. In most cases, statements contained in notes in the specification, which were intended as information, are not included in the PICS.

A.1.2 Normative References

[1] af-cs-0201.000, PNNI Routing Resynchronization Control, Version 1, June 2004.

[2] ISO/IEC 9646-1: 1994, Information technology – Open systems interconnection – Conformance testing methodology and framework – Part 1: General Concepts (See also ITU Recommendation X.290 (1995)).

[3] ISO/IEC 9646-7: 1995, Information technology – Open systems interconnection – Conformance testing methodology and framework – Part 7: Implementation Conformance Statements (See also ITU Recommendation X.296 (1995)).

[4] ISO/IEC 9646-3:1998, Information technology – Open systems interconnection – Conformance testing methodology and interconnection – Part 3: The Tree and Tabular Combined Notation (TTCN) (See also ITU telecommunication X.292 (1998)).

[5] af-pnni-0055.002, Private Network-Network Interface Specification Version 1.1 (PNNI 1.1) – April 2002

[6] af-cs-0200.000, PNNI Routing Congestion Control, Version 1, June 2004

A.1.3 Definitions

Terms defined in [1] and [5]

Terms defined in ISO/IEC 9646-1 and in ISO/IEC 9646-7

In particular, the following terms defined in ISO/IEC 9646-1 apply:

Protocol Implementation Conformance Statement (PICS): A statement made by the supplier of an implementation or system, stating which capabilities have been implemented for a given protocol.

PICS proforma: A document, in the form of a questionnaire, designed by the protocol specifier or conformance test suite specifier, which when completed for an implementation or system becomes the PICS.

A.1.4 Acronyms

ASN.1 Abstract Syntax Notation One

ATS Abstract Test Suite
IUT Implementation Under Test
PICS Protocol Implementation Conformance Statement
SUT System Under Test

A.1.5 Conformance

The PICS does not modify any of the requirements detailed in PNNI Routing Resynchronization Control, Version 1.0. In case of apparent conflict between the statements in the base specification and in the annotations of "M" (mandatory) and "O" (optional) in the PICS, the text of the base specification takes precedence.

The supplier of a protocol implementation, which is claimed to conform to the ATM Forum PNNI Routing Resynchronization Control, is required to complete a copy of the PICS proforma provided in this document and is required to provide the information necessary to identify both the supplier and the implementation.

A.2 Identification of the Implementation

Identification of the Implementation Under Test (IUT) and system in which it resides (the System Under Test (SUT)) should be filled in so as to provide as much detail as possible regarding version numbers and configuration options.

The product supplier information and client information should both be filled in if they are different.

A person who can answer queries regarding information supplied in the PICS should be named as the contact person.

A.2.1 Date of Statement

_____

A.2.2 Implementation Under Test (IUT) Identification

IUT Name: _____

_____

_____

IUT Version: _____

A.2.3 System Under Test (SUT) Identification

SUT Name: _____

_____

Hardware Configuration: _____

_____

_____

Operating System: _____

A.2.4 Product Supplier

Name: _____

Address: _____

_____

_____

Telephone Number: _____

Facsimile Number: _____

Email Address: _____

Additional Information: _____

_____

_____

A.2.5 Client

Name: _____

Address: _____

_____

_____

Telephone Number: _____

Facsimile Number: _____

Email Address: _____

Additional Information: _____

_____

_____

A.2.6 PICS Contact Person

(A person to contact if there are any queries concerning the content of the PICS)

Name: _____

Telephone Number: _____

Facsimile Number: _____

Email Address: _____

Additional Information: _____

_____

_____

Identification of the Protocol Specification
This PICS proforma applies to the following specification:

af-cs-0201.000, PNNI Routing Resynchronization Control, Version 1, June 2004.

A.3 PICS Proforma

A.3.1 Global statement of conformance

The implementation described in this PICS meets all of the mandatory requirements of the reference protocol.

[ ] YES

[ ] NO

Note: Answering "No" indicates non-conformance to the specified protocol. Non-supported mandatory capabilities are to be identified in the following tables, with an explanation by the implementor explaining why the implementation is non-conforming.

A.3.2 Instructions for Completing the PICS Proforma

The PICS Proforma is a fixed-format questionnaire. Answers to the questionnaire should be provided in the rightmost columns, either by simply indicating a restricted choice (such as Yes or No), or by entering a value or a set of range of values.

The following notations, defined in ISO/IEC 9647-7, are used for the support column:

Yes supported by the implementation

No not supported by the implementation

The following notations, defined in ISO/IEC 9647-7, are used for the status column:

M mandatory – the capability is required to be supported.

O optional – the capability may be supported or not.

O.i qualified optional – for mutually exclusive or selectable options from a set. “i” is an integer which identifies a unique group of related optional items and the logic of their selection is defined immediately following the table.

A supplier may also provide additional information, categorized as exceptional or supplementary information. These additional information should be provided as items labeled X.<i> for exceptional information, or S.<i> for supplemental information, respectively, for cross reference purposes, where <i> is any unambiguous identification for the item. The exception and supplementary information are not mandatory and the PICS is complete without such information. The presence of optional supplementary or exception information should not affect test execution, and will in no way affect interoperability verification. The column labeled ‘Reference’ gives a pointer to sections of the protocol specification for which the PICS Proforma is being written.

A.4 PICS for the support of PNNI Routing Resynchronization Control at the PNNI interface

A.4.1 Major Capability at PNNI (MCP)

Item Number	Item Description	Status	Condition for status	Reference	Support
MCP1	Does the IUT support Database Resynchronization at the PNNI interface?	M			Yes__ No__
MCP2	Does the IUT support Database Backup and Graceful Restart at the PNNI interface?	O	MCP1		Yes__ No__
Comments:					

A.4.2 Routing Procedures at the PNNI (RPP)

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP1	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup meets requirements specified in [GR-472-CORE]?	M	MCP2	3.1.1	Yes__No__
RPP2	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: up to date routing tables?	M	MCP2	3.1.1	Yes__No__
RPP3	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: SVCC-based RCC binding information, if any?	M	MCP2	3.1.1	Yes__No__
RPP4	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: Hello FSM data structures on the node and associated LGNs running on the node?	M	MCP2	3.1.1	Yes__No__
RPP5	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: neighboring peer FSM data structures on the node and associated LGNs running on the node?	M	MCP2	3.1.1	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP6	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: self-originated PTSE information?	M	MCP2	3.1.1	Yes__No__
RPP7	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: PGL election FSM data structure on the node and associated LGNs running on the node?	M	MCP2	3.1.1	Yes__No__
RPP8	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where: - the database backup contains: any other relevant state information (e.g. peer group reachability information, aggregation information, nodal state parameters for complex node implementations, etc.) such that the state of the node and other nodes in the network are not affected during the restart?	M	MCP2	3.1.1	Yes__No__
RPP9	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where the database backup contains all non-self-originated PTSEs?	O	MCP2	3.1.1	Yes__No__
RPP10	Does the IUT, prior to restarting, provide a local, non-volatile memory backup where all items are backed up at least every Tbackup seconds?	M	MCP2	3.1.1	Yes__No__
RPP11	Does the IUT initiate PNNI graceful restart automatically after failure and recovery by starting the GracefulRestartTimer timer with GracefulRestartInterval?	M	MCP2	3.1.2.1	Yes__No__
RPP12	Does the IUT perform the following step during PNNI graceful restart: hold off on sending PNNI routing packets until the corresponding FSM states are restored?	M	MCP2	3.1.2.1	Yes__No__
RPP13	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: up to date routing tables that can be reused to establish calls during the restart?	M	MCP2	3.1.2.1	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP14	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: SVCC-based RCC binding information, if any, such that the SVCC-based RCC is maintained across the restart?	M	MCP2	3.1.2.1	Yes__No__
RPP15	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: Hello FSM data structures on the node and associated LGNs running on the node?	M	MCP2	3.1.2.1	Yes__No__
RPP16	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: neighboring peer FSM data structures on the node and associated LGNs running on the node, adjusting the states as follows: - Negotiating, Exchanging, Loading SHALL become Negotiating - Full, Full(Resynch Allowed), Negotiating in Full, Exchanging in Full, Loading in Full SHALL become Full?	M	MCP2	3.1.2.1	Yes__No__
RPP17	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: self-originated PTSE information, such that self-originated PTSEs can be retained during the restart?	M	MCP2	3.1.2.1	Yes__No__
RPP18	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: PGL election FSM data structure on the node and associated LGNs running on the node, such that elected PGL information is not lost?	M	MCP2	3.1.2.1	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP19	Does the IUT perform the following step during PNNI graceful restart: restore data from the local backup memory including: any other relevant state information (e.g. peer group reachability information, aggregation information, nodal state parameters for complex node implementations, etc.) such that the state of the node and other nodes in the network are not affected during the restart?	M	MCP2	3.1.2.1	Yes__No__
RPP20	Does the IUT during PNNI graceful restart complete the restoration of the data from the local backup memory within the hello inactivity interval seconds?	M	MCP2	3.1.2.1	Yes__No__
RPP21	Does the IUT during PNNI graceful restart restore any non-self-originated PTSEs contained in the local backup memory?	M	MCP2	3.1.2.1	Yes__No__
RPP22	Does the IUT during PNNI graceful restart take the restored state information into account when sending PNNI routing packets?	M	MCP2	3.1.2.1	Yes__No__
RPP23	Does the IUT during PNNI graceful restart originate a new Nodal IG with the CSI bits indicating either low or high congestion?	O	MCP2, MCP1 of [6]	3.1.2.1	Yes__No__
RPP24	Does the IUT during PNNI graceful restart follow the procedures in Section 3.2 to recover any missing PTSE information by performing a resynchronization with each neighboring peer?	M	MCP2	3.1.2.1	Yes__No__
RPP25	Does the IUT during PNNI graceful restart limit the number of adjacencies resynchronizing simultaneously to a maximum of Nmaxresync?	M	MCP2	3.1.2.1	Yes__No__
RPP26	Does the IUT during PNNI graceful restart set the hello interval advertised to neighbors in Hello packets to StressInactivityFactorRestart times HelloInterval, set the inactivity factor to StressInactivityFactorRestart times InactivityFactor, and set the horizontal link inactivity time to StressInactivityFactorRestart times HorizontalLinkInactivityTime?	M	MCP2	3.1.2.1	Yes__No__
RPP27	Does the IUT during PNNI graceful restart NOT change its HelloInterval in the hello data structure (the frequency of sending hellos to neighbors does not change)?	M	MCP2	3.1.2.1	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP28	Does the IUT during PNNI graceful restart exit PNNI graceful restart when either of the following occurs: 1. the restarting node has performed a database resynchronization with all of its neighbor nodes and neighbor LGNs, or 2. the GracefulRestartTimer expires?	M	MCP2	3.1.2.2	Yes__No__
RPP29	Does the IUT during PNNI graceful restart take the following actions when exiting PNNI graceful restart: 1. recompute routing tables using the information currently in the PTSE topology database, 2. stop the GracefulRestartTimer if it is still running, and 3. perform database synchronizations starting from the Negotiating state on any neighbor nodes and neighbor LGNs that were not resynchronized during the PNNI graceful restart?	M	MCP2	3.1.2.2	Yes__No__
RPP30	When in Full(Resynch Allowed) state and Add Port event occurs, and this is lowest level neighboring peer, which is connected by physical links or VPC, is the port ID added to the Port ID list as local information (neighboring peer data structure) and a new instance of a PTSE to be originated?	M		Table 5-12 Ds8	Yes__No__
RPP31	When in Negotiating In Full state and Add Port event occurs, and this is lowest level neighboring peer, which is connected by physical links or VPC, is the port ID added to the Port ID list as local information (neighboring peer data structure) and a new instance of a PTSE to be originated?	M		Table 5-12 Ds8	Yes__No__
RPP32	When in Exchanging In Full state and Add Port event occurs, and this is lowest level neighboring peer, which is connected by physical links or VPC, is the port ID added to the Port ID list as local information (neighboring peer data structure) and a new instance of a PTSE to be originated?	M		Table 5-12 Ds8	Yes__No__
RPP33	When in Loading In Full state and Add Port event occurs, and this is lowest level neighboring peer, which is connected by physical links or VPC, is the port ID added to the Port ID list as local information (neighboring peer data structure) and a new instance of a PTSE to be originated?	M		Table 5-12 Ds8	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP34	When in Full(Resynch Allowed) state and Negotiation Done event occurs, does the IUT begin sending Database Summary packets with information and SynchIn Full bit set, stop the Resynch Retry Timer if not previously stopped, start the Resynch Inactivity Time if not already running, and enter the Exchanging in Full state?	M		Table 5-12 Ds16	Yes__No__
RPP35	When in Negotiating In Full state and Negotiation Done event occurs, does the IUT begin sending Database Summary packets with information and SynchIn Full bit set, stop the Resynch Retry Timer if not previously stopped, start the Resynch Inactivity Time if not already running, and enter the Exchanging in Full state?	M		Table 5-12 Ds16	Yes__No__
RPP36	When in Full(Resynch Allowed) state and DS Mismatch event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP37	When in Exchanging In Full state and DS Mismatch event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP38	When in Loading In Full state and DS Mismatch event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP39	When in Negotiating In Full state and DS Mismatch event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP40	When in Full(Resynch Allowed) state and Bad PTSE Request event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP41	When in Exchaning In Full state and Bad PTSE Request event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP42	When in Loading In Full state and Bad PTSE Request event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP43	When in Full state and Resynch Inactivity Timer Expired event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP44	When in Full(Resynch Allowed) state and Resynch Inactivity Timer Expired event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP45	When in Negotiating In Full state and Resynch Inactivity Timer Expired event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP46	When in Exchanging In Full state and Resynch Inactivity Timer Expired event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP47	When in Loading In Full state and Resynch Inactivity Timer Expired event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; is the DS sequence number incremented; is a Database Summary packet sent; is the DS Rxmt timer started and the Negotiating state entered?	M		Table 5-12 Ds6	Yes__No__
RPP48	When in Full(Resynch Allowed) state and DS resynch Init event occurs, is the Resynch Retry Timer stopped if not previously stopped; does the node initiate the database resynchronization and start sending Database Summary packets, with the Initialize, More, Master, and Synch In Full bits set and the DS sequence number set to the value stored in the Last Received Database Summary Packet's Identifying Information of the neighboring peer data structure, with no PTSE Summaries included in this packet; and start the DS Rxmt Timer and retransmit the Database Summary packet each DSRxmtInterval time, and start the Resynch Inactivity Timer, and enter Negotiating In Full state?	M		Table 5-12 Ds15	Yes__No__
RPP49	When in Full(Resynch Allowed) state and Resynch Retry Timer Expired event occurs, is the Resynch Retry Timer stopped if not previously stopped; does the node initiate the database resynchronization and start sending Database Summary packets, with the Initialize, More, Master, and Synch In Full bits set and the DS sequence number set to the value stored in the Last Received Database Summary Packet's Identifying Information of the neighboring peer data structure, with no PTSE Summaries included in this packet; and start the DS Rxmt Timer and retransmit the Database Summary packet each DSRxmtInterval time, and start the Resynch Inactivity Timer, and enter Negotiating In Full state?	M		Table 5-12 Ds15	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP50	When in Exchanging In Full state and Exchange Done event occurs, is the DS Rxmt Timer stopped, PTSE Request packets sent, and the Loading In Full state entered?	M		Table 5-12 Ds3	Yes__No__
RPP51	When in Exchanging In Full state and Synch Done event occurs, are the DS Rxmt Timer and Resynch Inactivity Timer stopped and the Full state entered?	M		Table 5-12 Ds17	Yes__No__
RPP52	When in Exchanging In Full state and Synch Done(ResynchAllowed) event occurs, are the DS Rxmt Timer and Resynch Inactivity Timer stopped and the Full(Resynch Allowed) state entered?	M		Table 5-12 Ds17	Yes__No__
RPP53	When in Loading In Full state and Loading Done event occurs, are the DS Rxmt Timer and Resynch Inactivity Timer stopped and the Full state entered?	M		Table 5-12 Ds17	Yes__No__
RPP54	When in Loading In Full state and Loading Done(Resynch Allowed) event occurs, are the DS Rxmt Timer and Resynch Inactivity Timer stopped and the Full(Resynch Allowed) state entered?	M		Table 5-12 Ds17	Yes__No__
RPP55	When in Negotiating In Full state and DS Resynch Init event occurs, are the Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if not previously stopped, and the PTSE Request List and all related timers cleared; and the exchange of database summaries with the Synch In Full bit set started over again; and the node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master, and Synch In Full bits set with no PTSE summaries included in this packet; and the DS Rxmt Timer starting and the Database Summary packet retransmitted each DSRxmtInterval time, and Negotiating In Full state entered?	M		Table 5-12 Ds18	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP56	When in Exchanging In Full state and DS Resynch Init event occurs, are the Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if not previously stopped, and the PTSE Request List and all related timers cleared; and the exchange of database summaries with the Synch In Full bit set started over again; and the node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master, and Synch In Full bits set with no PTSE summaries included in this packet; and the DS Rxmt Timer starting and the Database Summary packet retransmitted each DSRxmtInterval time, and Negotiating In Full state entered?	M		Table 5-12 Ds18	Yes__No__
RPP57	When in Loading In Full state and DS Resynch Init event occurs, are the Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if not previously stopped, and the PTSE Request List and all related timers cleared; and the exchange of database summaries with the Synch In Full bit set started over again; and the node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master, and Synch In Full bits set with no PTSE summaries included in this packet; and the DS Rxmt Timer starting and the Database Summary packet retransmitted each DSRxmtInterval time, and Negotiating In Full state entered?	M		Table 5-12 Ds18	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP58	When in Negotiating In Full state and DS Resynch Mismatch event occurs, are the Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if not previously stopped, and the PTSE Request List and all related timers cleared; and the exchange of database summaries with the Synch In Full bit set started over again; and the node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master, and Synch In Full bits set with no PTSE summaries included in this packet; and the DS Rxmt Timer starting and the Database Summary packet retransmitted each DSRxmtInterval time, and Negotiating In Full state entered?	M		Table 5-12 Ds18	Yes__No__
RPP59	When in Exchanging In Full state and DS Resynch Mismatch event occurs, are the Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if not previously stopped, and the PTSE Request List and all related timers cleared; and the exchange of database summaries with the Synch In Full bit set started over again; and the node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master, and Synch In Full bits set with no PTSE summaries included in this packet; and the DS Rxmt Timer starting and the Database Summary packet retransmitted each DSRxmtInterval time, and Negotiating In Full state entered?	M		Table 5-12 Ds18	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP60	When in Loading In Full state and DS Resynch Mismatch event occurs, are the Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped if not previously stopped, and the PTSE Request List and all related timers cleared; and the exchange of database summaries with the Synch In Full bit set started over again; and the node increments the DS sequence number for this neighboring peer, declares itself master (sets the Master bit to one), and starts sending Database Summary packets with the Initialize, More, Master, and Synch In Full bits set with no PTSE summaries included in this packet; and the DS Rxmt Timer starting and the Database Summary packet retransmitted each DSRxmtInterval time, and Negotiating In Full state entered?	M		Table 5-12 Ds18	Yes__No__
RPP61	When in Exchanging state and Synch Done(Resynch Allowed) event occurs, is the DS Rxmt Timer stopped and the Full(Resynch Allowed) state entered?	M		Table 5-12 Ds4	Yes__No__
RPP62	When in Loading state and Loading Done(Resynch Allowed) event occurs, is the DS Rxmt Timer stopped and the Full(Resynch Allowed) state entered?	M		Table 5-12 Ds4	Yes__No__
RPP63	When in Loading state and DS Resynch Init event occurs, is the Resynch Retry Timer restarted?	M		Table 5-12 Ds19	Yes__No__
RPP64	When in Full state and DS Resynch Init event occurs, is the Resynch Retry Timer restarted?	M		Table 5-12 Ds19	Yes__No__
RPP65	When in Loading state and Resynch Retry Timer Expired event occurs, is the Resynch Retry Timer restarted?	M		Table 5-12 Ds19	Yes__No__
RPP66	When in Full state and Resynch Retry Timer Expired event occurs, is the Resynch Retry Timer restarted?	M		Table 5-12 Ds19	Yes__No__
RPP67	When in Negotiating state and Resynch Not Allowed event occurs, does the IUT do nothing?	M		Table 5-12 Ds0	Yes__No__
RPP68	When in Exchangining state and Resynch Not Allowed event occurs, does the IUT do nothing?	M		Table 5-12 Ds0	Yes__No__
RPP69	When in Loading state and Resynch Not Allowed event occurs, does the IUT do nothing?	M		Table 5-12 Ds0	Yes__No__
RPP70	When in Full(Resynch Allowed) state and Resynch Not Allowed event occurs, does the node enter Full state?	M		Table 5-12 Ds0	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP71	When in Negotiating In Full state and Resynch Not Allowed event occurs, does the IUT do nothing?	M		Table 5-12 Ds0	Yes__No__
RPP72	When in Exchanging In Full state and Resynch Not Allowed event occurs, does the IUT do nothing?	M		Table 5-12 Ds0	Yes__No__
RPP73	When in Loading In Full state and Resynch Not Allowed event occurs, does the IUT do nothing?	M		Table 5-12 Ds0	Yes__No__
RPP74	When in Full(Resynch Allowed) state and Drop Port event occurs, and this was the last active link to this neighbor, is the event DropPortLast generated?	M		Table 5-12 Ds9	Yes__No__
RPP75	When in Negotiating In Full state and Drop Port event occurs, and this was the last active link to this neighbor, is the event DropPortLast generated?	M		Table 5-12 Ds9	Yes__No__
RPP76	When in Exchanging In Full state and Drop Port event occurs, and this was the last active link to this neighbor, is the event DropPortLast generated?	M		Table 5-12 Ds9	Yes__No__
RPP77	When in Loading In Full state and Drop Port event occurs, and this was the last active link to this neighbor, is the event DropPortLast generated?	M		Table 5-12 Ds9	Yes__No__
RPP78	When in Full(Resynch Allowed) state and Drop Port Last event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped, if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; and is the NPDown state entered?	M		Table 5-12 Ds10	Yes__No__
RPP79	When in Negotiating In Full state and Drop Port Last event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped, if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; and is the NPDown state entered?	M		Table 5-12 Ds10	Yes__No__
RPP80	When in Exchanging In Full state and Drop Port Last event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped, if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; and is the NPDown state entered?	M		Table 5-12 Ds10	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP81	When in Loading In Full state and Drop Port Last event occurs, are timers: Peer Delayed Ack Timer, DS Rxmt Timer, and Request Rxmt Timer stopped, if running; are the lists: Peer Retransmission, Peer Delayed Acks, and PTSE Request, and related timers cleared; and is the NPDown state entered?	M		Table 5-12 Ds10	Yes__No__
RPP82	When in Negotiating In Full state and DS Rxmt Timer Expired event occurs, does the IUT send the previous Database Summary packet to the neighbor and restart the DS Rxmt timer?	M		Table 5-12 Ds11	Yes__No__
RPP83	When in Exchanging In Full state and DS Rxmt Timer Expired event occurs, does the IUT send the previous Database Summary packet to the neighbor and restart the DS Rxmt timer?	M		Table 5-12 Ds11	Yes__No__
RPP84	When in Exchanging In Full state and Request Rxmt Timer Expired event occurs, does the IUT send a PTSE Request packet containing one or more entries from the PTSE Request List to this neighboring peer, and/or optionally to any other neighboring peers, and restart the Request Rxmt timer?	M		Table 5-12 Ds12	Yes__No__
RPP85	When in Loading In Full state and Request Rxmt Timer Expired event occurs, does the IUT send a PTSE Request packet containing one or more entries from the PTSE Request List to this neighboring peer, and/or optionally to any other neighboring peers, and restart the Request Rxmt timer?	M		Table 5-12 Ds12	Yes__No__
RPP86	When in Full (Resynch Allowed) state and PTSE Rxmt Timer Expired event occurs, does the IUT encapsulate in a PTSP those PTSEs that were last transmitted more than PTSE Retransmission Interval seconds ago and have not yet been acknowledged, transmit that PTSP to the neighboring peer, and restart all related PTSE Retransmission Timers?	M		Table 5-12 Ds13	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP87	When in Negotiating In Full state and PTSE Rxmt Timer Expired event occurs, does the IUT encapsulate in a PTSP those PTSEs that were last transmitted more than PTSE Retransmission Interval seconds ago and have not yet been acknowledged, transmit that PTSP to the neighboring peer, and restart all related PTSE Retransmission Timers?	M		Table 5-12 Ds13	Yes__No__
RPP88	When in Exchanging In Full state and PTSE Rxmt Timer Expired event occurs, does the IUT encapsulate in a PTSP those PTSEs that were last transmitted more than PTSE Retransmission Interval seconds ago and have not yet been acknowledged, transmit that PTSP to the neighboring peer, and restart all related PTSE Retransmission Timers?	M		Table 5-12 Ds13	Yes__No__
RPP89	When in Loading In Full state and PTSE Rxmt Timer Expired event occurs, does the IUT encapsulate in a PTSP those PTSEs that were last transmitted more than PTSE Retransmission Interval seconds ago and have not yet been acknowledged, transmit that PTSP to the neighboring peer, and restart all related PTSE Retransmission Timers?	M		Table 5-12 Ds13	Yes__No__
RPP90	When in Full(Resynch Allowed) state and Peer Delayed Ack Timer Expired event occurs, does the IUT send a PTSE Acknowledgement packet containing all PTSE identifying information items from the Peer Delayed Acks List to the neighboring peer, and delete the acknowledged PTSEs from the Peer Delayed Acks List?	M		Table 5-12 Ds14	Yes__No__
RPP91	When in Negotiating In Full state and Peer Delayed Ack Timer Expired event occurs, does the IUT send a PTSE Acknowledgement packet containing all PTSE identifying information items from the Peer Delayed Acks List to the neighboring peer, and delete the acknowledged PTSEs from the Peer Delayed Acks List?	M		Table 5-12 Ds14	Yes__No__

Item Number	Item Description	Status	Condition for status	Reference	Support
RPP92	When in Exchanging In Full state and Peer Delayed Ack Timer Expired event occurs, does the IUT send a PTSE Acknowledgement packet containing all PTSE identifying information items from the Peer Delayed Acks List to the neighboring peer, and delete the acknowledged PTSEs from the Peer Delayed Acks List?	M		Table 5-12 Ds14	Yes__No__
RPP93	When in Loading In Full state and Peer Delayed Ack Timer Expired event occurs, does the IUT send a PTSE Acknowledgement packet containing all PTSE identifying information items from the Peer Delayed Acks List to the neighboring peer, and delete the acknowledged PTSEs from the Peer Delayed Acks List?	M		Table 5-12 Ds14	Yes__No__
RPP94	If the IUT's internal resources (memory, buffer management, CPU utilization, etc.) are unable to handle a database resynchronization between the neighboring peer and the node, does the IUT set the RCI in PTSPs and PTSE Acknowledgements packets to inhibit that neighbor from attempting a database resynchronization?	M		5.7.10	Yes__No__
RPP95	If the ITU's internal resources become available and a database resynchronization can be performed safely between the neighboring peer and the node, does the IUT set the RCI to not congested in PTSPs and PTSE Acknowledgements transmitted to a neighboring peer?	M		5.7.10	Yes__No__
RPP96	If the Neighboring Peer Congestion Status is set to Congested, does the IUT then NOT attempt the database resynchronization but starts the Resynch Retry Timer, if it is not already running?	M		5.7.10	Yes__No__
Comments:					