INTERNATIONAL STANDARD

ISO/IEC 10026-3

First edition
1992-12-15

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Information technology — Open Systems Interconnection — Distributed Transaction Processing —

Part 3:
Protocol specification

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — Traitement transactionnel réparti —

Partie 3: Spécification du protocole

Reference number
ISO/IEC 10026-3:1992(E)

Contents

Introduction x
1 Scope 1
2 Normative references 1
3 Definitions 2
4 Abbreviations 2
5 Conventions 3
6 Model of the PM 3
6.1 Overview 3
6.1.1 Principles of association usage 3
6.1.2 Dialogue establishment 4
6.1.3 Channel management 4
6.1.4 Channel utilization 5
6.1.5 Token control 5
6.1.6 Concatenation/separation 5
6.1.7 Embedding 6
6.2 OSI TP Protocol structure 6
6.2.1 Components of the PM 6
6.2.1.1 The TPPM 7
6.2.1.2 The CPM 8
7 Execution Rules 8
7.1 Operation of the PM 8
7.1.1 Relationship of SAO(s) to MACF(s) 8
7.1.2 Input events to the PM 9
7.1.3 Action sequences 9
7.1.4 SACF queuing 9
7.1.5 Input event blocking at the PSAP 9
7.1.6 PM error conditions 10
7.2 Procedure rules 10
7.3 Definitions 12
7.4 Log records used by the PM 15
7.4.1 Log-ready record 15
7.4.2 Log-commit record 15
7.4.3 Log-heuristic record 15
7.4.4 Log-damage record 15
7.5 Recovery-context-handle 15
8 Use of ACSE, CCR and the Presentation Layer 15
8.1 Introduction 15
8.2 Use of ACSE Service primitives 15
8.2.1 Use of the A-ASSOCIATE parameters 16
8.2.2 Use of the A-RELEASE parameters 16
8.2.3 Use of the A-ABORT and A-P-ABORT parameters 16
8.3 Use of CCR Service primitives 17
8.4 Use of the Presentation Layer 17
8.4.1 Use of Presentation Service primitives 17
8.4.2 Mapping of C-ROLLBACK-RI to Presentation 18
8.5 Association Management 18
8.5.1 Introduction 18
8.5.2 Association/dialogue compatibility 18
8.5.3 Association/channel compatibility 18
8.5.4 Initiating an association establishment 18
8.5.5 Receiving an association establishment indication 19
8.5.6 Responding to association establishment 19

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

© ISO/IEC 1992
All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

8.5.7 Receiving confirmation of association establishment 20
8.5.8 Initiating an association release 20
8.5.9 Aborting an association 20
9 TP-ASE description 21
9.1 Introduction 21
9.2 AF Service Definition 21
9.2.1 AF-BEGIN-DIALOGUE 21
9.2.2 AF-BID 21
9.2.3 AF-END-DIALOGUE 22
9.2.4 AF-U-ERROR 22
9.2.5 AF-ABORT 22
9.2.6 AF-GRANT-CONTROL 22
9.2.7 AF-REQUEST-CONTROL 22
9.2.8 AF-HANDSHAKE 22
9.2.9 AF-HANDSHAKE-AND-GRANT-CONTROL 22
9.2.10 AF-DEFER 22
9.2.11 AF-PREPARE 22
9.2.12 AF-HEURISTIC-REPORT 22
9.2.13 AF-ABORT-AND-HEURISTIC-REPORT 22
9.2.14 AF-RECOVER 23
9.2.15 AF-TOKEN-GIVE 23
9.2.16 AF-TOKEN-PLEASE 23
9.3 AF-Services and TP APDUs: parameters and fields Mappings 23
9.3.1 AF-BEGIN-DIALOGUE request/indication/response/confirm, TP-BEGIN-DIALOGUE-RI/-RC APDU 23
9.3.2 AF-BID request/indication/response/ confirm, TP-BID-RI/-RC APDU 25
9.3.3 AF-END-DIALOGUE request/ indication/response/confirm, TP-END-DIALOGUE-RI/-RC APDU 25
9.3.4 AF-U-ERROR request/indication/ response/confirm, TP-U-ERROR-RI/-RC APDU26
9.3.5 AF-ABORT request/indication, TP-ABORT-RI APDU 26
9.3.6 AF-GRANT-CONTROL request/ indication, TP-GRANT-CONTROL-RI APDU 26
9.3.7 AF-REQUEST-CONTROL request/ indication, TP-REQUEST-CONTROL-RI APDU 26
9.3.8 AF-HANDSHAKE request/indication/ response/confirm, TP-HANDSHAKE-RI/-RC APDU 26
9.3.9 AF-HANDSHAKE-AND-GRANT-CONTROL request/indication/response confirm, TP-HANDSHAKE-AND-GRANT-CONTROL-RI/-RC APDU 27
9.3.10 AF-DEFER request/indication, TP-DEFER-RI APDU 27
9.3.11 AF-PREPARE request/indication, TP-PREPARE-RI APDU 27
9.3.12 AF-HEURISTIC-REPORT request/ indication, TP-HEURISTIC-REPORT-RI APDU 27
9.3.13 AF-ABORT-AND-HEURISTIC-REPORT request/indication 28
9.3.14 AF-RECOVER request/indication, TP-RECOVER-RI APDU 28
9.3.15 AF-TOKEN-GIVE request/indication, TP-TOKEN-GIVE-RI APDU 28
9.3.16 AF-TOKEN-PLEASE request/ indication, TP-TOKEN-PLEASE-RI APDU 29
9.4 Procedures 29
9.4.1 AF-BEGIN-DIALOGUE request 29
9.4.2 TP-BEGIN-DIALOGUE-RI TP APDU 29
9.4.3 AF-BEGIN-DIALOGUE response 29
9.4.4 TP-BEGIN-DIALOGUE-RC TP APDU 29
9.4.5 AF-BID request 29
9.4.6 TP-BID-RI TP APDU 29
9.4.7 AF-BID response 29
9.4.8 TP-BID-RC TP APDU 29
9.4.9 AF-END-DIALOGUE request 29
9.4.10 TP-END-DIALOGUE-RI TP APDU 29
9.4.11 AF-END-DIALOGUE response 29
9.4.12 TP-END-DIALOGUE-RC TP APDU 29
9.4.13 AF-U-ERROR request 29
9.4.14 TP-U-ERROR-RI TP APDU 29
9.4.15 AF-U-ERROR response 29
9.4.16 TP-U-ERROR-RC TP APDU 29

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

9.4.17 AF-ABORT request....30
9.4.18 TP-ABORT-RI TP APDU....30
9.4.19 AF-GRANT-CONTROL request....30
9.4.20 TP-GRANT-CONTROL-RI TP APDU....30
9.4.21 AF-REQUEST-CONTROL request....30
9.4.22 TP-REQUEST-CONTROL-RI TP APDU....30
9.4.23 AF-HANDSHAKE request....30
9.4.24 TP-HANDSHAKE-RI TP APDU....30
9.4.25 AF-HANDSHAKE response....30
9.4.26 TP-HANDSHAKE-RC TP APDU....30
9.4.27 AF-HANDSHAKE-AND-GRANT-CONTROL request....30
9.4.28 TP-HANDSHAKE-AND-GRANT-CONTROL-RI TP APDU....30
9.4.29 AF-HANDSHAKE-AND-GRANT-CONTROL response....30
9.4.30 TP-HANDSHAKE-AND-GRANT-CONTROL-RC TP APDU....30
9.4.31 AF-DEFER request....30
9.4.32 TP-DEFER-RI TP APDU....30
9.4.33 AF-PREPARE request....30
9.4.34 C-PREPARE indication....30
9.4.35 AF-HEURISTIC-REPORT request....30
9.4.36 AF-ABORT-AND-HEURISTIC-REPORT request....30
9.4.37 AF-RECOVER request....30
9.4.38 C-RECOVER indication....30
9.4.39 A-ABORT indication....31
9.4.40 C-ROLLBACK indication....31
9.4.41 C-ROLLBACK confirm....31
9.4.42 C-COMMIT indication....31
9.4.43 C-COMMIT confirm....31
9.4.44 C-RECOVER confirm....31
9.4.45 P-TOKEN-GIVE (sync-minor) indication....31
9.4.46 AF-TOKEN-GIVE request....32
9.4.47 P-TOKEN-PLEASE (sync-minor) indication....32
9.4.48 AF-TOKEN-PLEASE request....32
9.5 Mapping....32
10 SACF description....33
10.1 Introduction....33
10.2 SACF states....33
10.3 Service definitions for SAF-DETACH-ASSOCIATION and SAF-ASSOCIATION-LOST....34
10.3.1 SAF-DETACH-ASSOCIATION request....34
10.3.2 SAF-ASSOCIATION-LOST indication....34
10.4 Procedure for SAF-DETACH-ASSOCIATION request....34
10.5 Procedures for TP-ASE, CCR, ACSE, and Presentation Service Primitives....35
10.5.1 AF-BEGIN-DIALOGUE request....35
10.5.2 AF-BEGIN-DIALOGUE indication....35
10.5.3 AF-BEGIN-DIALOGUE response....36
10.5.4 AF-BEGIN-DIALOGUE confirm....36
10.5.5 AF-BID indication....36
10.5.6 AF-BID confirm....37
10.5.7 AF-END-DIALOGUE request....37
10.5.8 AF-END-DIALOGUE indication....37
10.5.9 AF-END-DIALOGUE confirm....38
10.5.10 AF-U-ERROR request....38
10.5.11 AF-U-ERROR indication....38
10.5.12 AF-U-ERROR confirm....38
10.5.13 AF-ABORT request....38
10.5.14 AF-ABORT (provider, abortRI) indication....38
10.5.15 AF-ABORT (user, dataRI) indication....38
10.5.16 A-ABORT request or A-RELEASE (Result=affirmative) response....38
10.5.17 A-[P-]ABORT indication or A-RELEASE (Result=affirmative) confirm....38
10.5.18 AF-GRANT-CONTROL request....39
10.5.19 AF-GRANT-CONTROL indication....39
10.5.20 AF-REQUEST-CONTROL request....39
10.5.21 AF-REQUEST-CONTROL indication....39

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

10.5.22 AF-HANDSHAKE request 39
10.5.23 AF-HANDSHAKE indication 39
10.5.24 AF-HANDSHAKE confirm 39
10.5.25 AF-HANDSHAKE-AND-GRANT-CONTROL request 39
10.5.26 AF-HANDSHAKE-AND-GRANT-CONTROL indication 39
10.5.27 AF-HANDSHAKE-AND-GRANT-CONTROL confirm 39
10.5.28 AF-DEFER request 39
10.5.29 AF-DEFER indication 40
10.5.30 AF-PREPARE request 40
10.5.31 AF-PREPARE indication 40
10.5.32 AF-HEURISTIC-REPORT (commitRC) indication, or AF-HEURISTIC-REPORT (recoverDoneRC) indication 40
10.5.33 C-BEGIN request 40
10.5.34 C-BEGIN indication 40
10.5.35 C-BEGIN confirm 40
10.5.36 C-READY indication 40
10.5.37 C-COMMIT indication or C-COMMIT+C-BEGIN indication 40
10.5.38 AF-ABORT (user, commitRI) indication or AF-ABORT (user, commitRC) indication 41
10.5.39 C-COMMIT confirm 41
10.5.40 AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication 41
10.5.41 C-ROLLBACK request 41
10.5.42 C-ROLLBACK indication 41
10.5.43 AF-ABORT (rollbackRI) indication, AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication, or AF-HEURISTIC-REPORT (rollbackRI) indication 41
10.5.44 C-ROLLBACK confirm, AF-HEURISTIC-REPORT (rollbackRC) indication, AF-ABORT (user/provider, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication 42
10.5.45 AF-RECOVER indication 42
10.5.46 C-RECOVER request or AF-RECOVER request 42
10.5.47 C-RECOVER indication 42
10.5.48 C-RECOVER confirm 42
10.5.49 U-ASE request 42
10.5.50 U-ASE indication 42
10.5.51 AF-TOKEN-GIVE (regular) indication 43
10.5.52 AF-TOKEN-GIVE (keep) indication 43
10.5.53 AF-TOKEN-GIVE (two-way-recovery) request 43
10.5.54 AF-TOKEN-GIVE (two-way-recovery) indication 43
10.5.55 P-TOKEN-GIVE (sync-minor) indication 44
10.5.56 AF-TOKEN-PLEASE request 44
10.5.57 AF-TOKEN-PLEASE indication 44
10.5.58 P-TOKEN-PLEASE indication 44
10.5.59 Protocol error 44
10.5.60 Other service primitives 44
10.6 SACF internal events 44
10.6.1 Unsolicited BID reject 44
10.7 Concatenation 45
10.7.1 Mapping precedence 45
10.7.2 Concatenation rules 45
10.8 Routeing 46
11 MACF description 46
11.1 Introduction 46
11.2 CAF service definition 46
11.2.1 CAF-PLEASE request 46
11.2.2 CAF-GIVE indication 47
11.2.3 CAF-FAIL indication 47
11.2.4 CAF-DETACH request 47
11.2.5 CAF-RECOVER indication 47
11.3 Main procedures 48
11.3.1 TP-BEGIN-DIALOGUE request 48
11.3.2 AF-BEGIN-DIALOGUE indication (TPPM and CPM) 48
11.3.3 TP-BEGIN-DIALOGUE response 49

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

11.3.4 AF-BEGIN-DIALOGUE (accepted) confirm on a Dialogue 49
11.3.5 AF-BEGIN-DIALOGUE (rejected, dataRI) confirm on a Dialogue 50
11.3.6 AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm 50
11.3.7 AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) confirm 51
11.3.8 AF-BEGIN-DIALOGUE confirm (CPM) 51
11.3.9 SAF-ASSOCIATION-LOST indication....52
11.3.10 SAF-ASSOCIATION-LOST indication (CPM) 52
11.3.11 TP-END-DIALOGUE request....52
11.3.12 AF-END-DIALOGUE indication....52
11.3.13 AF-END-DIALOGUE indication (CPM) 54
11.3.14 TP-END-DIALOGUE response....54
11.3.15 AF-END-DIALOGUE confirm 54
11.3.16 TP-U-ERROR request....54
11.3.17 AF-U-ERROR indication 54
11.3.18 AF-U-ERROR confirm....54
11.3.19 TP-U-ABORT request....54
11.3.20 AF-ABORT (user, dataRI) indication 55
11.3.21 Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm on a dialogue 56
11.3.22 Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm on a channel....58
11.3.23 Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm (CPM) 58
11.3.24 TP-GRANT-CONTROL request 58
11.3.25 AF-GRANT-CONTROL indication....58
11.3.26 TP-REQUEST-CONTROL request....58
11.3.27 AF-REQUEST-CONTROL indication 58
11.3.28 TP-HANDSHAKE request....59
11.3.29 AF-HANDSHAKE indication....59
11.3.30 TP-HANDSHAKE response....59
11.3.31 AF-HANDSHAKE confirm 59
11.3.32 TP-HANDSHAKE-AND-GRANT-CONTROL request....59
11.3.33 AF-HANDSHAKE-AND-GRANT-CONTROL indication....59
11.3.34 TP-HANDSHAKE-AND-GRANT-CONTROL response....59
11.3.35 AF-HANDSHAKE-AND-GRANT-CONTROL confirm 60
11.3.36 TP-BEGIN-TRANSACTION request 60
11.3.37 C-BEGIN indication 60
11.3.38 C-BEGIN confirm....61
11.3.39 TP-DATA request 61
11.3.40 U-ASE indication....61
11.3.41 TP-DEFERRED-END-DIALOGUE request....61
11.3.42 TP-DEFERRED-GRANT-CONTROL request....61
11.3.43 AF-DEFER indication 61
11.3.44 TP-PREPARE request....61
11.3.45 TP-COMMIT request....62
11.3.46 AF-PREPARE indication....62
11.3.47 C-READY indication....62
11.3.48 C-COMMIT indication or C-COMMIT+C-BEGIN indication 62
11.3.49 AF-ABORT (user, commitRI) indication....62
11.3.50 TP-DONE request....63
11.3.51 C-COMMIT confirm or AF-HEURISTIC-REPORT (commitRC) indication 63
11.3.52 AF-ABORT (user, commitRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication 64
11.3.53 TP-ROLLBACK request....64
11.3.54 C-ROLLBACK indication or AF-HEURISTIC-REPORT (rollbackRI) indication 64
11.3.55 AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication 64
11.3.56 C-ROLLBACK confirm or AF-HEURISTIC-REPORT (rollbackRC) indication 65
11.3.57 AF-ABORT (user/provider, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

REPORT (rollbackRC) indication ... 66
11.3.58 CAF-RECOVER (ready) indication ... 67
11.3.59 C-RECOVER (ready) indication or AF-RECOVER (ready) indication (CPM) ... 68
11.3.60 CAF-RECOVER (commit) indication ... 68
11.3.61 C-RECOVER (commit) indication ... 69
11.3.62 C-RECOVER (commit) indication or AF-RECOVER (commit) indication (CPM) 69
11.3.63 C-RECOVER (done) confirm or AF-HEURISTIC-REPORT (recoverDoneRC) indication ... 70
11.3.64 C-RECOVER (unknown) confirm ... 70
11.3.65 C-RECOVER (unknown) confirm (CPM) ... 70
11.3.66 C-RECOVER (retry-later) confirm ... 70
11.3.67 C-RECOVER (retry-later) confirm (CPM) ... 70
11.3.68 AF-TOKEN-GIVE (two-way-recovery) indication on a channel (TPPM) ... 70
11.3.69 AF-TOKEN-GIVE indication (CPM) ... 70
11.3.70 AF-TOKEN-PLEASE indication on a channel (TPPM) ... 71
11.3.71 AF-TOKEN-PLEASE indication (CPM) ... 71
11.3.72 CAF-PLEASE request (CPM) ... 71
11.3.73 CAF-GIVE indication ... 71
11.3.74 CAF-FAIL indication ... 72
11.3.75 CAF-DETACH request (CPM) ... 72
11.4 Internal event procedures ... 72
11.4.1 Delay recovery ... 72
11.4.2 Heuristic damage compensation for subtree ... 72
11.4.3 Restart after node crash (CPM) ... 72
11.4.4 Retry recovery ... 72
11.4.5 Taking a heuristic decision ... 73
11.4.6 Terminating a channel (CPM) ... 73
11.4.7 TPPM creation after node crash ... 73
11.4.8 TPPM-initiated rollback ... 73
11.5 Common Procedures ... 73
11.5.1 Completing commitment ... 73
11.5.2 Entering READY state ... 75
11.5.3 First request/response ... 75
11.5.4 Initiating a transaction branch ... 75
11.5.5 Initiating rollback at TPPM ... 76
11.5.6 Initiating transaction after rollback ... 77
11.5.7 Making commitment decision ... 77
11.5.8 Receiving commit order ... 78
11.5.9 Recording the heuristic condition ... 78
11.5.10 Reporting rollback to superior ... 79
11.5.11 Rollback next transaction ... 80
11.5.12 Sending commit order ... 80
12 Structure and encoding of TP APDUs ... 82
12.1 Abstract syntax of the TP-ASE APDUs ... 82
12.2 Rules of extensibility ... 88
13 Conformance ... 88
13.1 Static conformance requirements ... 88
13.1.1 Conformance classes ... 88
13.1.1.1 General requirements ... 88
13.1.1.2 Application transaction branches class ... 88
13.1.1.3 Chained provider-supported transaction branches class ... 88
13.1.1.4 Unchained provider-supported transaction branches class ... 88
13.1.2 Capabilities ... 88
13.1.2.1 General capabilities ... 88
13.1.2.2 Constraints ... 89
13.1.3 Functional units ... 89
13.1.3.1 Definition ... 89
13.1.3.2 Description of recovery functional unit ... 89
13.1.3.3 Requirements on TP APDUs ... 89
13.1.4 Dependencies on other standards ... 91
13.2 Dynamic conformance requirements ... 91
13.2.1 General requirements ... 91

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

13.2.2 Specific requirements 92
13.3 Protocol Implementation Conformance Statement 92
13.4 Receiving TP APDUs 92
14 Compliance 92
15 Precedence statement 92
16 Index of Actions and Events 93
Annexes
A OSI TP protocol — State tables 97
B Requirements for writing U-ASEs and application contexts 207
C Scenarios 209
D Summary of assigned object identifier values 315
E Recovery from destruction of atomic action data 317
F TPPM transaction states 319
G Estelle description of the TP protocol 323
H LOTOS description of the TP protocol 509

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 10026-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

ISO/IEC 10026 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — Distributed Transaction Processing*:

— *Part 1: OSI TP Model*

— *Part 2: OSI TP Service*

— *Part 3: Protocol specification*

— *Part 4: Protocol implementation conformance statement (PICS) proforma*

— *Part 5: Application context proforma*

— *Part 6: Unstructured data transfer*

Annexes A and B form an integral part of this part of ISO/IEC 10026. Annexes C, D, E, F, G and H are for information only.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Introduction

ISO/IEC 10026, Distributed Transaction Processing (OSI TP), is one of a set of standards produced to facilitate the interconnection of computer systems. It is related to other International Standards in the set as defined by the Reference Model for Open Systems Interconnection (ISO 7498). The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

The aim of Open Systems Interconnection (OSI) is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of computer systems:

a) from different manufacturers;

b) under different management;

c) of different levels of complexity; and

d) of different technologies.

ISO/IEC 10026 defines an OSI TP Model, an OSI TP Service and specifies an OSI TP Protocol available within the Application Layer of the OSI Reference Model.

The OSI TP Service is an Application Layer service. It is concerned with identifiable information which can be related as transactions, which may involve two or more Open Systems.

ISO/IEC 10026 provides sufficient facilities to support transaction processing, and establishes a framework for coordination across multiple TP resources in separate open systems.

ISO/IEC 10026 does not specify the interface to local resources, nor does it specify an application programming interface within the local system.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 3: Protocol specification

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

1 Scope

This part of ISO/IEC 10026 provides

a) a statement (clauses 6 to 11) of the nature of the automaton giving the necessary behaviour of each of the participating entities which are providing the OSI TP Service, covering

1) the actions to be taken on receiving request and response primitives issued by a TP Service user invocation;

2) the actions to be taken on receiving indication and confirm primitives issued by the presentation service-provider;

3) the actions to be taken as a result of certain events within the local system;

4) the actions to be taken as a result of interactions with other ASEs;

b) the definition (clause 12) of the abstract syntax required to convey the TP protocol control information;

c) the conformance requirements to be met by implementations of this protocol (clause 13).

The scope of this part of ISO/IEC 10026 is limited to the interconnection of systems; it does not specify or restrict the implementation of possible interfaces within a computer system.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10026. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 10026 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 7498 : 1984, *Information processing systems - Open Systems Interconnection - Basic Reference Model.*

ISO 7498-2 : 1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security architecture.*

ISO 7498-3 : 1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 3: Naming and addressing.*

ISO 8326 : 1987, *Information processing systems - Open Systems Interconnection - Basic connection oriented session service definition.*

ISO 8326 : 1987/Amd 4: —[1], *Information technology - Open Systems Interconnection - Basic connection oriented session service definition - Amendment 4: Additional synchronization functionality.*

ISO 8327 : 1987/Amd 3: —[1], *Information processing systems - Open Systems Interconnection - Basic connection oriented session protocol specification. - Amendment 3:*

1) To be published.

Additional synchronization functionality.

ISO/TR 8509 : 1987, *Information processing systems - Open Systems Interconnection - Service conventions.*

ISO 8649 : 1988, *Information processing systems - Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO 8650 : 1988, *Information processing systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element.*

ISO 8822 : 1988, *Information processing systems - Open Systems Interconnection - Connection oriented presentation service definition.*

ISO 8822 : 1988/Amd 5: —[1], *Information technology - Open Systems Interconnection - Connection oriented presentation service definition - Amendment 5: Additional synchronization functionality.*

ISO 8823 : 1988/Amd 5: —[1], *Information technology - Open Systems Interconnection - Connection oriented presentation protocol specification - Amendment 5: Additional synchronization functionality.*

ISO/IEC 8824 : 1990, *Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825 : 1990, *Information technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO/IEC 9072-1 : 1989, *Information processing systems - Text communication - Remote Operations - Part 1: Model, notation, and service definition.*

ISO/IEC 9072-2 : 1989, *Information processing systems - Text communication - Remote Operations - Part 2: Protocol Specification.*

ISO/IEC 9545 : 1989, *Information technology - Open Systems Interconnection - Application Layer Structure.*

ISO/IEC 9579-1 : —[1], *Information technology - Open Systems Interconnection - Remote database access - Part 1: Generic model, service, and protocol.*

ISO/IEC 9579-2 : —[1], *Information technology - Open Systems Interconnection - Remote database access - Part 2: SQL specialization.*

ISO/IEC 9594-2 : 1990, *Information technology - Open Systems Interconnection - The Directory - Part 2: Models.*

ISO/IEC 9594-6 : 1990, *Information technology - Open Systems Interconnection - The Directory - Part 6: Selected Attribute Types.*

ISO/IEC 9646-1 : 1991, *Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts.*

ISO/IEC 9804 : 1990, *Information technology - Open Systems Interconnection - Service definition for the Commitment, Concurrency and Recovery service element.*

ISO/IEC 9804/Amd 2: —[1], *Information technology - Open Systems Interconnection - Service definition for the Commitment, Concurrency and Recovery service element - Amendment 2: Session mapping changes.*

ISO/IEC 9805 : 1990, *Information technology - Open Systems Interconnection - Protocol specification for the Commitment, Concurrency and Recovery service element.*

ISO/IEC 9805/Amd 2: —[1], *Information technology - Open Systems Interconnection - Protocol specification for the Commitment, Concurrency and Recovery service element - Amendment 2: Session mapping changes.*

3 Definitions

For the purposes of this part of ISO/IEC 10026, the definitions given in ISO/IEC 10026-1 (TP Model) and ISO/IEC 10026-2 (TP Service), in addition to those given in 7.3 of this part of ISO/IEC 10026, apply.

Definitions of terms specific to the OSI TP protocol specification are contained in 7.3.

4 Abbreviations

Abbreviations used in the OSI TP protocol specifications are defined in ISO/IEC 10026-1 (OSI TP model), except for the following, which are used in some tables:

cnf confirm primitive

ind indication primitive

req request primitive

rsp response primitive

and for the following, which are used as prefixes to auxiliary facilities services:

1) To be published.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

AF Auxiliary Facility

CAF Channel Auxiliary Facility

SAF SACF Auxiliary Facility

5 Conventions

ISO/IEC 10026-2 defines services for Distributed Transaction Processing guided by the descriptive conventions defined in ISO/TR 8509.

However, the terms "request" and "indication" are sometimes used in the following ways:

a) a single request may result in multiple indications (an example is that a single TP-COMMIT request can result in TP-PREPARE indications to each direct subordinate TPSUI);

b) several requests may result in a single indication (an example is that a single TP-COMMIT-COMPLETE indication may be issued to a superior TPSUI only after TP-DONE requests have been issued by the TPSUI and all subordinate TPSUIs in the transaction tree);

c) the convention that a request primitive results in an indication primitive of the same name is not always followed (for example, a TP-COMMIT request will cause a TP-PREPARE indication to be issued).

For a given primitive or APDU, the presence of each parameter or field is described by one of the following values:

blank: not applicable
M: presence is mandatory
U: presence is a user option
O: presence is a provider option
C: presence is conditional

In addition, the notation (=) indicates that a parameter or field value is semantically equal to the value of the parameter or field of the preceding primitive or APDU in the table. This notation is in some instances combined with another value above, e.g. "(=)/M", and signifies that in some cases the primitive follows as a result of a preceding primitive or APDU (that is, "(=)" applies) and in other cases (when "M" applies), either (i) there is no preceding primitive or APDU, or (ii) the value from the preceding primitive or APDU can be changed.

6 Model of the PM

6.1 Overview

This clause provides an overview of those aspects of the TPPM which are specific to this part of ISO/IEC 10026. These include association usage and management, the details of dialogue establishment and channel management, the use of the Session synchronize-minor token, concatenation, and embedding.

6.1.1 Principles of association usage

An association is used by a TPPM to support either a

— TP dialogue; or

— TP Channel.

An association may be established at any time, according to a local decision. The setting up of an association may be done in parallel with the actions of the PM. An association that has been established and is not currently being used is considered to be in a pool of free associations.

On receipt of a dialogue request, the PM needs to be assigned an association to support this dialogue. Any association that is assigned must have attributes compatible with the dialogue it is to be used for, as described in 8.5.2 (dialogues) and 8.5.3 (channels).

An association may be assigned to the PM from the pool of free associations, or attempts may be made to establish a new association for use with this dialogue or channel. If, as a local decision, it is decided that a compatible association cannot be assigned, the begin dialogue request will be rejected.

Associations may be released at any time they are not in use by the TPPM. The point at which an association becomes unused, and therefore may be released, is defined in the SACF procedures in clause 10.

On the establishment of an association, one AEI is assigned to be the "contention-winner" and the other as the "contention-loser". The assignment of contention-winner and contention-loser remains for the duration of the association. An AEI may be the contention-winner on some associations and contention-loser on others.

The direction from the contention-winner to the contention-loser is the preferred direction of dialogue establishment because the contention-winner has the right of use of the association. The contention-winner may grant to the contention-loser the use of the association for the purpose of establishment of a dialogue, if it is not using or has not reserved this association. The contention-winner may also deny the use of the association by the contention-loser for the purpose of dialogue establishment.

The contention-loser may formally request the rights of the contention-winner temporarily in order to attempt establishment of a single dialogue. This is done using the bid mechanism. The use of the bid mechanism is declared to be either optional or mandatory at association

CONFORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

establishment time. This declaration does not change for the life of the association.

6.1.2 Dialogue establishment

When a TPSUI attempts to establish a new dialogue, a compatible association must first be assigned for use with this dialogue as described above in 6.1.1.

Should two TPPMs (at different AEIs) attempt to establish a dialogue over the same association (without using the bid mechanism), the TPPM at the contention-winner AEI will succeed, disrupting the attempt from the contention-loser.

Use of the bid mechanism affects dialogue establishment. The contention-loser may request the right to establish a single dialogue without the possibility of a conflict by issuing a bid request. The contention-winner may accept or reject the bid request. If accepted, the contention-loser issues the dialogue establishment request. If rejected, the contention-loser may not issue a dialogue establishment request until after the receipt of a dialogue establishment request from the contention-winner.

Bidding by the contention-loser before attempting to establish a dialogue is mandatory in either of the following cases:

a) if bidding is mandatory for the association;

b) if the condition exists where an unexpected C-BEGIN indication may appear (see 10.3.1).

A dialogue establishment request may be rejected for the following reasons:

a) the TPPM is unable to select or establish an association which meets the requirements of the dialogue;

b) the TPPM is a contention-loser on the association and its dialogue establishment or bid request collides with a dialogue establishment request from the contention-winner. It is worthwhile to distinguish two types of collisions:

1) the contention-winner is still within a dialogue;

2) the contention-winner is not within a dialogue. This can happen if the contention-winner begins a dialogue and ends it with no response required from the partner;

c) the partner TPPM or TPSUI rejects the dialogue. This can happen for a variety of reasons: TPSU title not found, insufficient resources, etc. The reason for the rejection is carried on a TP APDU.

NOTE — Although this part of ISO/IEC 10026 specifies that the dialogue establishment is rejected, this does not preclude an implementation from attempting to retry the dialogue establishment.

Because of the use of unconfirmed dialogue or channel termination, it may happen that "stray APDUs" arrive from the partner after a dialogue establishment request. To detect and discard these stray APDUs, a correlator value is sent on the dialogue establishment request, which is returned by the partner at dialogue establishment confirmation time. As dialogue establishment confirmation always precedes any other request issued by the partner, APDUs received before that confirmation are discarded. (The same mechanism is used during channel establishment.)

The mechanism for detecting the dialogue reject situation described in "b) 2)" above is the use of a "last partner identifier" (LPI). When the contention-loser issues a bid request or a dialogue establishment request without a bid request, the request carries the correlator of the previous dialogue establishment indication received from the contention-winner. If the contention-winner receives an LPI with a value different from that of the correlator on the previously issued dialogue establishment request, the bid (or dialogue establishment) request is rejected. The LPI is not provided if there was no previous dialogue establishment indication from the contention-winner.

6.1.3 Channel management

After a node crash or a communications failure, a TPPM may (depending on the transaction state as recorded in a log record found for that transaction) be responsible for recovery. In order to meet this requirement, the TPPM needs a channel for the purpose of recovery. The establishment of channels is in most respects similar to the establishment of dialogues; there is, however, one essential difference: unlike dialogues, channels are not established by the TPPMs themselves, but rather they are established and managed by a channel protocol machine (CPM). There is only a single CPM per AEI and this CPM deals with the channels requested and used by all the TPPMs residing at this AEI.

The interactions between a TPPM and the CPM are modeled by the CAF-service (CAF for Channel Auxiliary Facility). A TPPM uses a CAF service request to request the CPM to establish a channel to a specific partner TPPM. Upon receipt of this request, the CPM either selects an existing channel or establishes a new channel using a procedure similar to that of dialogue establishment (see 6.1.2).

When recovery has been performed, i.e., when a TPPM either has issued a C-RECOVER response or received a C-RECOVER confirm, the TPPM uses another CAF service request in order to inform the CPM that it has no further use for the channel. The CPM then may either terminate the channel or keep it for subsequent use.

Besides satisfying the requests for channels that are issued by TPPMs residing at its AEI, the CPM is also responsible

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

for responding to all channel establishment indications addressed to its AEI and issued by other CPMs. Moreover, the CPM receives all recovery initiating indications on a channel and directs them to the TPPMs to which they are addressed; whenever no such TPPM may be located, the CPM must respond to the recovery initiating indication.

6.1.4 Channel utilization

A channel is established as either a one-way-recovery channel or a two-way-recovery channel. With either type of channel, due to restrictions in ISO/IEC 9804, only a single C-RECOVER request is allowed to be outstanding on a channel until it has been responded to; moreover the issuer of a C-RECOVER request must own the token unless the C-RECOVER request is issued in response to a C-RECOVER indication (C-RECOVER (commit) request in response to a C-RECOVER (ready) indication) or in some circumstances on a two-way-recovery channel.

On a one-way-recovery channel, only the initiator of the channel has the right to initiate recovery. The token, once owned by the initiator, is never transferred to the partner.

On a two-way-recovery channel, either side of the channel may initiate recovery, provided that it owns the token. The token is transferred to the partner after each C-RECOVER request or AF-RECOVER request; this allows interleaving of recovery exchanges over the channel. When a CPM does not need to initiate recovery any longer, it may transfer the token to its partner; on the other hand, if the side that does not own the token wants to initiate recovery on a channel, it may issue an AF-TOKEN-PLEASE request.

6.1.5 Token control

NOTE 1 See annex B for U-ASE use of tokens.

CCR requires the Session Layer synchronize-minor token (hereinafter called the token—see 7.3) to be owned when beginning a transaction, committing a transaction, or initiating recovery. The TPPM guarantees that the token will be available at the appropriate times in the absence of the movement of the token by the TPSUI or U-ASE. The TPPM uses the following rules for moving the token:

a) the token is owned by the contention-winner when an association is established;

b) the token is returned to the contention-winner at the termination of the dialogue;

c) if the token is received by a contention-loser while the association is not assigned to a dialogue it is returned to the contention-winner. This rule does not hold if the contention-loser is attempting to establish a dialogue and has received a confirmation that a bid request was accepted;

NOTE 2 This happens when a U-ASE request to move the token collides with an unconfirmed dialogue termination request.

d) the token is moved to the contention-loser upon acceptance of a bid request carrying a parameter requesting the token;

e) the token is moved to the contention-loser upon receipt of a dialogue establishment indication (without prior bidding) selecting the Commit functional unit if the contention-winner owns the token and has not reserved the association for other use. If the token is not owned by the contention-winner, it will eventually arrive and then be sent back to the contention-loser;

NOTE 3 This mechanism ensures that the token is always at the dialogue superior when the Commit functional unit is selected to enable the dialogue superior to begin a transaction.

f) if the token is owned by the subordinate after the receipt of the prepare request, the token is sent to the superior when sending the commitment offer.

g) on a two-way-recovery channel, the token is sent to the partner after initiating each recover request.

There are some cases where the token may not be immediately available when required for beginning a transaction or initiating recovery due to the movement of tokens by previous dialogues or other factors. In these cases, except for the case when the U-ASE moves the token within the same dialogue and the token is needed for beginning a transaction, the rules guarantee that the token will eventually arrive, so the TPPM simply waits until it does. A U-ASE that moved the token prior to beginning a transaction is responsible for getting it if it does not have it.

6.1.6 Concatenation/separation

Concatenation is an optional feature which allows multiple APDUs generated by the TPPM (this includes TP, CCR, ACSE, and U-ASE APDUs) to be mapped onto a single PSDU, reducing the number of PSDUs and optimizing performance. Concatenation is performed by the concatenator part of the SACF and is not included in action sequences (the remainder of the SACF is included in action sequences—see 7.1.3).

When concatenation is not used, the state of the supporting layers is always synchronized with the state of the TPPM because of the one-to-one correspondence between PSDUs (which affect the state of the supporting layers) and APDUs (which affect the state of the TPPM). When concatenation is used, it is important to maintain this synchronization between the TPPM and the supporting layers. Therefore, the concatenation rules are constructed such that the APDU directly related to the PSDU causing a state change in the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

supporting layers is always delivered to the TPPM before any other APDUs in the PSDU (which might not be related to the state change in the supporting layers).

NOTE — An example of this is the C-ROLLBACK-RI CCR APDU. If another APDU was concatenated before this APDU, say, a TP-GRANT-CONTROL-RI APDU, then the following scenario would result. The incoming PSDU containing both APDUs would be processed by Session, resulting in a Session state where a P-RESYNCHRONIZE indication was given. The TPPM would, however, process the TP-GRANT-CONTROL-RI APDU first, without seeing the C-ROLLBACK indication. The TPPM could then allow the TPSUI to issue a TP-ROLLBACK request, resulting in an outgoing C-ROLLBACK-RI CCR APDU which is mapped to a P-RESYNCHRONIZE request. This P-RESYNCHRONIZE request is received after the P-RESYNCHRONIZE indication was given previously, resulting in a violation of the Session service-user rules (in some cases). The incoming C-ROLLBACK indication still has not been processed by the TPPM. If nothing could be concatenated before the C-ROLLBACK-RI CCR APDU, then the TPPM would have changed its state to reflect the state change in Session (i.e., process the rollback) before it would accept any other input events from the TPSUI.

Separation involves accepting an incoming PSDU and generating a separate event corresponding to each APDU contained in the PSDU. The entire PSDU is processed before another PSDU is accepted. While concatenation is an optional feature, separation is mandatory; all concatenation sequences that conform to the rules specified in clause 10.7 shall be supported in incoming PSDUs.

6.1.7 Embedding

Where semantics of a combination of APDUs require that they be interpreted as a single unit in order to determine the required action, these APDUs are embedded one within the other. This might arise, for example, when an APDU of the TP-ASE defines added value to a CCR APDU.

NOTE — An example of the use of this principle includes the embedding of TP-PREPARE-RI APDU in C-PREPARE request.

The SACF contains a "router" component which handles TP APDUs embedded in indications and confirms. The router causes a CCR or ACSE indication or confirm not carrying a TP APDU to be directly passed to the MACF (through the applicable SACF procedures). If the CCR or ACSE indication or confirm contains an embedded TP APDU, the router will cause the indication or confirm to be passed to the TP-ASE. The TP-ASE then decodes the TP APDU and passes an AF indication or confirm to the MACF (through the applicable SACF procedures) which expresses the combined semantics of the CCR or ACSE service and TP APDU.

6.2 OSI TP Protocol structure

6.2.1 Components of the PM

The protocol specified in this part of ISO/IEC 10026 provides the services defined by the OSI TP Service, ISO/IEC 10026-2.

The protocol specification for TP is presented as a TP Protocol Machine (TPPM), supplemented by a Channel Protocol Machine (CPM). Within an AEI, there is one CPM with which TPPMs of the same AEI interact for the purpose of recovery.

Interactions between a TPPM and the CPM are represented by a service called the Channel Auxiliary Facility (CAF) service. The CAF service, provided by the CPM to TPPMs, models the ability for channels to be dynamically attached to and detached from a particular TPPM.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

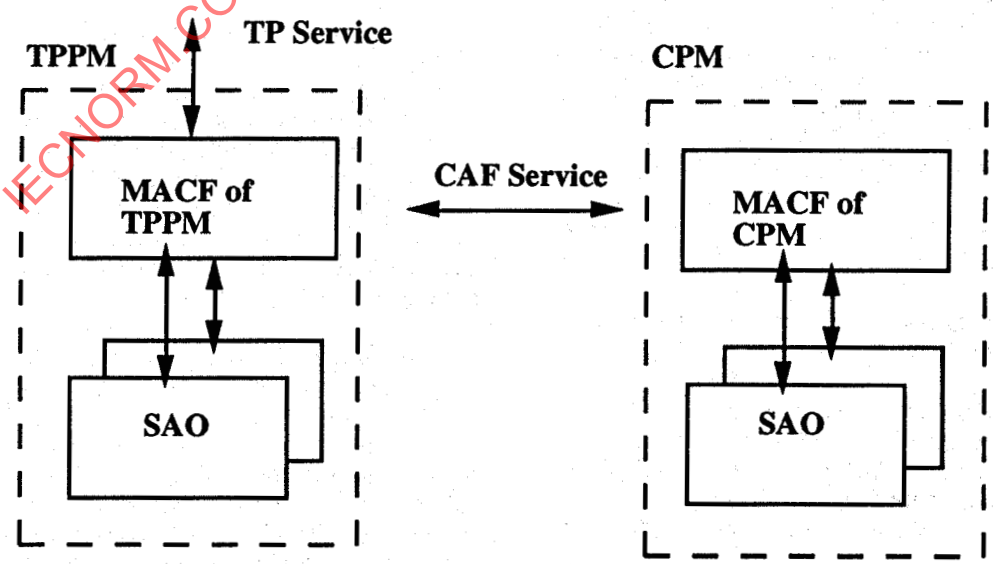
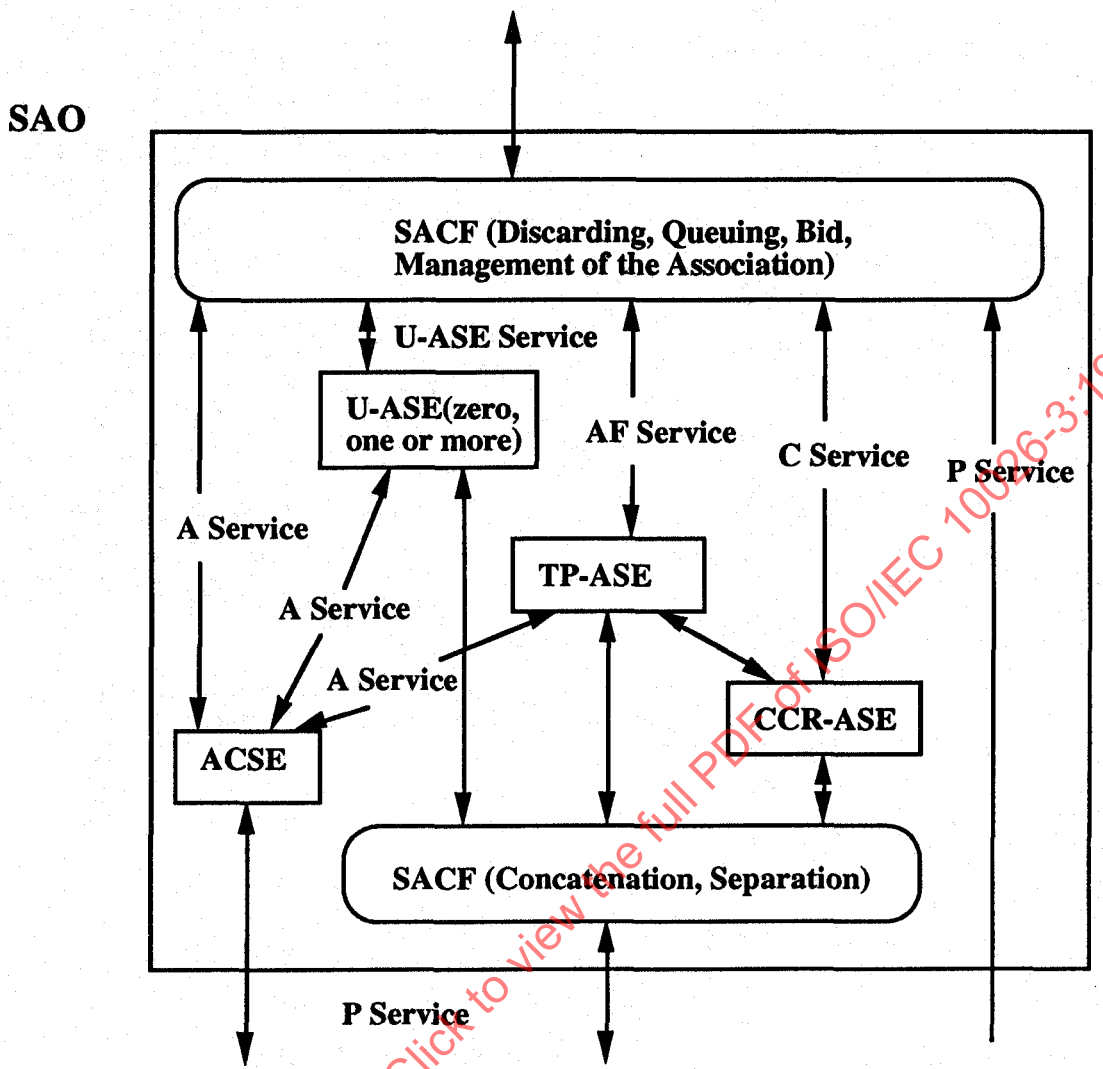


Figure 1 — Relationship between the TPPM and the CPM



NOTE — Only SAOs included in the CPM do not include U-ASE.

Figure 2 — Structure of the SAO

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The TPPM and the CPM comprise a collection of SAOs controlled by a MACF.

The structure of the OSI TP Protocol is shown in Figures 1 and 2.

6.2.1.1 The TPPM

The MACF of the TPPM provides both the TP services over multiple associations and the associated temporal ordering rules. The MACF of the TPPM maps TP services onto the functional capabilities of SAOs included in the TPPM, and onto the CAF service, when appropriate.

Each SAO of a TPPM is composed of

a) ACSE, to establish and terminate associations. The ACSE services are not invoked directly from the TP Service, but are invoked by the MACF of the TPPM (see 6.1.1 and 8.2), or, in some cases, by some source other than the TPPM MACF procedures (see 11.3.21 [11.3.22 in the case of a CPM] and 10.5.59);

b) TP-ASE, to provide for TP APDU generation and reception. Interactions between the TP-ASE and the MACF (through the applicable SACF procedures) of the TPPM are represented by a service called the Auxiliary Facility (AF) service.

The AF service, provided by the TP-ASE, models the ability for TP APDUs to be exchanged and mapped onto appropriate underlying services;

c) CCR, to provide support for commitment, rollback, and recovery functions, when required. CCR services are

used by the MACF of the TPPM either:

1) directly, when no TP APDUs are conveyed by CCR; or,

2) indirectly through the TP-ASE, when TP APDUs are conveyed by CCR (see 8.3);

d) one or more U-ASEs, to provide for application specific protocol. U-ASE services are represented by the TP-DATA service to model TP sequencing constraints on application specific protocols. U-ASE APDUs are mapped onto the Presentation Service, directly or indirectly, as specified by the U-ASE standards;

e) SACF, to coordinate the ASEs in the SAO, in order to maintain a consistent behaviour on the association. The SACF consists of the following components:

1) procedures to examine the services passing between the MACF and SAO and handle actions related to dialogue establishment and managing an association while not in use by a dialogue. These actions can be to allow the service primitive to pass through, discard it, generate additional service primitives, or queue the service primitive. In addition, the procedures provide the SACF Auxiliary Facility (SAF) Service, which represent interactions between the MACF and the SACF;

2) a router which ensures that indications or confirms carrying TP APDUs are directed to the appropriate component. See 6.1.7 on embedding;

3) a concatenator which optionally constructs a single PSDU based on multiple APDUs for outgoing APDUs, and separates each APDU contained in a PSDU for incoming PSDUs.

Both the AF and the SAF services are purely internal to the TPPM: they are not visible outside of the TPPM, and thus are not available for use by anything other than the MACF of the TPPM. The MACF of the TPPM is the only "user" of both the AF and the SAF services. There shall be no conformance requirements to either the AF service or the SAF service.

An application context for an association to be used by TPPM to assign dialogues shall include

a) ACSE;

b) TP-ASE;

c) CCR, when the Commit functional unit is selected;

d) one or more U-ASEs, specifically identified. When multiple U-ASEs are included, the relationships between them shall be specified as part of their specification, or as part of the application context specification; and,

NOTE 1 Annex B contains guidelines for writing U-ASEs eligible for inclusion in a TP application context.

e) SACF and MACF procedures.

NOTE 2 The provision for an application context proforma is a candidate for further standardization as an amendment.

6.2.1.2 The CPM

The MACF of the CPM includes the necessary provisions to establish and terminate channels, as appropriate. The MACF of the CPM provides the CAF service. The MACF of the TPPM is the only "user" of the CAF services. There shall be no conformance requirements to the CAF service.

SAOs included in the CPM are the same as those of a TPPM except that no U-ASE is required. The AF and SAF services used by the MACF of the CPM are identical in their definitions to those of a TPPM. They are subject to the same restrictions.

An application context for an association to be used by a CPM to assign channels shall include:

a) ACSE;

b) TP-ASE;

c) CCR; and,

d) SACF and MACF procedures.

7 Execution Rules

7.1 Operation of the PM

This clause describes the interactions between the components of the PM.

7.1.1 Relationship of SAO(s) to MACF(s)

ISO/IEC 9545 defines a one-to-one correspondence between an association and an SAO. While a dialogue or channel is in use on the association, this SAO is associated with a MACF (of a PM). This part of ISO/IEC 10026 provides for the detaching of the SAO from the MACF at the completion of the dialogue or channel such that the association can subsequently be used by a MACF in another PM. In general, when no dialogue or channel exists on an association, the SAO is detached from the MACF. A detached SAO is not considered to be part of any PM. When an incoming request to establish a dialogue or channel is made, the SAO is then attached to a MACF so that the MACF can process the request. When a MACF initiates the establishment of a dialogue or channel, an SAO that is not attached to any other MACF may be

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

attached to the MACF establishing the dialogue or channel. Alternatively, attempts may be made to create a new SAO to attach to the MACF.

All channels that are not in use for recovery are attached to the CPM. When a request is made of the CPM by a TPPM to perform recovery, the CPM satisfies this request by transferring an unused channel (which is associated with the correct AEI) to the TPPM. When an incoming C-RECOVER indication appears on an unused channel, the CPM finds the TPPM corresponding to the transaction being recovered and transfers the channel to that TPPM. As soon as the TPPM completes recovery, the channel is transferred back to the CPM.

7.1.2 Input events to the PM

PM input events occur either as a result of

— request and response primitives issued by the TPSUI or indication primitives issued by the CPM in the case of the TPPM; a request issued by a TPPM in the case of the CPM;

— events triggered by local conditions (internal events); or

— events caused by APDUs and Presentation indications and confirmations output from the concatenator part of the SACF or directly from Presentation if concatenation is not used.

7.1.3 Action sequences

An action sequence is a contiguous sequence of execution of procedures in clause 9, 10, and 11 as a result of a single input event to either the TPPM or the CPM. The CPM and each TPPM have separate action sequences, which proceed independently. An action sequence includes all processing of input events by the PM, except that of the concatenator part of the SACF (see 6.1.6).

An SAO that is not attached to any MACF is not part of any PM action sequences. It may therefore act in parallel with the actions of the PM.

NOTE — This allows association management to be performed independently of normal PM processing.

An action sequence executes completely (i.e., has finished all procedures) before the PM becomes available for handling any subsequent input events.

An action sequence is a serial and atomic execution of the procedures (except in the event of a node crash), with a single exception where parallelism is allowed: when a request or response is issued to the SAO using the (S)AF, ACSE, CCR, or Presentation services, the SAO may continue this action sequence in parallel with the action sequence of the MACF. The SAO ensures that requests or responses issued by the MACF are processed in the order issued. The SAO completes its part of the action sequence when its procedures are finished.

Within an SAO an A-ABORT request or an A-RELEASE (Result=affirmative) response may be issued. At most one such an event may occur in an action sequence. Such an event is processed by the MACF in order to handle the loss of the association. The event is processed by the MACF in the action sequence in which it was issued once the MACF and all SAOs have completed execution of their procedures (that is, just before the end of the action sequence).

7.1.4 SACF queuing

Certain TP services sometimes cannot be completed in a single action sequence (for example TP-BEGIN-DIALOGUE request). In order to have the TPPM available to accept input events at the TPSUI boundary at all times, it may be necessary to queue requests within the TPPM. Thus, a request that cannot immediately be handled may trigger queuing in the SACF, but is always accepted from the TPSUI. In order to support this, the TPPM describes explicitly queuing operations when they apply.

An example of the need for queuing is that during dialogue establishment for the contention-loser, a bid may be required before the begin dialogue TP APDU is sent. The begin dialogue AF service is therefore queued by the SACF until the bid is complete. While this queuing is taking place, the TPSUI may make further TP service invocations on this dialogue, which will also be queued.

When it becomes necessary to queue, the queuing takes place within the SACF. Certain AF and CCR services are subject to queuing, in that these services may in certain circumstances be queued for a period of time by the SACF before being issued to the TP or CCR ASE. A queue is established for a single dialogue. This queue may subsequently be fully or partially flushed, which causes all or some of the pending service primitives to be processed in a single action sequence, or it may be discarded, which causes the pending service primitives to be discarded. When a queue is fully flushed, any subsequent service primitives are no longer made pending.

7.1.5 Input event blocking at the PSAP

When completing processing for a transaction, APDUs related to a subsequent transaction may be received on an association. The corresponding service primitives cannot be issued to the TPSUI until the transaction is complete.

When service primitives from an association cannot be processed, the TPPM stops accepting service primitives at the PSAP. When processing can resume, the TPPM resumes accepting service primitives at the PSAP including those previously blocked.

NOTE — The concatenator/separator part of the SACF is not prevented from delivering service primitives to the rest

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

of the SAO while service primitives are blocked at the PSAP. These service primitives would be the result of a PSDU already received by the SACF prior to blocking input events at the PSAP. The concatenation rules are such that these incoming events may be processed correctly in that case.

The TPPM stops accepting input events at the PSAP under the following conditions:

a) when a commit confirm has been received from a subordinate and the dialogue has neither been aborted nor had a deferred end dialogue;

b) when a rollback response is issued to or a rollback confirm is received from a subordinate, the Unchained Transactions functional unit is selected, and the dialogue has not been aborted (by the TPSUI);

c) when a C-BEGIN indication is received in the DECIDED (rollback) state and a TP-DONE request is owed;

d) when a rollback confirm is received from or a rollback response is issued to the superior, the Unchained Transactions functional unit is selected, and a TP-DONE request is owed.

The TPPM resumes accepting input events at the PSAP under the following conditions:

a) when a TP-COMMIT-COMPLETE indication is issued, for each subordinate dialogue with a coordination level of "commitment";

b) when a TP-ROLLBACK-COMPLETE indication is issued, for each subordinate dialogue with a coordination level of "commitment" on which the Unchained Transactions functional unit is selected;

c) when a TP-ROLLBACK-COMPLETE indication is issued, for the superior dialogue;

d) when a TP-U-ABORT request is received on a dialogue on which a rollback response was issued or a rollback confirm was received, and the Unchained Transactions functional unit is selected.

7.1.6 PM error conditions

This part of ISO/IEC 10026 defines three types of error conditions which may occur during the operation of a PM:

a) protocol error - a condition as described in 7.2 e). The association is aborted when a protocol error occurs. When a protocol error occurs, the relevant Protocol Error procedure (10.5.59, 11.3.21, 11.3.22, or 11.3.23) is invoked.

b) internal error - a condition where a local decision is made that the operation of a particular dialogue or channel cannot continue normally. When an internal error on a dialogue is detected, the procedure "Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm on a dialogue" (11.3.21) is invoked. When an internal error on a channel owned by the TPPM is detected, the procedure "Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm on a channel" (11.3.22) is invoked. When an internal error on a channel owned by the CPM is detected, the procedure "Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm (CPM)" (11.3.23) is invoked.

NOTE 1 Irregularities in the APDU exchanges between U-ASEs are not visible to the TPPM unless they are intentionally signalled to the MACF; in such a case they may be treated as a TP internal error. Although the U-ASE may define such an irregularity as a protocol error, it is not a protocol error in the scope of ISO/IEC 10026.

c) node crash - a condition where the operation of the PM cannot continue as determined by local considerations. Upon restart after a node crash, the internal event "Restart after node crash" (11.4.3) is triggered.

An internal error or node crash may be triggered by an unexpected event which is received from the TPSUI. Note that rule f) in 7.2 specifies that the procedures assume the operation of the TPSUI is correct according to the service rules, but the state tables in annex A check for violations of these rules. Should such a violation occur, either an internal error or node crash may be triggered, depending on the scope of the violation as determined locally.

NOTES

2 An example of a TPSUI triggering a node crash may occur when an error is made in a TP service which affects all dialogues, such as the TP-COMMIT request.

3 A timer expiring is an example of a condition which may trigger an internal error.

7.2 Procedure rules

The following rules govern the execution of the procedures:

a) Parameter inheritance

The tables of 9.3 use the notation "(=)" to represent matching of parameters/fields values from the values of parameters/fields of the preceding (causal) primitive/APDU. This value matching is assumed implicitly, so that explicit calling out of the setting of

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

these values is not done in the procedures.

b) Procedure atomicity

Each procedure executes atomically, except in the case of a node crash (see 7.1.3).

c) Inspection of transaction-related state

The transaction-related state consists of the transaction state, the node type (that is, root, intermediate, or leaf), and whether the node is a superior, subordinate, or does not exist in the transaction tree. Any inspection of the transaction-related state within the procedures refers to the state at the time of the input event which triggered the current action sequence, with one exception: if the MACF is being re-entered in the same action sequence as a result of an A-ABORT request or an A-RELEASE (Result=affirmative) response, inspection of the transaction-related state refers to its value at the time the MACF is re-entered.

NOTE 1 This handles the situation where the MACF transitioned from the ACTIVE state to the DECIDED (rollback) state, and then the U-ASE issued an A-ABORT request (for example). The state inspection when the MACF is re-entered will not cause a rollback, since the MACF will see the DECIDED (rollback) state and not the ACTIVE state.

d) Procedure sequence

The execution of each action specified in the procedure is assumed to be in the sequence specified in the procedure, unless explicitly stated otherwise.

e) Protocol error detection

A protocol error is declared when one of the following conditions occurs:

1) a procedure, other than an internal event procedure (refer to 10.6 and 11.4) is invoked and

i) no action is taken and no condition is applicable as a result of that procedure invocation; or

NOTE 2 The action "continue" is used in the procedures as a null action, to avoid incurring errors of this type. Other conditions in the same procedure may be applicable, however, resulting in non-null actions being taken.

ii) the procedure precondition is violated. (Preconditions are presented at the beginning of some procedures in the form of declarative sentences.);

2) a TP indication or confirm is issued that violates the constraints specified in ISO/IEC 10026-2;

3) a CCR or ACSE protocol error occurs;

4) a TP, CCR, or ACSE APDU is received by the PM which is either not correct ("invalid", ref. ISO/IEC 9646-1) or is not expected ("inopportune", ref. ISO/IEC 9646-1) in the current state of the PM.

When a protocol error is declared,

1) if the protocol error occurs in the TP-ASE or SACF with no attached MACF, procedure "Protocol error" (10.5.59) is executed;

2) if a TPPM MACF is attached to the association on which the protocol error occurred and a dialogue is active, the procedure "Protocol error or ... on a dialogue" (see 11.3.21) is executed with an indication of a protocol error. As a local decision, this procedure may be invoked on multiple associations, if the protocol error is the result of the attempted issuance of a TP indication or confirm not associated with a particular dialogue;

3) if a TPPM MACF is attached to the association on which the protocol error occurred and a channel is active, the procedure "Protocol error or ... on a channel" (see 11.3.22) is executed with an indication of a protocol error. As a local decision, this procedure may be invoked on multiple associations, if the protocol error is the result of the attempted issuance of a TP indication or confirm not associated with a particular dialogue.

4) if a CPM MACF is attached to the association on which the protocol error occurred, the procedure "Protocol error or ... (CPM)" (see 11.3.23) is executed with an indication of a protocol error.

f) Service rules assumption

Each TP service primitive is assumed to have been received in accordance with the constraints and conditions specified in ISO/IEC 10026-2 and the requirements in annex F.

g) No superior/subordinate assumption

If an action refers to a superior or subordinate, and there is no superior or subordinate (because the node is a root or leaf, respectively), the action is not taken.

h) Bounds of the past

In certain cases, the text refers to a TP or CCR service which has been received. For CCR and TP and related AF services contained in the Commit and Unchained Functional Units, this means that the service primitive was received since the beginning of the current transaction. For all other services, this means that the service primitive was received since the beginning of the current dialogue. In any case, this type of memory of previously received service primitives never survives

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

a node crash.

i) Dialogue assumption

Any dialogue or channel referred to in a procedure is assumed to be the dialogue or channel on which the service primitive was received/issued, unless stated otherwise.

j) Tree topology terminology

Whenever the terms superior, subordinate, root node, intermediate node, or leaf node occur without the qualification "dialogue", they are understood to refer to the transaction tree.

7.3 Definitions

When the following definitions or their logical negations are used in the text, they will appear in italics. AF services used in these definitions are defined in 9.2.

Attach - The specified SAO and association becomes part of the PM. Any indications or confirms made from that SAO are seen by the PM; the PM may direct requests or responses to the SAO.

Close the PSAP - No further PSDUs are accepted at the PSAP unless and until the PSAP is opened (see 7.3, "Open the PSAP").

Commit confirm - One of the following:

— C-COMMIT confirm;

— C-RECOVER(done) confirm;

— AF-HEURISTIC-REPORT (commitRC) indication;

— AF-HEURISTIC-REPORT (recoverDoneRC) indication;

— AF-ABORT (user, commitRC) indication; or

— AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication.

Commit indication - One of the following:

— C-COMMIT indication;

— C-COMMIT+C-BEGIN indication;

— AF-ABORT (user, commitRI) indication;

— C-RECOVER (commit) indication; or

— AF-RECOVER (commit) indication.

Commit request - One of the following:

— C-COMMIT request;

— C-COMMIT+C-BEGIN request;

— AF-ABORT (user, commitRI) request;

— C-RECOVER(commit) request; or

— AF-RECOVER (commit) request.

Detach - The specified SAO and association ceases being part of the PM. The SAO is no longer able to make indications or confirms to any PM; no PM may direct requests or responses to the SAO.

Dialogue has been detached - One of the following service primitives has been issued/received by the PM for the dialogue:

— SAF-DETACH-ASSOCIATION request;

— SAF-ASSOCIATION-LOST indication;

— A-ABORT request;

— A-[P-]ABORT indication;

— A-RELEASE (Result=affirmative) response or confirm.

Dialogue is chaining transaction branches; chaining dialogue - When all of the following conditions are met:

— the dialogue exists;

— the Chained Transactions functional unit was selected;

— no TP-U-ABORT request was received for the dialogue;

— no AF-ABORT, AF-ABORT-AND-HEURISTIC-REPORT, or A[-P]-ABORT indication was received for the dialogue; and

— if the outcome of the transaction is commitment, no TP-DEFERRED-END-DIALOGUE request and no AF-DEFER (end-dialogue) indication has been received for the dialogue.

Dialogue will have a coordination level of "commitment" - a dialogue established with the following:

— the Chained Transactions functional unit; or

— the begin-transaction parameter is set to "true" on the TP-BEGIN-DIALOGUE request or indication.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

NOTE 1 This definition is necessary because the coordination level does not become defined until the TP-BEGIN-DIALOGUE indication is made. This definition is intended only for use during dialogue establishment.

Discard a queue - The SACF discards all queued service primitives.

NOTE 2 This definition is used only when a dialogue is terminated.

Establish a queue - the SACF ensures that the AF, CCR, and Presentation service primitives invoked subsequent to the establishment of the queue, are queued.

Flush a queue - the SAO processes some or all (depending on the context in which the definition is used) of the queued service primitives in the order in which they were queued. If all requests and responses were flushed, the SACF then ensures that any subsequently invoked AF, CCR, ACSE, and Presentation requests and responses are not queued.

NOTE 3 The processing of all requests and responses on the queue is done by the SAO atomically as part of the action sequence in which the queue is flushed.

Forget a transaction - Remove from secure storage the log-ready or log-commit record concerning this transaction, retaining, if any, the log-heuristic and/or the log-damage records.

Last commit confirm was received - When all of the following conditions are met:

— the TPSUI does not *owe a TP-DONE request*;

— a *commit confirm* has been received from each of the subordinates to which a *commit request* was issued.

Last partner identifier is valid - The last-partner-identifier parameter of the most recently received AF-BEGIN-DIALOGUE indication or AF-BID indication is either

— absent or may carry any value, if no AF-BEGIN-DIALOGUE request was issued on this association; or

— has a value equal to the Correlator of the most recently issued AF-BEGIN-DIALOGUE request on this association.

Last ready was received - When all of the following conditions are met:

— a TP-COMMIT request has been received;

— if the TPPM previously started to set the *TPPM bound data* into the ready-to-commit state, the *TPPM bound data* are in the ready-to-commit state; and,

NOTE 4 The TPPM may either atomically set the bound data to the ready-to-commit state when writing the log-ready record, or may start to set the bound data to the ready-to-commit state before writing the log-ready record. This condition applies only to the case where the TPPM previously started to set the bound data to the ready-to-commit state.

— a C-READY indication has been received from each of the subordinates to which an AF-PREPARE request was issued.

Last rollback confirm was received - When all of the following conditions are met:

— the TPSUI does not *owe a TP-DONE request*;

— a *rollback indication* or *rollback confirm* has been received from each of the subordinates whose *dialogue has not been detached*.

Open the PSAP - PSDUs are now accepted at the PSAP.

Rollback confirm - One of the following:

— C-ROLLBACK confirm;

— AF-HEURISTIC-REPORT (rollbackRC) indication;

— AF-ABORT (user/provider, rollbackRC) indication;

— AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication; or

— AF-BEGIN-DIALOGUE (accepted/rejected(user), rollbackRC) confirm.

Rollback indication - One of the following:

— C-ROLLBACK indication;

— AF-HEURISTIC-REPORT (rollbackRI) indication;

— AF-ABORT (user/provider, rollbackRI) indication;

— AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication; or

— AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm.

Rollback reporting has completed - A *rollback response* was issued to the superior or a *rollback confirm* was received from the superior, or there is no superior dialogue.

Rollback request - One of the following:

— C-ROLLBACK request;

— AF-HEURISTIC-REPORT (rollbackRI) request;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— AF-ABORT (user/provider, rollbackRI) request; or

— AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) request.

Rollback response - One of the following:

— C-ROLLBACK response;

— AF-HEURISTIC-REPORT (rollbackRC) request;

— AF-ABORT (user/provider, rollbackRC) request; or

— AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) request.

Token - The Session Layer synchronize-minor token which is required by CCR.

A TP-DONE request is owed - When either or both of the following cases holds:

a) the TPPM has received or issued one or more of the following service primitives without having received a subsequent TP-DONE request:

— a TP-COMMIT indication;

— a TP-ROLLBACK request/indication;

b) the TPPM is in the DECIDED (commit) or DECIDED (rollback) state and has received or issued one or more of the following service primitives on a dialogue with coordination level of "commitment" without having received a subsequent TP-DONE request:

— a TP-P-ABORT indication;

— a TP-U-ABORT request/indication; or

— a TP-BEGIN-DIALOGUE (rejected) confirm.

TPPM bound data - The bound data controlled by the TPPM.

Transaction initiation purging period - A state of a TPPM with respect to a given dialogue which is entered upon receipt of a TP-BEGIN-TRANSACTION request if

a) the Shared Control functional unit is selected; and

b) there is a *user error purging period.*

The *transaction initiation purging period* is terminated as soon as

a) the number of AF-U-ERROR confirms, AF-END-DIALOGUE indications with the Confirmation parameter set to "true", and AF-HANDSHAKE indications received since the reception of the TP-BEGIN-TRANSACTION request equals the number of TP-U-ERROR requests that were outstanding at that time; or

b) a *rollback confirm* is received.

Transfer the channel - The specified SAO is transferred from the PM to another PM. The specified service invocation and any subsequent service invocations made at this SAO are sent/received to/by the PM to which the SAO is transferred.

User error purging period - A state of a TPPM with respect to a given dialogue which is entered upon receipt of a TP-U-ERROR request if

a) the Polarized Control functional unit is selected and the TPPM does not have control and there is no handshake or dialogue termination indication outstanding; or

b) the Shared Control functional unit is selected and there is no handshake or dialogue termination indication outstanding.

The *user error purging period* is terminated as soon as

a) if the Polarized Control functional unit is selected and a TP-HANDSHAKE indication, a TP-GRANT-CONTROL indication, a TP-HANDSHAKE-AND-GRANT-CONTROL indication, or a TP-END-DIALOGUE indication is issued; or

b) if the Shared Control functional unit is selected and the number of AF-U-ERROR confirms, AF-END-DIALOGUE indications with the confirmation parameter set to "true", and AF-HANDSHAKE indications received since the beginning of the *user error purging period* equals the number of TP-U-ERROR requests received during that period; or

c) a *rollback confirm* is received.

Write the log-commit record - The TPPM ensures that the information specified in 7.4.2 will be available even after the occurrence of a node crash. Remove from secure storage the log-ready record concerning this transaction, if one exists.

Write the log-damage record - The TPPM ensures that the information specified in 7.4.4 will be available even after the occurrence of a node crash.

Write the log-heuristic record - The TPPM ensures that the information specified in 7.4.3 will be available even after the occurrence of a node crash.

Write the log-ready record - The TPPM ensures that the information specified in 7.4.1 will be available even after

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

the occurrence of a node crash.

7.4 Log records used by the PM

The transaction identifier defined in ISO/IEC 10026-1 is the atomic action identifier as defined in ISO/IEC 9804. The transaction branch identifier defined in ISO/IEC 10026-1 is the branch identifier as defined in ISO/IEC 9804.

7.4.1 Log-ready record

The log-ready record contains the following information written in secure storage:

a) to identify the transaction:

— atomic action identifier;

b) to identify the branch to the superior:

— branch identifier;

— if provided on the TP-INITIALIZE-RI or -RC APDU, the recovery-context-handle received from the superior;

c) for each subordinate, if any, to identify the subordinate:

— branch identifier;

— subordinate AE Title;

— if provided on the TP-INITIALIZE-RI or -RC APDU, the recovery-context-handle received from the subordinate.

7.4.2 Log-commit record

The log-commit record contains the following information written in secure storage:

a) to identify the transaction:

— atomic action identifier;

b) for each subordinate, to identify the subordinate:

— branch identifier;

— subordinate AE Title;

— if provided on the TP-INITIALIZE-RI or -RC APDU, the recovery-context-handle received from the subordinate.

7.4.3 Log-heuristic record

The log-heuristic record contains the following information written in secure storage:

a) the atomic action identifier;

b) the state of the bound data;

c) information necessary to execute compensating actions, if required.

NOTE — The log-heuristic record models the need for an open system to retain, beyond node crashes, information about the heuristic decision.

7.4.4 Log-damage record

The log-damage record contains the following information written in secure storage:

a) the atomic action identifier;

b) the current known state of the bound data in the node's subtree. Its value is either "heuristic-hazard" or "heuristic-mix".

7.5 Recovery-context-handle

A recovery-context-handle is an identification of a grouping of log records, that will be used for all transactions on a particular association. The use of this grouping and associated recovery-context-handle are optional on any particular association.

NOTE — A possible use of the recovery-context-handle is to allow the partitioning of the set of log records, each part having a different value for the recovery-context-handle.

When the recovery-context-handle is provided by the partner for a given association, its value shall be logged for all transaction branches that are initiated by the partner on this association, and, if initiation of recovery is subsequently required for any of these transaction branches, the value of the recovery-context-handle shall be conveyed on the consequent recovery primitives.

8 Use of ACSE, CCR and the Presentation Layer

8.1 Introduction

This clause identifies the use and requirements made of ACSE, CCR, and the Presentation Layer, and the rules for establishing, assigning, and terminating associations.

8.2 Use of ACSE Service primitives

The Association Control Service Element (ACSE) is used as described in ISO/IEC 8649 to establish and release associations.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 1 — Use of A-ASSOCIATE parameters

A-ASSOCIATE Parameters	Used by TPPM	
	Application-supported Transactions	Provider-supported Transactions or TP Channel
Mode	Yes	Yes
Application Context Name	Yes	Yes
Calling AP Title	Conditional	Yes
Calling AE Qualifier	Conditional	Yes
Calling AP Invocation Identifier	Conditional	Conditional
Calling AE Invocation Identifier	Conditional	Conditional
Called AP Title	Conditional	Conditional
Called AE Qualifier	Conditional	Conditional
Called AP Invocation Identifier	Conditional	Conditional
Called AE Invocation Identifier	Conditional	Conditional
Responding AP Title	Conditional	Yes
Responding AE Qualifier	Conditional	Yes
Responding AP Invocation-Identifier	Conditional	Conditional
Responding AE Invocation-Identifier	Conditional	Conditional
User Information	Yes	Yes
Result	Yes	Yes
Result Source	Yes	Yes
Diagnostic	Conditional	Conditional
Calling Presentation Address	Yes	Yes
Called Presentation Address	Yes	Yes
Responding Presentation Address	Yes	Yes
Presentation Context Definition List	Yes	Yes
Presentation Context Definition Result List	Yes	Yes
Default Presentation Context Name	No	No
Default Presentation Context Result	No	No
Quality Of Service	Yes	Yes
Presentation Requirements	Conditional	Conditional
Session Requirements	Yes	Yes[1)]
Initial Synchronization Point Serial Number	Conditional	Yes
Initial Assignment of Tokens	Conditional	Yes[2)]
Session-Connection Identifier	No	No

1) See 8.5.2 and 8.5.3 for constraints on this parameter.
2) See 8.5.4 for the assignment of the parameter.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The TPPM uses the following ACSE services:

— A-ASSOCIATE and A-RELEASE to establish and release associations; and

— A-ABORT to abruptly release an association.

The TPPM also must react to the occurrence of the A-P-ABORT indication primitive.

8.2.1 Use of the A-ASSOCIATE parameters

The parameters of the A-ASSOCIATE service are used by the PMs as specified in table 1.

8.2.2 Use of the A-RELEASE parameters

The parameters of the A-RELEASE service are used by the PMs as specified in table 2.

Table 2 — Use of A-RELEASE parameters

A-RELEASE Parameters	Used by TPPM
Reason	No
User Information	No
Result	Yes

8.2.3 Use of the A-ABORT and A-P-ABORT parameters

The parameters of the A-ABORT and A-P-ABORT services

are used by the PMs as specified in tables 3 and 4.

Table 3 — Use of A-ABORT parameters

A-ABORT Parameters	Used by TPPM
Abort Source	No
User Information	Yes

Table 4 - Use of A-P-ABORT parameters

A-P-ABORT Parameters	Used by TPPM
Provider Reason	No

8.3 Use of CCR Service primitives

The Commitment, Concurrency and Recovery Service Element (CCR) is used for provider-supported transactions.

The PMs use the following CCR services:

— C-BEGIN, C-PREPARE, C-READY, C-COMMIT and C-ROLLBACK for TP services supported by commitment-related functional units; and

— C-RECOVER for transaction recovery.

The PMs use the User Data parameter of some CCR services to convey certain TP APDUs. These CCR services and TP APDUs are specified in table 30.

The parameters of the CCR services are used by the PMs as specified in tables 5 to 10.

Table 5 — Use of C-BEGIN parameters

C-BEGIN Parameters	Used by TPPM
Atomic Action Identifier - Master's Name	Yes
Atomic Action Identifier - Suffix	Yes
Branch Identifier - Superior's name	Yes
Branch Identifier - Suffix	Yes
User Data	No

Where the TP procedures refer to the "atomic-action-identifier" parameter of C-BEGIN request and indication, this is the combination of the "Atomic Action Identifier - Master's Name" and the "Atomic Action Identifier - Suffix". Where the TP procedures refer to the "atomic-action-branch-identifier" parameter of C-BEGIN request and indication, this is the combination of the "Branch Identifier - Superior's Name" and the "Branch Identifier - Suffix".

Table 6 — Use of C-PREPARE parameters

C-PREPARE Parameters	Used by TPPM
User Data	Yes

Table 7 — Use of C-READY parameters

C-READY Parameters	Used by TPPM
User Data	No

Table 8 — Use of C-COMMIT parameters

C-COMMIT Parameters	Used by TPPM
User Data	Conditional

Table 9 — Use of C-ROLLBACK parameters

C-ROLLBACK Parameters	Used by TPPM
User Data	Conditional

Table 10 — Use of C-RECOVER parameters

C-RECOVER Parameters	Used by (TP)PM
Recovery State	Yes
Atomic Action Identifier	Yes
Branch Identifier	Yes
User Data	Conditional

8.4 Use of the Presentation Layer

8.4.1 Use of Presentation Service primitives

Implementations shall take care that parallelism between the lower layers and the SAO does not result in a violation of the service user rules of those layers.

NOTE — An example of this problem may occur in the case of a rollback. A C-ROLLBACK indication, which is mapped to a P-RESYNCHRONIZE indication, may be processed by the Session layer and not yet processed by the SAO. Before this indication is processed by the SAO, a C-ROLLBACK request, mapped to a P-RESYNCHRONIZE request may be sent to Session, resulting in a violation of the Session Service user rules.

In addition to Presentation Service primitives used by CCR and ACSE Protocol Machines, the TPPM makes use of

— P-TOKEN-PLEASE and P-TOKEN-GIVE for token management (to position the tokens correctly for CCR);

— P-DATA for all other services.

The U-ASE may make use of the P-TOKEN-GIVE or P-TOKEN-PLEASE service to manage the Session tokens. In general, the use of these services by the U-ASE is manifested in the TPPM as a TP-DATA request or a U-ASE indication. There are some cases, however, where the P-TOKEN-GIVE indication which refers to the synchronize-minor token may be seen by the PM. In this case, the procedures of the PM will refer to a P-TOKEN-GIVE (sync-minor) request or indication.

The parameters of the Presentation services are used by the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

TPPMs as specified in tables 11 to 13.

Table 11 - Use of P-TOKEN-PLEASE parameters

P-TOKEN-PLEASE Parameters	Used by TPPM
Tokens (synchronize-minor)	Yes
User data	Yes

Table 12 — Use of P-TOKEN-GIVE (sync-minor) parameters

P-TOKEN-GIVE (sync-minor) Parameters	Used by TPPM
Tokens (synchronize-minor)	Yes
User data	Yes

Table 13 — Use of P-DATA parameters

P-DATA Parameters	Used by TPPM
User data	Yes

8.4.2 Mapping of C-ROLLBACK-RI to Presentation

CCR requires a C-ROLLBACK-RI to be mapped to a P-RESYNCHRONIZE request. When CCR is used with TP, the Tokens parameter of the P-RESYNCHRONIZE request shall be set so that the synchronize-minor token is passed to the superior. TP places no requirements on the setting of the Tokens parameter for the other available tokens.

NOTE — If the C-ROLLBACK-RI is issued by a superior, CCR does not specify a value for the Tokens parameter of the P-RESYNCHRONIZE request.

8.5 Association Management

8.5.1 Introduction

This clause defines the requirements of this part of ISO/IEC 10026 with respect to the management and use of associations.

8.5.2 Association/dialogue compatibility

An association is said to be compatible with a dialogue if it meets the following conditions:

a) the association shall have been established with an AEI fulfilling the requirements expressed in the Application Context Name, Recipient-AP-Title, and any of the following for which a value was specified: Recipient-TPSU-Title, Recipient-API-Identifier, Recipient-AE-Qualifier, and Recipient-AEI-Identifier parameters as specified by the TP-BEGIN-DIALOGUE request;

b) the association shall have selected the kernel and full duplex Session functional units, the kernel Presentation functional unit, and, if the Commit functional unit is selected, the Session functional units as required by CCR;

NOTE — This includes the Session Data Separation functional unit required by CCR version 2.

c) if the Commit functional unit is selected, CCR version 2's abstract syntax name shall be found in the Presentation Context Definition List and the application context;

d) The association shall have been established with a Quality of Service parameter compatible with the Quality-of-Service parameter specified by the TP-BEGIN-DIALOGUE request, if any.

8.5.3 Association/channel compatibility

An association is said to be compatible with a channel if it meets the following conditions:

a) the association shall have been established with an AEI identified by the AE-title as specified in the log record(s) for the transaction(s) to be recovered, and with an application context appropriate for recovery;

b) the association shall have selected the kernel and full duplex Session functional units, the kernel Presentation functional unit, and the Session functional units as required by CCR;

c) CCR version 2's abstract syntax name shall be found in the Presentation Context Definition List and the application context.

8.5.4 Initiating an association establishment

When establishing an association, a TP-INITIALIZE-RI APDU shall be constructed and issued as User Information of the A-ASSOCIATE request. Table 14 lists the fields of this TP APDU.

Table 14 — TP-INITIALIZE-RI/RC APDUs' fields

TP APDU Field	TP-INITIALIZE-RI	TP-INITIALIZE-RC
Protocol-Version	M	M
Contention-Winner-Assignment	M	
Bid-Mandatory	M	
Recovery-Context-Handle	O	O
Diagnostic		O

The fields of the TP-INITIALIZE-RI APDU are set as

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

follows:

a) the Protocol-Version field indicates the possible versions of the TP Protocol that can be supported. The TPPM may support more than one version of the protocol;

NOTE 1 A TPPM may also propose different subsets of the versions it supports; which versions to propose is a local matter.

b) the Contention-Winner-Assignment field identifies whether or not the initiating TPPM will be considered as the contention-winner of this association. This field takes one of the following values:

"true" (the initiator is the contention-winner);
"false" (the initiator is the contention-loser).

If CCR is included in the application context of the association, then the following applies to the setting of the A-ASSOCIATE request parameter "Initial Assignment of Tokens":

1) if the value of the Contention-Winner-Assignment field is "true", the value of the A-ASSOCIATE request parameter "Initial Assignment of Tokens" (see 8.2.1) is "requestor side";

2) if the value of this field is "false", the value of the A-ASSOCIATE request parameter "Initial Assignment of Tokens" is "acceptor side";

c) the Bid-Mandatory field indicates whether the use of the bid mechanism (by the contention-loser) is mandatory or not. This field takes one of the following values:

"true";
"false";

d) the Recovery-Context-Handle field is optionally used to provide a value that is to be supplied when recovery is requested by the remote TPPM;

NOTE 2 An association, once established, may be required to support dialogues with different requirements, and also TP channels. Therefore, the requirements declared when the association is established, must be sufficient for all intended uses of the association. For example, a Recovery-Context-Handle may be specified even though initial use of the association may be for a dialogue without the Commit functional unit or for a TP channel, neither of which have any use for a Recovery-Context-Handle.

8.5.5 Receiving an association establishment indication

Upon receipt of an A-ASSOCIATE indication, a TP-INITIALIZE-RI APDU shall be received as User Information of the A-ASSOCIATE indication. Table 14 lists the fields of this TP APDU.

The fields of the TP-INITIALIZE-RI APDU are used as follows:

a) if the Protocol-Version field contains a version that is supported by the TPPM, the association may be accepted. If not, the association shall be rejected. The TPPM ignores any values that indicate a later version of the protocol than it can support;

b) if the value of the Contention-Winner-Assignment field is acceptable to the TPPM, the association may be accepted. Otherwise, the association shall be rejected;

c) if the value of the Bid-Mandatory field is acceptable to the TPPM, the association may be accepted. Otherwise, the association shall be rejected;

d) if the Recovery-Context-Handle field is present, its value shall be stored in the recovery log (the log-ready record or the log-commit record, whichever applies, see 7.5) for all transactions on the association;

e) if CCR will be used on the association (for a dialogue with coordination level "Commitment" or a channel), and the A-ASSOCIATE indication either

i) does not contain an entry for CCR version 2's abstract syntax name in the Presentation Context Definition List parameter; or

ii) the Session functional units required by CCR version 2 are not selected in the Session Requirements parameter;

then the association shall be rejected. Otherwise, the association may be accepted.

8.5.6 Responding to association establishment

When responding to association establishment, a TP-INITIALIZE-RC APDU shall be constructed and issued as User Information of the A-ASSOCIATE response. Table 14 lists the fields of this TP APDU.

The fields of the TP-INITIALIZE-RC APDU (and the Result parameter of the A-ASSOCIATE response) are set as follows:

a) if the association is accepted,

1) the Result parameter of the A-ASSOCIATE response shall be set to "accepted";

2) the Protocol-Version field of the TP-INITIALIZE-RC APDU shall be set to the version of ISO/IEC 10026 to be used for this association. This version shall be one of the protocol versions

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

proposed on the TP-INITIALIZE-RI APDU;

3) the Recovery-Context-Handle field of the TP-INITIALIZE-RC APDU may (optionally) be set to a value that shall be used when recovery is requested by the remote TPPM;

4) the Diagnostic field of the TP-INITIALIZE-RC APDU shall be omitted;

b) if the association is rejected,

1) the Result parameter of the A-ASSOCIATE response shall be set to

i) "rejected(permanent)", if the Diagnostic field of the TP-INITIALIZE-RC APDU contains any of the settings

a) "tp-protocol-version-incompatibility";
b) "ccr-version-2-not-available";

ii) "rejected(transient)", otherwise;

2) the Protocol-Version field of the TP-INITIALIZE-RC APDU shall be set to the versions of ISO/IEC 10026 that can be supported;

3) the Recovery-Context-Handle field of the TP-INITIALIZE-RC APDU shall be omitted;

4) the Diagnostic field of the TP-INITIALIZE-RC APDU shall be set to all applicable values of the following:

i) "ccr-version-2-not-available", if CCR is required on the association and either or both the CCR Version 2 is not available or the Session functional units required for CCR Version 2 are not selected in the A-ASSOCIATE indication;

ii) "tp-protocol-version-incompatibility", if none of the values of the Protocol-Version field offered in the TP-INITIALIZE-RI APDU can be supported for this association;

iii) "contention-winner-assignment-rejected", if the value of the Contention-Winner-Assignment field of the TP-INITIALIZE-RI APDU is not acceptable for this association;

iv) "bid-mandatory-value-rejected", if the value of the Bid-Mandatory field of the TP-INITIALIZE-RI APDU is not acceptable for this association;

v) "no-reason-given", if none of the above Diagnostic values applies.

8.5.7 Receiving confirmation of association establishment

Upon receipt of an A-ASSOCIATE confirm, a TP-INITIALIZE-RC APDU shall be received as User Information of the A-ASSOCIATE confirm. Table 14 lists the fields of this TP APDU.

The fields of the TP-INITIALIZE-RC APDU are used as follows:

a) if the association is accepted, Protocol-Version defines the protocol version of ISO/IEC 10026 to be used for this association. If the value of the Protocol-Version field is not one of the versions that was proposed on the TP-INITIALIZE-RI APDU, this is an error and the association shall be released;

b) if the association is rejected, the Protocol-Version field contains the protocol versions of ISO/IEC 10026 that can be supported;

c) if the association is accepted and if the Recovery-Context-Handle field is present, its value shall be stored in the recovery log (the log-ready record or the log-commit record, whichever applies, see 7.5) for all transactions on the association.

The A-ASSOCIATE confirm with no embedded TP-INITIALIZE-RC APDU shall only be received when the association establishment was rejected by ACSE.

If the Application Context Name parameter on the A-ASSOCIATE confirm is different from the Application Context Name parameter on the A-ASSOCIATE request, the PM makes a local decision whether to 1) accept the association using this new application context, or 2) release the association.

NOTE — When an association has been rejected, it is a local decision as to what to do next. The TPPM, based on a local decision, may notify the TPSUI that the dialogue is being rejected, it may try again (immediately) to establish the association, it may wait for some period of time and then retry, etc.

8.5.8 Initiating an association release

Release of an association may be initiated according to a local decision at any time the SACF is in the FREE state (see 10.2).

8.5.9 Aborting an association

An A-ABORT request may be issued by a U-ASE at any time. If a U-ASE issues an A-ABORT request, then the TPPM will take the actions appropriate to association abort which may include rolling back the current transaction or initiating recovery.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 15— AF Service Primitives and Associated TP APDUs

AF Service Primitives	TP APDUs
AF-BEGIN-DIALOGUE req/ind AF-BEGIN-DIALOGUE rsp/cnf	TP-BEGIN-DIALOGUE-RI TP-BEGIN-DIALOGUE-RC
AF-BID req/ind AF-BID rsp/cnf	TP-BID-RI TP-BID-RC
AF-END-DIALOGUE req/ind AF-END-DIALOGUE rsp/ind	TP-END-DIALOGUE-RI TP-END-DIALOGUE-RC
AF-U-ERROR req/ind AF-U-ERROR rsp/cnf	TP-U-ERROR-RI TP-U-ERROR-RC
AF-ABORT req/ind	TP-ABORT-RI
AF-ABORT-AND-HEURISTIC-REPORT req/ind	TP-ABORT-RI + TP-HEURISTIC-REPORT-RI
AF-GRANT-CONTROL req/ind	TP-GRANT-CONTROL-RI
AF-REQUEST-CONTROL req/ind	TP-REQUEST-CONTROL-RI
AF-HANDSHAKE req/ind AF-HANDSHAKE rsp/cnf	TP-HANDSHAKE-RI TP-HANDSHAKE-RC
AF-HANDSHAKE-AND-GRANT-CONTROL req/ind AF-HANDSHAKE-AND-GRANT-CONTROL rsp/cnf	TP-HANDSHAKE-AND-GRANT-CONTROL-RI TP-HANDSHAKE-AND-GRANT-CONTROL-RC
AF-DEFER req/ind	TP-DEFER-RI
AF-PREPARE req/ind	TP-PREPARE-RI
AF-HEURISTIC-REPORT req/ind	TP-HEURISTIC-REPORT-RI
AF-TOKEN-GIVE req/ind	TP-TOKEN-GIVE-RI
AF-TOKEN-PLEASE req/ind	TP-TOKEN-PLEASE-RI
AF-RECOVER req/ind	TP-RECOVER-RI

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

9 TP-ASE description

9.1 Introduction

This clause defines the Service Primitives provided by the TP-ASE to the MACF (through the applicable SACF procedures). It further defines the TP APDUs generated to, and received from, the partner TP-ASE. It also defines the TP APDU mappings onto CCR, ACSE, and Presentation services.

Table 15 lists the AF Service primitives together with associated TP APDUs.

9.2 AF Service Definition

The primitives and parameters of the AF services are given in 9.3. The sequence of primitives is given by the appropriate tables in 9.3, reading from left-to-right. This standard (clauses 7 to 11 and, indirectly by reference, the TP Service) defines the constraints on, effects, and collisions of the service primitives.

9.2.1 AF-BEGIN-DIALOGUE

This service is initiated as a direct result of a TP-BEGIN-DIALOGUE request service primitive, or is initiated directly by the CPM to establish a channel.

This service is a confirmed service for channels and an conditionally confirmed service for dialogues.

> NOTE — The only time this service is not confirmed is when the confirmation parameter of the AF-BEGIN-DIALOGUE request is set to "negative", the dialogue is not rejected, and a rollback-initiating service primitive or a TPPM-initiated rollback occurs at the partner prior to any other requests or responses by that partner. In this case, a C-ROLLBACK indication or confirm will serve to confirm the dialogue establishment.

The AF-BEGIN-DIALOGUE response and confirm are used as delimiters for discarding stray APDUs.

When the response and confirm service primitives are referenced in the procedure clauses, the first value listed in the parenthetical argument is the value of the Result parameter; the second is the value of the Mapping parameter.

9.2.2 AF-BID

This confirmed service is used by the SACF in order to gain the rights of the contention-winner temporarily for attempting establishment of the subsequent dialogue or channel.

When the response and confirm service primitives are

referenced in the procedure clauses, the value listed in the parenthetical argument is the value of the Result parameter.

9.2.3 AF-END-DIALOGUE

This service is initiated as a direct result of a TP-END-DIALOGUE request service primitive, or is initiated directly by the CPM to terminate a channel.

This is an optionally confirmed service for dialogues. This is an unconfirmed service for channels.

9.2.4 AF-U-ERROR

This service is initiated as a direct result of a TP-U-ERROR request service primitive.

This is an unconfirmed service in Polarized Control.

In Shared Control, this is an unconfirmed service when it serves as a negative response to a prior AF-HANDSHAKE indication or an AF-END-DIALOGUE indication with the Confirmation parameter set to "true". Otherwise, this is a confirmed service and the request primitive begins the *user error purging period*; in this case, either an AF-U-ERROR confirm, an AF-HANDSHAKE indication, or an AF-END-DIALOGUE indication with the Confirmation parameter set to "true" serves as the confirmation to the AF-U-ERROR request (but see the definition of *user error purging period* in 7.3 for more details when more than one AF-U-ERROR request is unconfirmed).

9.2.5 AF-ABORT

This service is initiated as a direct result of a TP-U-ABORT request service primitive, or is initiated directly by the PM to abnormally terminate a dialogue or channel.

This is an unconfirmed service.

When this service primitive is referenced in the procedure clauses, the first value listed in the parenthetical argument is the value of the Type parameter; the second is the value of the Mapping parameter.

9.2.6 AF-GRANT-CONTROL

This service is initiated as a direct result of a TP-GRANT-CONTROL request service primitive.

This is an unconfirmed service.

9.2.7 AF-REQUEST-CONTROL

This service is initiated as a direct result of a TP-REQUEST-CONTROL request service primitive.

This is an unconfirmed service.

9.2.8 AF-HANDSHAKE

This service is initiated as a direct result of a TP-HANDSHAKE request service primitive.

This is a confirmed service.

9.2.9 AF-HANDSHAKE-AND-GRANT-CONTROL

This service is initiated as a direct result of a TP-HANDSHAKE-AND-GRANT-CONTROL request service primitive.

This service is a confirmed service.

9.2.10 AF-DEFER

This service is initiated as a result of either a TP-DEFERRED-END-DIALOGUE or a TP-DEFERRED-GRANT-CONTROL request service primitive.

This is an unconfirmed service.

When this service primitive is referenced in the procedure clauses, the value listed in the parenthetical argument is the value of the Type parameter.

9.2.11 AF-PREPARE

This service is initiated as a direct result of either a TP-PREPARE request service primitive or a TP-COMMIT request service primitive by a superior.

This is an unconfirmed service.

9.2.12 AF-HEURISTIC-REPORT

This service is initiated by a subordinate as a result of the existence of log-damage in the subtree.

This is an unconfirmed service.

When this service primitive is referenced in the procedure clauses, the value listed in the parenthetical argument is the value of the Mapping parameter.

9.2.13 AF-ABORT-AND-HEURISTIC-REPORT

This service is initiated by a subordinate as a result of the existence of log-damage in the subtree along with a dialogue abort condition.

This is a combination of the AF-ABORT and AF-HEURISTIC-REPORT service and has the combined semantics of the AF-ABORT and AF-HEURISTIC-REPORT services.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

This is an unconfirmed service.

When this service primitive is referenced in the procedure clauses, the value listed in the parenthetical argument is the value of the Mapping parameter.

9.2.14 AF-RECOVER

This service enables TPPMs to invoke recovery after a failure when the recovery-context-handle is provided.

This service is unconfirmed.

When this service primitive is referenced in the procedure clauses, the value listed in the parenthetical argument is the value of the Recovery-State parameter.

9.2.15 AF-TOKEN-GIVE

This service is used to transfer the *token* to the peer PM either during two-way-recovery or while establishing or terminating a dialogue.

This is an unconfirmed service.

9.2.16 AF-TOKEN-PLEASE

This service is used to request the *token* from the peer PM; it will be used by CPMs only.

This is an unconfirmed service.

9.3 AF-Services and TP APDUs: parameters and fields Mappings

9.3.1 AF-BEGIN-DIALOGUE request/indication/response/confirm, TP-BEGIN-DIALOGUE-RI/-RC APDU

Tables 16 and 17 give the parameters and fields mappings for Dialogue Establishment and Channel Establishment, respectively. The parameters and fields are as described in the "Primitives and parameters" subclause for the TP-BEGIN-DIALOGUE service in ISO/IEC 10026-2, with the following exceptions:

a) **Functional-Units:** The meaning and usage of this parameter/field is as described for the TP-BEGIN-DIALOGUE Service in ISO/IEC 10026-2, except that when it is used on a Channel, the parameter specifies only the Recovery Functional Unit (see 13.1.3);

Table 16 — Dialogue Establishment mappings

Service Primitive/TP APDU	TP-	AF-	TP-	AF-	TP-		TP-	AF-	TP-	AF-	TP-
	←——————————BEGIN-DIALOGUE——————————→										
Parameter/Field	**req**	**req**	**-RI**	**ind**	**ind**		**rsp**	**rsp**	**-RC**	**cnf**	**cnf**
Initiating-AP-Title					O						
Initiating-API-Identifier					O						
Initiating-AE-Qualifier					O						
Initiating-AEI-Identifier					O						
Initiating-TPSU-Title	U	(=)	(=)	(=)	(=)						
Recipient-AP-Title	M										
Recipient-API-Identifier	C										
Recipient-AE-Qualifier	C										
Recipient-AEI-Identifier	C										
Recipient-TPSU-Title	U	(=)	(=)	(=)							
Functional-Units	M	(=)	(=)	(=)	(=)			C	(=)	(=)	(=)
Quality-of-Service	U										
Application Context Name	M										
Begin-Transaction	C	(=)	(=)	(=)	(=)						
Confirmation	M	(=)	(=)	(=)	(=)						
Result							M	(=)/M	(=)	(=)	(=)
Diagnostic								C	(=)	(=)/M	(=)
Rollback											M
Correlator		M	(=)	(=)				(=)	(=)	(=)	
Channel-Utilization											
Mapping								M		(=)	
Last-Partner-Identifier		C	(=)	(=)							
User-Data	U	(=)	(=)	(=)	(=)		U	(=)	(=)	(=)	(=)

ECTORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 17 — Channel Establishment mappings

Service Primitive/TP APDU	AF-	TP-	AF-	AF-	TP-	AF-
	——BEGIN-DIALOGUE——					
Parameter/Field	req	-RI	ind	rsp	-RC	cnf
Initiating-AP-Title						
Initiating-API-Identifier						
Initiating-AE-Qualifier						
Initiating-AEI-Identifier						
Initiating-TPSU-Title						
Recipient-AP-Title						
Recipient-API-Identifier						
Recipient-AE-Qualifier						
Recipient-AEI-Identifier						
Recipient-TPSU-Title						
Functional-Units	M	(=)	(=)			
Quality-of-Service						
Application Context Name						
Begin-Transaction						
Confirmation						
Result				M	(=)	(=)
Diagnostic				C	(=)	(=)
Rollback						
Correlator	M	(=)	(=)	(=)	(=)	(=)
Channel-Utilization	M	(=)	(=)			
Mapping				M		(=)
Last-Partner-Identifier	C	(=)	(=)			
User-Data						

b) **Result:** The meaning and usage of this parameter/field is as described for the TP-BEGIN-DIALOGUE Service in ISO/IEC 10026-2, except that the value "rejected(provider)" is permitted on the AF-BEGIN-DIALOGUE response. For a channel, the value "rejected(user)" is not used;

c) **Diagnostic:** The meaning and usage of this parameter/field is as described for the TP-BEGIN-DIALOGUE Service in ISO/IEC 10026-2, except that the following additional values are allowed:

1) "two-way-recovery-not-supported" when "two-way-recovery" was the value of the Channel-Utilization parameter/field. This value only applies to channels;

2) "association-reserved" when the contention-winner has reserved the association for its use;

3) "tppm-recovery-not-available" when the CPM is unable to support recovery, due to a local condition. This value only applies to channels.

The following values do not apply to channels:

1) "recipient-tpsu-title-unknown";
2) "tpsu-not-available(permanent)";
3) "tpsu-not-available(transient)";
4) "recipient-tpsu-title-required";
5) "functional-unit-combination-not-supported";

d) **Correlator:** A correlator unique within the scope of an association.

Due to the use of unconfirmed services and re-use of associations, it may happen that APDUs that are foreign to a particular dialogue are received. Such APDUs are referred to as "stray APDUs".

To resolve such an ambiguity, a correlator is conveyed with the request for, and the acknowledgment/rejection of a dialogue/channel establishment to identify the dialogue/channel.

The value of this parameter is set on the request/response when the service primitive passes through the SACF;

e) **Channel-Utilization:** Identifies the use of the channel. It takes one of the following values:

1) "one-way-recovery";

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 18 — Bid mechanism mappings

Service Primitive/TP APDU	AF-	TP-	AF-		AF-	TP-	AF-
	——————BID——————						
Parameter/Field	req	-RI	ind		rsp	-RC	cnf
CCR-Token-Requested	M	(=)	(=)				
Last-Partner-Identifier	C	(=)	(=)				
Result					M	(=)	(=)

2) "two-way-recovery";

depending on the desired mode of recovery and is present only if the Recovery Functional Unit (see 13.1.3.2) is selected in the value of the Functional-Units parameter of the AF-BEGIN-DIALOGUE request;

f) **Mapping:** Indicates the underlying service to which this AF service is mapped. For AF-BEGIN-DIALOGUE response, it may have one of the following values:

"dataRI "- Mapped to P-DATA request
"rollbackRI" - Mapped to C-ROLLBACK request
"rollbackRC" - Mapped to C-ROLLBACK response

For AF-BEGIN-DIALOGUE confirm, it may have one of the following values:

"dataRI" - Mapped from P-DATA indication
"rollbackRI" - Mapped from C-ROLLBACK indication
"rollbackRC" - Mapped from C-ROLLBACK confirm

g) **Last-Partner-Identifier:** Contains the value of the Correlator of the last TP-BEGIN-DIALOGUE-RI APDU received by the contention-loser. If no dialogue has been established on the association, the bid mechanism has been used for this dialogue, or the requestor is the contention-winner, this parameter/field does not exist.

9.3.2 AF-BID request/indication/response/confirm, TP-BID-RI/-RC APDU

Table 18 gives the parameters and fields mappings for the bid mechanism. The parameters and fields are as follows:

a) **CCR-Token-Requested:** Indicates whether the *token* required by CCR is requested. It takes the following values:

1) "true", when the *token* is requested;
2) "false", when the *token* is not requested;

b) **Last-Partner-Identifier:** Contains the value of the Correlator of the last TP-BEGIN-DIALOGUE-RI APDU received by the contention-loser. If no dialogue has been established on the association, this parameter/field does not exist;

c) **Result:** Indicates whether the contention-loser has been temporarily granted the rights of the contention-winner. It takes the following values:

1) "accepted", when the rights are granted;
2) "rejected", when the rights are not granted.

9.3.3 AF-END-DIALOGUE request/indication/response/confirm, TP-END-DIALOGUE-RI/-RC APDU

Table 19 gives the parameter and field mappings for Dialogue Termination. The parameter and field are as described in the "Primitives and parameters" subclause for the TP-END-DIALOGUE service in ISO/IEC 10026-2.

Table 20 gives the parameter and field mappings for Channel Termination, The parameter and field is as follows:

a) **Confirmation:** This must be set to "false" (see 11.4.6, "Terminating a channel").

Table 19 — Dialogue termination mappings

Service Primitive/ TP APDU	TP-	AF-	TP-	AF-	TP-
		——END-DIALOGUE——			
Parameter/Field	req	req	RI	ind	ind
Confirmation	M	(=)	(=)	(=)	(=)

Table 20 — Channel termination mappings

Service Primitive/ TP APDU	AF-	TP-	AF-
	END-DIALOGUE		
Parameter/Field	req	RI	ind
Confirmation	M	(=)	(=)

NOTE — The confirmation parameter is always "false" for Channel Termination (see 11.4.6, "Terminating a channel.")

ICSNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The AF-END-DIALOGUE response and confirm primitives have no parameters; the TP-END-DIALOGUE-RC APDU has no fields.

9.3.4 AF-U-ERROR request/indication/ response/confirm, TP-U-ERROR-RI/-RC APDU

These services have no parameters and these TP APDUs have no fields.

9.3.5 AF-ABORT request/indication, TP-ABORT-RI APDU

Tables 21 and 22 give the parameters and fields mappings for User Abort and Provider Abort, respectively. The parameters and fields are as described in the "Primitives and parameters" subclause for the TP-U-ABORT and TP-P-ABORT services in ISO/IEC 10026-2, with the following exceptions:

a) **Type:** Indicates the type of abort (user or provider). It takes one of the following values:

1) "user";
2) "provider";

b) **Mapping:** Indicates the underlying service to which this AF service is mapped. For AF-ABORT request, it takes one of the following values:

"abortRI" - Mapped to A-ABORT request
"dataRI" - Mapped to P-DATA request
"commitRI" - Mapped to C-COMMIT request
"commitRC" - Mapped to C-COMMIT response
"rollbackRI" - Mapped to C-ROLLBACK request
"rollbackRC" - Mapped to C-ROLLBACK response

For AF-ABORT indication, it takes one of the following values:

"abortRI" - Mapped from A-ABORT indication
"dataRI" - Mapped from P-DATA indication
"commitRI" - Mapped from C-COMMIT indication
"commitRC" - Mapped from C-COMMIT confirm
"rollbackRI" - Mapped from C-ROLLBACK indication
"rollbackRC" - Mapped from C-ROLLBACK confirm

c) **Diagnostic:** Does not take the values "end-dialogue-collision" or "begin-transaction-end-dialogue-collision".

NOTE — These values are generated locally by the MACF.

Table 21 — User Abort mappings

Service Primitive/ TP APDU	TP-U	AF-	TP-	AF-	TP-U
		ABORT			
Parameter/Field	req	req	RI	ind	ind
Type		M	(=)	(=)	
Mapping		M		(=)	
Diagnostic					
Rollback					M
User-Data	U	(=)	(=)	(=)	(=)

Table 22 — Provider Abort mappings

Service Primitive/ TP APDU	AF-	TP-	AF-	TP-P
		ABORT		
Parameter/Field	req	RI	ind	ind
Type	M	(=)	(=)	
Mapping	M		(=)	
Diagnostic	M	(=)	(=)	(=)/M
Rollback				M
User-Data				

9.3.6 AF-GRANT-CONTROL request/ indication, TP-GRANT-CONTROL-RI APDU

These services have no parameters and this TP APDU has no fields.

9.3.7 AF-REQUEST-CONTROL request/ indication, TP-REQUEST-CONTROL-RI APDU

These services have no parameters and this TP APDU has no fields.

9.3.8 AF-HANDSHAKE request/indication/ response/confirm, TP-HANDSHAKE-RI/-RC APDU

Table 23 gives the parameter and field mappings for Handshake. The parameter and field are as described in the "Primitives and parameters" subclause for the TP-HANDSHAKE service in ISO/IEC 10026-2.

The AF-HANDSHAKE response and confirm primitives have no parameters; the TP-HANDSHAKE-RC APDU has no fields.

Table 23 — Handshake mappings

Service Primitive/ TP APDU	TP-	AF-	TP-	AF-	TP-
		HANDSHAKE			
Parameter/Field	req	req	RI	ind	ind
Confirmation-Urgency	C	(=)	(=)		

ECMORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

9.3.9 AF-HANDSHAKE-AND-GRANT-CONTROL request/indication/response confirm, TP-HANDSHAKE-AND-GRANT-CONTROL-RI/-RC APDU

Table 24 gives the parameter and field mappings for Handshake and Grant Control. The parameter and field are as described in the "Primitives and parameters" subclause for the TP-HANDSHAKE-AND-GRANT-CONTROL service in ISO/IEC 10026-2.

The AF-HANDSHAKE-AND-GRANT-CONTROL response and confirm primitives have no parameters; the TP-HANDSHAKE-AND-GRANT-CONTROL-RC APDU has no fields.

Table 24 — Handshake and Grant Control mappings

Service Primitive/ TP APDU	TP-	AF-	TP-	AF-	TP-
	HANDSHAKE-AND-GRANT-CONTROL				
Parameter/Field	**req**	**req**	**RI**	**ind**	**ind**
Confirmation-Urgency	M	(=)	(=)		

9.3.10 AF-DEFER request/indication, TP-DEFER-RI APDU

Table 25 gives the parameter and field mappings for Deferred End Dialogue and Deferred Grant Control. The parameter and field are as described following:

a) **Type:** Indicates the type of Deferred service (End Dialogue or Grant Control). It takes one of the following values:

1) "end-dialogue";
2) "grant-control".

9.3.11 AF-PREPARE request/indication, TP-PREPARE-RI APDU

Table 26 gives the parameter and field mappings for TP/AF-PREPARE. The parameters and fields are as described in the "Primitives and parameters" subclauses for TP-PREPARE request and TP-PREPARE indication in ISO/IEC 10026-2.

Table 26 — TP/AF-PREPARE mappings

Service Primitive/ TP APDU	TP-	AF-	TP-	AF-	TP-
	———PREPARE———				
Parameter/Field	**req**	**req**	**RI**	**ind**	**ind**
Data-Permitted	C	(=)	(=)	(=)	(=)

9.3.12 AF-HEURISTIC-REPORT request/indication, TP-HEURISTIC-REPORT-RI APDU

Table 27 gives the parameters and field mappings for Heuristic Reporting. The parameters and field are as described in the "Primitives and parameters" subclause for the TP-HEURISTIC-REPORT service in ISO/IEC 10026-2, and in the "C-RECOVER parameters" subclause in ISO/IEC 9804, with the following exception:

a) **Mapping:** Indicates the underlying service to which this AF service is mapped. For AF-HEURISTIC-REPORT request, it takes one of the following values:

"commitRC" - Mapped to C-COMMIT response
"recoverDoneRC" - Mapped to C-RECOVER (done) response
"rollbackRI" - Mapped to C-ROLLBACK request
"rollbackRC" - Mapped to C-ROLLBACK response

Table 27 — Heuristic Reporting mappings

Service Primitive/TP APDU	AF-	TP-	AF-	TP-
	HEURISTIC-REPORT			
Parameter/Field	**req**	**RI**	**ind**	**ind**
Mapping	M		(=)	
Heuristic-Report	M	(=)	(=)	(=)
Atomic Action Identifier	C[1)]			
Branch Identifier	C[1)]			

1) These parameters exist if the Mapping parameter is recoverDoneRC

Table 25 — Deferred End Dialogue and Deferred Grant Control mappings

Service Primitive/TP APDU	TP- D-E-D or D-G-C[1)]	AF-	TP-	AF-	TP- D-E-D or D-G-C[1)]
		——DEFER——			
Parameter/Field	**req**	**req**	**RI**	**ind**	**ind**
Type		M	(=)	(=)	

1) TP-DEFERRED-END-DIALOGUE or TP-DEFERRED-GRANT-CONTROL

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

For AF-HEURISTIC-REPORT indication, it takes one of the following values:

"commitRC" - Mapped from C-COMMIT confirm
"recoverDoneRC" - Mapped from C-RECOVER (done) confirm
"rollbackRI" - Mapped from C-ROLLBACK indication
"rollbackRC" - Mapped from C-ROLLBACK confirm

9.3.13 AF-ABORT-AND-HEURISTIC-REPORT request/indication

Table 28 gives the parameters mappings for the combined Auxiliary Facility Service AF-ABORT-AND-HEURISTIC-REPORT. The parameters are as described in the "Primitives and parameters" subclause for the TP-HEURISTIC-REPORT service and the TP-U-ABORT service in ISO/IEC 10026-2, with the following exceptions:

a) **Mapping:** Indicates the underlying service to which this AF service is mapped. For AF-ABORT-AND-HEURISTIC-REPORT request, it takes one of the following values:

"commitRC" - Mapped to C-COMMIT response
"rollbackRI" - Mapped to C-ROLLBACK request
"rollbackRC" - Mapped to C-ROLLBACK response

For AF-ABORT-AND-HEURISTIC-REPORT indication, it takes one of the following values:

"commitRC" - Mapped from C-COMMIT confirm
"rollbackRI" - Mapped from C-ROLLBACK indication
"rollbackRC" - Mapped from C-ROLLBACK confirm

Table 28 — AF-ABORT-AND-HEURISTIC-REPORT request/indication

Service Primitive/TP APDU	AF-ABORT-AND-HEURISTIC-REPORT	
Parameter/Field	**req**	**ind**
Mapping	M	(=)
Heuristic-Report	M	(=)
User-Data	C	(=)

The User-Data parameter of the AF-ABORT-AND-HEURISTIC-REPORT service is present if the TP-U-ABORT request has a User-Data parameter.

9.3.14 AF-RECOVER request/indication, TP-RECOVER-RI APDU

Table 29 gives the parameters and fields mappings for TP/AF-RECOVER. The parameters and fields are as described in the "C-RECOVER parameters" subclause in ISO/IEC 9804, with the following exceptions:

a) **Recovery-Context-Handle:** Takes the value of the field with the same name received on the TP-INITIALIZE-RI/RC APDU from the partner AEI.

Table 29 — TP/AF-RECOVER mappings

Service Primitive/TP APDU	AF-RECOVER	TP-RECOVER	AF-RECOVER
Parameter/Field	**req**	**RI**	**ind**
Recovery-State	M		(=)
Recovery-Context-Handle	M	(=)	(=)
Atomic Action Identifier	M		
Branch Identifier	M		

If a Recovery-Context-Handle does not exist for the remote partner, then this AF-Service and this TP APDU shall not be used.

9.3.15 AF-TOKEN-GIVE request/indication, TP-TOKEN-GIVE-RI APDU

Table 30 gives the parameters and fields mappings for the AF-TOKEN-GIVE service. The parameters and fields are as described following:

a) **Reason:** Indicates the reason for which the *token* is transferred. It takes one of the following values:

1) regular - the *token* is transferred under the following conditions:

i) when it arrives at the contention-loser outside of a dialogue (that is, in the SACF FREE, STRAY, or BIDDING states);

ii) when it arrives at the contention-winner after an AF-BID (accepted) response request is issued;

iii) after the issuance of a *rollback response* or the receipt of a *rollback confirm* outside of a dialogue (that is, in the SACF CLEANUP ROLLBACK INDICATION EXPECTED or CLEANUP ROLLBACK CONFIRM EXPECTED states) by the contention-loser;

2) keep - the *token* is transferred under the following conditions:

i) when an AF-BEGIN-DIALOGUE indication is received by the contention-winner and the bid

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

mechanism was not used;

ii) when an AF-TOKEN-GIVE (regular) indication is received by the contention-winner within a dialogue (that is, in the SACF BUSY state);

3) two-way-recovery - the *token* is transferred on a two-way-recovery channel after the issuance of a C-RECOVER request or AF-RECOVER request;

Table 30 — TP/AF-TOKEN-GIVE mappings

Service Primitive/TP APDU	AF-	TP-	AF-
	—TOKEN-GIVE—		
Parameter/Field	req	RI	ind
Reason	M	(=)	(=)
Correlator	C	(=)	(=)

b) **Correlator**: Contains the value of the Correlator of the last TP-BEGIN-DIALOGUE-RI APDU received by the contention-winner. The Correlator of the AF-TOKEN-GIVE request is present if the value of the Reason parameter is set to "keep".

9.3.16 AF-TOKEN-PLEASE request/ indication, TP-TOKEN-PLEASE-RI APDU

These services have no parameters and this TP APDU has no fields.

9.4 Procedures

The following subclauses specify the actions taken upon receipt of the specified AF-Service primitive, TP APDU, or ACSE or CCR Service primitive containing a TP APDU.

NOTES

1 ACSE and CCR indication and confirm service primitives that do not contain an embedded TP APDU are given to the MACF through the appropriate SACF procedures.

2 TP APDUs not contained in ACSE or CCR Service primitives are contained in Presentation Service primitives.

9.4.1 AF-BEGIN-DIALOGUE request

— Send a TP-BEGIN-DIALOGUE-RI APDU.

9.4.2 TP-BEGIN-DIALOGUE-RI TP APDU

— Issue an AF-BEGIN-DIALOGUE indication.

9.4.3 AF-BEGIN-DIALOGUE response

— Send a TP-BEGIN-DIALOGUE-RC APDU as user data in the service specified by the Mapping parameter.

9.4.4 TP-BEGIN-DIALOGUE-RC TP APDU

— Issue an AF-BEGIN-DIALOGUE confirm with the value of the Mapping parameter set to "dataRI".

9.4.5 AF-BID request

— Send a TP-BID-RI APDU.

9.4.6 TP-BID-RI TP APDU

— Issue an AF-BID indication.

9.4.7 AF-BID response

— Send a TP-BID-RC APDU.

9.4.8 TP-BID-RC TP APDU

— Issue an AF-BID confirm.

9.4.9 AF-END-DIALOGUE request

— Send a TP-END-DIALOGUE-RI APDU.

9.4.10 TP-END-DIALOGUE-RI TP APDU

— Issue an AF-END-DIALOGUE indication.

9.4.11 AF-END-DIALOGUE response

— Send a TP-END-DIALOGUE-RC APDU.

9.4.12 TP-END-DIALOGUE-RC TP APDU

— Issue an AF-END-DIALOGUE confirm.

9.4.13 AF-U-ERROR request

— Send a TP-U-ERROR-RI APDU.

9.4.14 TP-U-ERROR-RI TP APDU

— Issue an AF-U-ERROR indication.

9.4.15 AF-U-ERROR response

— Send a TP-U-ERROR-RC APDU.

9.4.16 TP-U-ERROR-RC TP APDU

— Issue an AF-U-ERROR confirm.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

9.4.17 AF-ABORT request

— Send a TP-ABORT-RI TP APDU as user data in the service specified by the Mapping parameter.

9.4.18 TP-ABORT-RI TP APDU

— Issue an AF-ABORT indication with the value of the Mapping parameter set to "dataRI".

9.4.19 AF-GRANT-CONTROL request

— Send a TP-GRANT-CONTROL-RI APDU.

9.4.20 TP-GRANT-CONTROL-RI TP APDU

— Issue an AF-GRANT-CONTROL indication.

9.4.21 AF-REQUEST-CONTROL request

— Send a TP-REQUEST-CONTROL-RI APDU.

9.4.22 TP-REQUEST-CONTROL-RI TP APDU

— Issue an AF-REQUEST-CONTROL indication.

9.4.23 AF-HANDSHAKE request

— Send a TP-HANDSHAKE-RI APDU.

9.4.24 TP-HANDSHAKE-RI TP APDU

— Issue an AF-HANDSHAKE indication.

9.4.25 AF-HANDSHAKE response

— Send a TP-HANDSHAKE-RC APDU.

9.4.26 TP-HANDSHAKE-RC TP APDU

— Issue an AF-HANDSHAKE confirm.

9.4.27 AF-HANDSHAKE-AND-GRANT-CONTROL request

— Send a TP-HANDSHAKE-AND-GRANT-CONTROL-RI APDU.

9.4.28 TP-HANDSHAKE-AND-GRANT-CONTROL-RI TP APDU

— Issue an AF-HANDSHAKE-AND-GRANT-CONTROL indication.

9.4.29 AF-HANDSHAKE-AND-GRANT-CONTROL response

— Send a TP-HANDSHAKE-AND-GRANT-CONTROL-RC APDU.

9.4.30 TP-HANDSHAKE-AND-GRANT-CONTROL-RC TP APDU

— Issue an AF-HANDSHAKE-AND-GRANT-CONTROL confirm.

9.4.31 AF-DEFER request

— Send a TP-DEFER-RI APDU.

9.4.32 TP-DEFER-RI TP APDU

— Issue an AF-DEFER indication.

9.4.33 AF-PREPARE request

— Send a TP-PREPARE-RI APDU as User Data of a C-PREPARE request.

9.4.34 C-PREPARE indication

The TP-PREPARE-RI APDU is received as User Data of the C-PREPARE indication.

— Issue an AF-PREPARE indication.

9.4.35 AF-HEURISTIC-REPORT request

— Send a TP-HEURISTIC-REPORT-RI APDU as user data in the service specified by the Mapping parameter.

9.4.36 AF-ABORT-AND-HEURISTIC-REPORT request

— Send a TP-HEURISTIC-REPORT-RI APDU followed by a TP-ABORT-RI APDU with the value of the Type field set to "user" as user data in the service specified by the Mapping parameter.

9.4.37 AF-RECOVER request

— Send a TP-RECOVER-RI APDU as user data of a C-RECOVER request.

9.4.38 C-RECOVER indication

The TP-RECOVER-RI APDU is received as User Data of the C-RECOVER indication.

— Issue an AF-RECOVER indication.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

9.4.39 A-ABORT indication

The TP-ABORT-RI APDU is received as User Information of the A-ABORT indication.

— Issue an AF-ABORT indication with the Mapping parameter set to "abortRI".

9.4.40 C-ROLLBACK indication

a) If both a TP-ABORT-RI APDU and a TP-HEURISTIC-REPORT-RI APDU are received as User Data,

— issue an AF-ABORT-AND-HEURISTIC-REPORT indication with the value of the

1) Heuristic-Report parameter set to that of the TP-HEURISTIC-REPORT-RI TP APDU;

2) Mapping parameter set to "rollbackRI";

b) if only a TP-ABORT-RI APDU is received as User Data,

— issue an AF-ABORT indication with the value of the Mapping parameter set to "rollbackRI";

c) if only a TP-HEURISTIC-REPORT-RI APDU is received as User Data,

— issue an AF-HEURISTIC-REPORT indication with the value of the Mapping parameter set to "rollbackRI";

d) if a TP-BEGIN-DIALOGUE-RC APDU is received as User Data,

— issue an AF-BEGIN-DIALOGUE confirm with the value of the Mapping parameter set to "rollbackRI".

9.4.41 C-ROLLBACK confirm

a) If both a TP-ABORT-RI APDU and a TP-HEURISTIC-REPORT-RI APDU are received as User Data,

— issue an AF-ABORT-AND-HEURISTIC-REPORT indication with the value of the

1) Heuristic-Report parameter set to that of the TP-HEURISTIC-REPORT-RI TP APDU;

2) Mapping parameter set to "rollbackRC";

b) if only a TP-ABORT-RI APDU is received as User Data,

— issue an AF-ABORT indication with the value of the Mapping parameter set to "rollbackRC";

c) if only a TP-HEURISTIC-REPORT-RI APDU is received as User Data,

— issue an AF-HEURISTIC-REPORT indication with the value of the Mapping parameter set to "rollbackRC;

d) if a TP-BEGIN-DIALOGUE-RC APDU is received as User Data,

— issue an AF-BEGIN-DIALOGUE confirm with the value of the Mapping parameter set to "rollbackRC".

9.4.42 C-COMMIT indication

The TP-ABORT-RI APDU is received as User Data of the C-COMMIT indication.

— Issue an AF-ABORT indication with the value of the Mapping parameter set to "commitRI".

9.4.43 C-COMMIT confirm

a) If both a TP-ABORT-RI APDU and a TP-HEURISTIC-REPORT-RI APDU are received as User Data,

— issue an AF-ABORT-AND-HEURISTIC-REPORT indication with the value of the

1) Heuristic-Report parameter set to that of the TP-HEURISTIC-REPORT-RI TP APDU;

2) Mapping parameter set to "commitRC";

b) if only a TP-ABORT-RI APDU is received as User Data,

— issue an AF-ABORT indication with the value of the Mapping parameter set to "commitRC";

c) if only a TP-HEURISTIC-REPORT-RI APDU is received as User Data,

— issue an AF-HEURISTIC-REPORT indication with the value of the Mapping parameter set to "commitRC.

9.4.44 C-RECOVER confirm

The TP-HEURISTIC-REPORT-RI APDU is received as User Data of the C-RECOVER confirm.

— Issue an AF-HEURISTIC-REPORT indication with the value of the Mapping parameter set to "recoverDoneRC".

9.4.45 P-TOKEN-GIVE (sync-minor) indication

The TP-TOKEN-GIVE-RI APDU is received as User data of

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

the P-TOKEN-GIVE (sync-minor) indication.

— Issue an AF-TOKEN-GIVE indication.

9.4.46 AF-TOKEN-GIVE request

— Send a TP-TOKEN-GIVE-RI APDU as User data of a P-TOKEN-GIVE(sync-minor) request.

9.4.47 P-TOKEN-PLEASE (sync-minor) indication

The TP-TOKEN-PLEASE-RI APDU is received as User data of the P-TOKEN-PLEASE (sync-minor) indication.

— Issue an AF-TOKEN-PLEASE indication.

9.4.48 AF-TOKEN-PLEASE request

— Send a TP-TOKEN-PLEASE-RI APDU as User data of a P-TOKEN-PLEASE (sync-minor) request.

9.5 Mapping

All TP APDUs are transferred as specified in clauses 9, 10, and 11, either as non-embedded TP APDUs or embedded in user data of other ASE service primitives.

The TP APDUs are carried by CCR, ACSE, or Presentation services as shown in table 31, except for the case of concatenation (see 10.7).

Table 31 — CCR, ACSE and Presentation services carrying TP APDUs

TP APDUs	CCR, ACSE and Presentation Services [1]
TP-BEGIN-DIALOGUE-RI	P-DATA req
TP-BEGIN-DIALOGUE-RC	P-DATA req or C-ROLLBACK req or C-ROLLBACK rsp
TP-BID-RI/RC	P-DATA req
TP-END-DIALOGUE-RI/RC	P-DATA req
TP-ABORT-RI	P-DATA req or C-ROLLBACK req or C-ROLLBACK rsp or C-COMMIT req or C-COMMIT rsp or A-ABORT req
TP-U-ERROR-RI/RC	P-DATA req
TP-GRANT-CONTROL-RI	P-DATA req
TP-REQUEST-CONTROL-RI	P-DATA req
TP-HANDSHAKE-RI/RC	P-DATA req
TP-HANDSHAKE-AND-GRANT-CONTROL-RI/RC	P-DATA req
TP-PREPARE-RI	C-PREPARE req
TP-DEFER-RI	P-DATA req
TP-HEURISTIC-REPORT-RI	C-ROLLBACK req, or C-ROLLBACK rsp, or C-COMMIT rsp, or C-RECOVER (done) rsp
TP-TOKEN-GIVE-RI	P-TOKEN-GIVE (sync-minor) req
TP-TOKEN-PLEASE-RI	P-TOKEN-PLEASE (sync-minor) req
TP-RECOVER-RI	C-RECOVER req
TP-INITIALIZE-RI	A-ASSOCIATE req
TP-INITIALIZE-RC	A-ASSOCIATE rsp

1) TP APDUs shown as mapping to the P-DATA Service may have a different mapping depending on the combined set of concatenation rules used in the SAO.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

10 SACF description

10.1 Introduction

The SACF procedures (described in 10.2 through 10.6) handle the queuing, bidding, and token control related to dialogue/channel establishment as well as the management of the association while not in use by a dialogue or channel. When unqualified, the term SACF refers to these procedures.

Dashed lists are used exclusively to present the actions.

The rules associated with the concatenator part of the SACF are described in clause 10.7.

The router part of the SACF is described in clause 10.8.

10.2 SACF states

The SACF has one of the following states at all times. These states are visible only to the SACF. The SACF state is changed either upon receipt of a relevant AF, CCR, or SAF service primitive or, spontaneously, upon certain SACF internal events. When the SAO is newly created, the SACF is in the FREE state. The SACF states are:

a) FREE

This state is used when the SAO is available to be *attached* to a PM for a dialogue or channel. It is therefore in the pool of free associations. When the association is newly established, the SACF is in the FREE state.

b) STRAY

This state is used to filter service primitives that may be received between the receipt of an AF-BEGIN-DIALOGUE request and the receipt of the corresponding AF-BEGIN-DIALOGUE confirm in the case where there is no bidding.

c) BIDDING (contention-loser only)

This state is used to filter service primitives which may be received between the receipt of an AF-BEGIN-DIALOGUE request and the receipt of an AF-BID confirm and the SACF bids. In the case of a dialogue, this state only applies to a subordinate dialogue.

d) BID CONFIRM RECEIVED (contention-loser only)

This state is used to detect protocol errors which become manifest by receiving an indication or confirm primitive between the receipt of a AF-BID confirm and an AF-BEGIN-DIALOGUE confirm. In the case of a dialogue, this state only applies to a subordinate dialogue.

e) BID INDICATION RECEIVED (contention-winner only)

This state is used to detect protocol errors which become manifest by receiving an indication or confirm primitive between the receipt of a AF-BID indication and an AF-BEGIN-DIALOGUE indication. In the case of a dialogue, this state only applies to the superior dialogue.

h) BUSY

This state is used to allow pass through of service primitives that may be received between the receipt of:

1) an AF-BEGIN-DIALOGUE indication by a contention-loser, or by a contention-winner without having issued an AF-BEGIN-DIALOGUE request; or

2) an AF-BEGIN-DIALOGUE confirm with a valid Correlator; or

3) a C-ROLLBACK indication or confirm after an AF-BEGIN-DIALOGUE request with the Confirmation parameter set to "negative" has been issued and an AF-BEGIN-DIALOGUE confirm has not been received

and the issuance of a SAF-DETACH-ASSOCIATION request.

i) CLEANUP ROLLBACK INDICATION EXPECTED (superior only)

This state is used when a *rollback indication* is expected outside of a dialogue and must be acknowledged with a *rollback response*.

j) CLEANUP BEGIN INDICATION EXPECTED (subordinate only)

This state is used when a C-BEGIN indication is expected outside of a dialogue and therefore the atomic action branch must be rolled back.

k) CLEANUP ROLLBACK CONFIRM EXPECTED (subordinate only)

This state is used when a *rollback confirm* is expected outside of a dialogue. If, however, a *rollback indication* is received, i.e., a rollback collision occurs, it is the responsibility of the SACF to resend the semantics of the previous *rollback request* in the *rollback response*, using the appropriate AF-service, if necessary.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

10.3 Service definitions for SAF-DETACH-ASSOCIATION and SAF-ASSOCIATION-LOST

10.3.1 SAF-DETACH-ASSOCIATION request

The MACF uses this service to notify the SACF to *detach* from the association.

When the request service primitive is referenced in the procedure clauses, the value listed in the parenthetical argument is the Status parameter.

This is an unconfirmed service.

Table 32 gives the parameter of this primitive.

Table 32 — SAF-DETACH-ASSOCIATION parameter

SAF-DETACH-ASSOCIATION	
Parameter	**req**
Status	M

a) **Status:** Indicates the status of the association that is being *detached*. The values are:

1) "free" - The association is available for assignment to another dialogue;

2) "rollback-indication-expected" - The association is in a state where the partner TPPM is expected to issue a *rollback request* and the corresponding *rollback indication* is expected. Once the *rollback indication* is received, the association will be available for assignment to another dialogue;

3) "rollback-confirm-expected" - A *rollback confirm* is expected. Once the *rollback confirm* is received the association is available for assignment. If a *rollback indication* is received, the AF-service corresponding to the previous *rollback request* is reissued as a *rollback response*. If the previous *rollback request* was not an AF-service, a C-ROLLBACK response is issued. An AF-ABORT (user, dataRI) indication may also be received since this may be an alternative to the C-BEGIN indication. When the AF-ABORT (user/dataRI) indication is received, the association is available;

4) "begin-indication-expected" - A C-BEGIN is expected to be received on this association. Once the C-BEGIN indication is received, a C-ROLLBACK request is issued to roll back the branch. Only when the *rollback confirm* is received is the association available;

5) "begin-fear" - a C-BEGIN indication may be received in the future due to one of the following conditions:

i) the Unchained Transactions functional unit is selected and the dialogue was terminated by an AF-ABORT (user, dataRI) request, an AF-END-DIALOGUE request with the Confirmation parameter set to "false", or an AF-BEGIN-DIALOGUE (rejected(user/provider)) response; or

ii) the *dialogue is chaining, rollback reporting has completed,* and an AF-ABORT (user, dataRI) request was issued while in the DECIDED (rollback) state.

10.3.2 SAF-ASSOCIATION-LOST indication

The SACF uses this service to notify the MACF that it is *detached* from the association.

10.4 Procedure for SAF-DETACH-ASSOCIATION request

If the value of the status parameter is

a) "free" or "begin-fear" and

1) a queue exists,

— continue;

2) no queue exists,

— enter the FREE state;

— issue an AF-TOKEN-GIVE (regular) request if the SACF is a contention-loser and the *token* is owned;

b) "rollback-indication-expected" and there is no queue,

— enter the CLEANUP ROLLBACK INDICATION EXPECTED state;

c) "rollback-confirm-expected" and there is no queue,

— enter the CLEANUP ROLLBACK CONFIRM EXPECTED state;

d) "begin-indication-expected",

— enter the CLEANUP BEGIN INDICATION EXPECTED state;

e) "rollback-indication-expected" or "rollback-confirm-expected" and there is a queue,

— *discard the queue*;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— issue an AF-TOKEN-GIVE (regular) request, if the SACF is the contention-loser and the *token* is owned;

— enter the FREE state.

Always

— *detach* the MACF from the association.

10.5 Procedures for TP-ASE, CCR, ACSE, and Presentation Service Primitives

Service primitives received from the MACF are assumed valid, i.e., issued in accordance with the procedures specified in clause 11.

10.5.1 AF-BEGIN-DIALOGUE request

This service primitive is received while the SACF is in the FREE state, and the association is compatible with the dialogue or channel.

If the SACF is a contention-loser and one or more of the following is true:

a) the bid-mandatory field of the TP-INITIALIZE-RI APDU was "true" for this association;

b) the last time this SAO was detached from the MACF this was done by a SAF-DETACH-ASSOCIATION request which had the Type parameter set to "begin-fear"; or

c) according to a local decision, the SACF will bid,

then

a) formulate the parameters of the AF-BID request with the value of the

1) Last-Partner-Identifier parameter

i) omitted if no AF-BEGIN-DIALOGUE indication has been received on this association since it was established;

ii) set to the Correlator of the most recently received AF-BEGIN-DIALOGUE indication otherwise;

2) CCR-Token-Requested parameter set to either

i) "true" if the Commit or Recovery Functional Unit is selected; or
ii) "false" otherwise;

b) take the following actions

— issue an AF-BID request;

— *establish a queue*;

— enter the BIDDING state.

If no AF-BID request was issued,

— enter the STRAY state.

If the SACF is a contention-winner and the last time this SAO was detached from the MACF this was done by a SAF-DETACH-ASSOCIATION request which had the Type parameter set to "begin-fear",

— *establish a queue*.

Always take the following actions:

a) formulate the parameters of the AF-BEGIN-DIALOGUE request with the value of the

1) Last-Partner-Identifier parameter

i) omitted if any of the following are true:

a) the SACF is a contention-winner;

b) an AF-BID request was issued; or

c) no AF-BEGIN-DIALOGUE indication has been received on this association;

ii) set to the Correlator of the most recently received AF-BEGIN-DIALOGUE indication otherwise;

2) Correlator parameter set to a value which is unique within the scope of the association;

b) if a queue exists,

— queue the AF-BEGIN-DIALOGUE request;

c) if a queue does not exist,

— pass through the AF-BEGIN-DIALOGUE request.

10.5.2 AF-BEGIN-DIALOGUE indication

If the SACF is the contention-winner and the Bid-Mandatory field of the TP-INITIALIZE-RI APDU was set to "false", and either

a) the SACF is in the FREE state and the *last partner identifier is not valid*; or

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

b) the SACF is in the STRAY state

— continue.

If the SACF is the contention-loser and in the STRAY or BIDDING states,

— issue a SAF-ASSOCIATION-LOST indication if no SAF-DETACH-ASSOCIATION request was received;

— *discard the queue* (if any).

NOTE 1 It is a local matter whether the TPPM retries establishing a dialogue or channel on another association; however, in the case where the SAF-ASSOCIATION-LOST indication is not issued because a SAF-DETACH-ASSOCIATION request was already received, retrying is recommended.

If all of the following conditions are true:

a) the SACF is the contention-winner in the FREE state and the *last partner identifier is valid*;

b) the Commit or Recovery functional unit is selected;

c) the Bid-Mandatory field of the TP-INITIALIZE-RI APDU was set to "false";

d) the association has not been reserved for other use; and

e) the *token* is owned by the SACF,

then

— issue an AF-TOKEN-GIVE (keep) request.

NOTE 2 If the Commit functional unit is not supported, this service primitive need not be generated, as the dialogue will be rejected by the MACF.

If the SACF is one of the following:

a) the contention-winner and either in the FREE state, the *last partner identifier is valid*, the Bid-Mandatory field of the TP-INITIALIZE-RI APDU was set to "false", and the association has not been reserved for other use, or in the BID INDICATION RECEIVED state and an AF-BID response has been sent; or

b) the contention-loser and in the FREE, STRAY, or BIDDING state;

then

— create a new MACF, if this is a dialogue;

— *attach* to the CPM, if this is a channel;

— enter the BUSY state;

— pass the service primitive through.

If the SACF is the contention-winner and in the FREE state and the Bid-Mandatory field of the TP-INITIALIZE-RI APDU was set to "false", the *last partner identifier is valid*, and the association has been reserved for other use,

— issue an AF-BEGIN-DIALOGUE (rejected(provider), dataRI) response with the Diagnostic parameter set to "association-reserved".

10.5.3 AF-BEGIN-DIALOGUE response

The Correlator parameter of the AF-BEGIN-DIALOGUE response is set to the value of the parameter on the previous AF-BEGIN-DIALOGUE indication.

— Pass the service primitive through.

10.5.4 AF-BEGIN-DIALOGUE confirm

If the SACF is in the FREE or BIDDING state and the Mapping parameter is set to "dataRI",

— continue.

If the SACF is in the STRAY state and the Mapping parameter is set to "dataRI" and the Correlator does not match the Correlator of the previous AF-BEGIN-DIALOGUE request,

— continue.

If the SACF is in the STRAY or BID CONFIRM RECEIVED state and the Correlator matches the Correlator of the previous AF-BEGIN-DIALOGUE request and the Diagnostic parameter is either absent or set to a value different from "association-reserved",

— enter the BUSY state;

— pass the service primitive through.

If the SACF is the contention-loser in the STRAY state and the Correlator matches the Correlator of the previous AF-BEGIN-DIALOGUE request and the Diagnostic parameter is set to "association-reserved" and the Mapping parameter is set to "dataRI",

— enter the BUSY state;

— pass the service primitive through.

10.5.5 AF-BID indication

The SACF shall be the contention-winner.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

If either of the following is true:

a) the SACF is in the FREE state and the *last partner identifier is not valid;* or

b) the SACF is in the STRAY or CLEANUP ROLLBACK INDICATION EXPECTED state;

then

— continue.

If the SACF is in the FREE state and the *last partner identifier is valid* and all of the following conditions are met:

a) the value of the CCR-Token-Requested parameter is set to "true"; and

b) the SACF does not have the *token*; and

NOTE 1 The situation that the value of the CCR-Token-Requested parameter is set to "true" and the TPPM does not have the *token* may arise if a P-TOKEN-GIVE (sync-minor) request crosses an unconfirmed dialogue ending APDU, and the AF-BID request was issued prior to receiving the P-TOKEN-GIVE (sync-minor) indication.

c) according to a local decision, the SACF intends to issue an AF-BID (accepted) response only after the *token* arrives,

then

— enter the BID INDICATION RECEIVED state.

If the SACF is in the FREE state and none of the preceding conditions apply,

a) always

— issue an AF-BID response with the Result parameter set based on a local decision;

NOTE 2 The decision on whether the Result parameter of an AF-BID response is set to "rejected" or to "accepted" will be made depending on whether the association on which the AF-BID indication was received has been reserved for other purposes or not.

b) if the Result parameter is set to "rejected"

— continue;

c) if the Result parameter is set to "accepted"

— enter the BID INDICATION RECEIVED state;

— issue an AF-TOKEN-GIVE (regular) request if the CCR-Token-Requested parameter on the AF-BID indication is set to "true", and the SACF has the *token*.

10.5.6 AF-BID confirm

If the SACF is in the BIDDING state and the result parameter is "accepted",

— *flush the queue* up to and excluding a C-BEGIN request;

— enter the FREE state if a SAF-DETACH-ASSOCIATION request was received;

NOTE 1 The MACF procedures in clause 11 are designed in such a way that a SAF-DETACH-ASSOCIATION request will never be received after a C-BEGIN request was queued and before the queue is entirely flushed.

— enter the BID CONFIRM RECEIVED state if no SAF-DETACH-ASSOCIATION request was received.

If the SACF is in the BIDDING state and the result parameter is "rejected",

— enter the FREE state;

— issue a SAF-ASSOCIATION-LOST indication if no SAF-DETACH-ASSOCIATION request was received;

— *discard the queue*.

NOTE 2 It is a local matter whether the TPPM retries establishing a dialogue or channel on another association; however, in the case where the SAF-ASSOCIATION-LOST indication is not issued because a SAF-DETACH-ASSOCIATION request was already received, retrying is recommended.

10.5.7 AF-END-DIALOGUE request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.8 AF-END-DIALOGUE indication

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

10.5.9 AF-END-DIALOGUE confirm

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.10 AF-U-ERROR request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.11 AF-U-ERROR indication

If the SACF is in the FREE, BIDDING, STRAY, or CLEANUP ROLLBACK INDICATION EXPECTED state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.12 AF-U-ERROR confirm

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.13 AF-ABORT request

If a queue exists and the value of the Mapping parameter is not "abortRI",

— queue the service primitive.

If a queue does not exist or the value of the Mapping parameter is "abortRI",

— pass the service primitive through.

NOTE — If the Mapping parameter is "abortRI", the association will cease to exist when the A-ABORT request is issued.

10.5.14 AF-ABORT (provider, abortRI) indication

NOTE — Upon receipt of an AF-ABORT (provider, abortRI) indication, the association ceases to exist.

If the SACF is in the FREE, BID INDICATION RECEIVED, CLEANUP ROLLBACK INDICATION EXPECTED, or CLEANUP BEGIN INDICATION RECEIVED state,

— continue.

If the SACF is in the STRAY, BIDDING, BID CONFIRM RECEIVED, or BUSY state,

— pass the service primitive through if no SAF-DETACH-ASSOCIATION request was received;

— *discard the queue*.

10.5.15 AF-ABORT (user, dataRI) indication

If the SACF is in the FREE, BIDDING, STRAY, CLEANUP ROLLBACK INDICATION EXPECTED, or state,

— continue.

If the SACF is in the CLEANUP BEGIN INDICATION EXPECTED state,

— enter the FREE state.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.16 A-ABORT request or A-RELEASE (Result=affirmative) response

If the SACF is in the STRAY, BIDDING, BID CONFIRM RECEIVED, or BUSY state,

— *discard the queue*;

— pass the service primitive through.

NOTE — Upon issuance of either of these service primitives, the association ceases to exist.

10.5.17 A-[P-]ABORT indication or A-RELEASE (Result=affirmative) confirm

If the SACF is in the BIDDING, STRAY, BID CONFIRM RECEIVED, or BUSY state,

— pass the service primitive through, if no SAF-

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

DETACH-ASSOCIATION request was received;

10.5.18 AF-GRANT-CONTROL request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.19 AF-GRANT-CONTROL indication

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.20 AF-REQUEST-CONTROL request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.21 AF-REQUEST-CONTROL indication

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.22 AF-HANDSHAKE request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.23 AF-HANDSHAKE indication

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.24 AF-HANDSHAKE confirm

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.25 AF-HANDSHAKE-AND-GRANT-CONTROL request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.26 AF-HANDSHAKE-AND-GRANT-CONTROL indication

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.27 AF-HANDSHAKE-AND-GRANT-CONTROL confirm

If the SACF is in the FREE, BIDDING, or STRAY state,

— continue.

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.28 AF-DEFER request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

10.5.29 AF-DEFER indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.30 AF-PREPARE request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.31 AF-PREPARE indication

If the SACF is in the BUSY state,

— pass the service primitive through.

If the SACF is in the CLEANUP ROLLBACK CONFIRM EXPECTED state,

— continue.

10.5.32 AF-HEURISTIC-REPORT (commitRC) indication, or AF-HEURISTIC-REPORT (recoverDoneRC) indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.33 C-BEGIN request

If the SACF does not have the *token* and is in the

a) STRAY or BID CONFIRM RECEIVED state,

— *establish a queue*; and

— queue the service primitive.

b) BIDDING state,

— queue the service primitive.

NOTE — Only under these conditions will the SACF queue for the *token* on a dialogue. Under other conditions, the TPSUI is assumed to have the *token* (see annex B). For conditions where the SACF will queue for the *token* on a channel, see 10.5.46.

If the SACF has the *token*,

— pass the service primitive through.

10.5.34 C-BEGIN indication

If the SACF is in the FREE state and the last time this SAO was detached from the MACF this was done by a SAF-DETACH-ASSOCIATION request which had the Type parameter set to "begin-fear", or if the SACF is in the CLEANUP BEGIN INDICATION EXPECTED state,

— issue a C-ROLLBACK request;

— enter the CLEANUP ROLLBACK CONFIRM EXPECTED state.

If the SACF is in the BUSY state,

— pass the service primitive through.

If the SACF is in the BIDDING or STRAY state and the last time this SAO was detached from the MACF this was done by a SAF-DETACH-ASSOCIATION request which had the Type parameter set to "begin-fear",

— issue a SAF-ASSOCIATION-LOST indication, if no SAF-DETACH-ASSOCIATION request was received;

— *discard the queue*;

NOTE — It is a local matter whether the TPPM retries establishing a dialogue or channel on another association; however, in the case where the SAF-ASSOCIATION-LOST indication is not issued because a SAF-DETACH-ASSOCIATION request was already received, retrying is recommended.

— issue a C-ROLLBACK request;

— enter the CLEANUP ROLLBACK CONFIRM EXPECTED state.

10.5.35 C-BEGIN confirm

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.36 C-READY indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.37 C-COMMIT indication or C-COMMIT+C-BEGIN indication

If the SACF is in the BUSY state,

— pass the service primitive through.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

10.5.38 AF-ABORT (user, commitRI) indication or AF-ABORT (user, commitRC) indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.39 C-COMMIT confirm

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.40 AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.41 C-ROLLBACK request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.42 C-ROLLBACK indication

NOTE 1 After receipt of a *rollback response* or a *rollback confirm*, the *token* is set according to the TP/CCR rules described in 8.4.2 and sent to the superior of the branch. Since the SACF is not aware of the superior of the branch, it returns the *token* if the SACF is the contention-loser.

If the SACF is in the BUSY state,

— pass the service primitive through.

If the SACF is in the STRAY or BID CONFIRM RECEIVED state,

— enter the BUSY state;

— pass the service primitive through.

If the SACF is in the

a) CLEANUP ROLLBACK CONFIRM EXPECTED state,

1) if the previous request or response was an AF-ABORT (provider, rollbackRI) request with the Diagnostic parameter set to "begin-transaction-reject",

— issue an AF-ABORT (provider, rollbackRC) request with the Diagnostic parameter set to "begin-transaction-reject";

2) if the previous request or response was an AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) response,

— issue an AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) response, with the same values of the corresponding parameters as in the previous AF-BEGIN-DIALOGUE response, except for the Mapping parameter;

3) if the previous request or response was neither an AF-ABORT (provider, rollbackRI) request with Diagnostic parameter set to "begin-transaction-request", nor an AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) response,

— issue a C-ROLLBACK response;

NOTE 2 The semantic of "abort" or "rejected dialogue" carried on the previous C-ROLLBACK request must be repeated on the C-ROLLBACK response (in c 2 and 3 above) because the C-ROLLBACK-RI has been suppressed by the Session Layer when the C-ROLLBACK-RI collision occurs.

4) if the SACF is the contention-loser and owns the *token*,

— issue an AF-TOKEN-GIVE (regular) request;

5) always

— enter the FREE state;

b) CLEANUP ROLLBACK INDICATION EXPECTED state,

1) always

— issue a C-ROLLBACK response;

— enter the FREE state;

2) if the SACF is the contention-loser and owns the *token*,

— issue an AF-TOKEN-GIVE (regular) request.

10.5.43 AF-ABORT (rollbackRI) indication, AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication, or AF-HEURISTIC-REPORT (rollbackRI) indication

NOTE — After receipt of a *rollback response* or a *rollback confirm*, the *token* is set according to the TP/CCR rules described in 8.4.2 and sent to the superior of the branch.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Since the SACF is not aware of the superior of the branch, it returns the *token* if the SACF is the contention-loser.

If the SACF is in the BUSY state,

— pass the service primitive through.

If the SACF is in the STRAY or BID CONFIRM RECEIVED state,

— enter the BUSY state;

— pass the service primitive through.

If the SACF is in the CLEANUP ROLLBACK CONFIRM EXPECTED state and this is an AF-ABORT (rollbackRI) indication,

a) always

— issue a C-ROLLBACK response;

— enter the FREE state;

b) if the SACF is the contention-loser and owns the token,

— issue an AF-TOKEN-GIVE (regular) request.

10.5.44 C-ROLLBACK confirm, AF-HEURISTIC-REPORT (rollbackRC) indication, AF-ABORT (user/provider, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication

NOTE — After receipt of a *rollback response* or a *rollback confirm*, the *token* is set according to the TP/CCR rules described in 8.4.2 and sent to the superior of the branch. Since the SACF is not aware of the superior of the branch, it returns the *token* if the SACF is the contention-loser.

If the SACF is in the BUSY state,

— pass the service primitive through.

If the SACF is in the STRAY or BID CONFIRM RECEIVED state,

— enter the BUSY state;

— pass the service primitive through.

If the SACF is in CLEANUP ROLLBACK CONFIRM EXPECTED state,

— enter the FREE state;

— issue an AF-TOKEN-GIVE (regular) request, if the SACF is a contention-loser and owns the *token*.

10.5.45 AF-RECOVER indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.46 C-RECOVER request or AF-RECOVER request

If the SACF is in BUSY state and

a) at least one of the two following conditions is true,

1) the *token* is owned,

2) a C-RECOVER (ready) indication or an AF-RECOVER (ready) indication was the most recently received primitive (including requests and responses from the MACF) on this association, and this is a C-RECOVER (commit) request,

— pass the service primitive through;

b) otherwise,

— *establish a queue*, if one does not exist;

— queue the service primitive.

10.5.47 C-RECOVER indication

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.48 C-RECOVER confirm

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.49 U-ASE request

If a queue exists,

— queue the service primitive.

If no queue exists,

— pass the service primitive through.

10.5.50 U-ASE indication

If the SACF is in the FREE, BIDDING, or STRAY state,

— suppress the service primitive.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

If the SACF is in the BUSY state,

— pass the service primitive through.

10.5.51 AF-TOKEN-GIVE (regular) indication

If the SACF is the contention-loser in the

a) FREE, STRAY, or BIDDING state,

— issue an AF-TOKEN-GIVE (regular) request;

b) BID CONFIRM RECEIVED state,

— *flush the queue*.

If the SACF is a contention-winner in the

a) STRAY state, and

1) there is a queue,

— *flush the queue*;

— enter the FREE state, if a SAF-DETACH-ASSOCIATION request has been received;

2) there is no queue,

— continue;

— enter the FREE state, if a SAF-DETACH-ASSOCIATION request has been received;

b) BID INDICATION RECEIVED state and received an AF-BID indication with the CCR-Token-requested parameter set to "false",

— continue;

c) CLEANUP ROLLBACK CONFIRM EXPECTED, or FREE state,

— continue;

d) BID INDICATION RECEIVED state and received an AF-BID indication with the CCR-Token-requested parameter set to "true, and

1) has not issued an AF-BID (accepted) response,

— issue an AF-BID (accepted) response;

— issue an AF-TOKEN-GIVE (regular) request;

2) has issued an AF-BID (accepted) response but has not issued an AF-TOKEN-GIVE (regular) request,

— issue an AF-TOKEN-GIVE (regular) request;

e) BUSY state and all of the following are true:

1) the Commit functional unit is selected;

2) an AF-BEGIN-DIALOGUE indication was received and was not preceded by an AF-BID indication; and

3) no AF-TOKEN-GIVE (keep) request was issued; then

— issue an AF-TOKEN-GIVE (keep) request;

10.5.52 AF-TOKEN-GIVE (keep) indication

If the SACF is a contention-loser and is in the STRAY or BUSY state and the value of the Correlator parameter is equal to the Correlator of the last TP-BEGIN-DIALOGUE-RI APDU sent,

— *flush the queue*; and

— pass the service primitive to the U-ASE.

If the SACF is a contention-loser and is either

a) in the FREE state; or

b) in the STRAY or BIDDING state and the value of the Correlator parameter is not equal to the Correlator of the last TP-BEGIN-DIALOGUE-RI APDU sent,

then

— issue an AF-TOKEN-GIVE (regular) request.

10.5.53 AF-TOKEN-GIVE (two-way-recovery) request

If the SACF is in BUSY state and

a) the *token* is owned,

— pass the service primitive through;

b) the *token* is not owned,

— *establish a queue*, if one does not exist;

— queue the service primitive.

10.5.54 AF-TOKEN-GIVE (two-way-recovery) indication

If the association is assigned to a channel and the SACF is in the BUSY state and there is no queue,

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— pass the service primitive through.

If the association is assigned to a channel and the SACF is in the BUSY state and there is a queue,

— *flush the queue*.

10.5.55 P-TOKEN-GIVE (sync-minor) indication

If the SACF is the contention-loser in the

a) FREE, STRAY, or BIDDING state,

— issue an AF-TOKEN-GIVE (regular) request;

b) BUSY state,

— pass the *token* to the U-ASE, if the Dialogue functional unit is selected.

If the SACF is the contention-winner in the

a) STRAY state, and

1) there is a queue,

— *flush the queue*;

— enter the FREE state, if a SAF-DETACH-ASSOCIATION request has been received;

2) there is no queue,

— continue;

b) FREE state,

— continue;

c) BUSY state,

— pass the *token* to the U-ASE, if the Dialogue functional unit is selected.

10.5.56 AF-TOKEN-PLEASE request

If the SACF is in BUSY state,

— pass the service primitive through.

10.5.57 AF-TOKEN-PLEASE indication

If the association is assigned to a channel and the SACF is in the

a) BUSY state,

— pass the service primitive through;

b) FREE, BIDDING, or STRAY state,

— continue.

10.5.58 P-TOKEN-PLEASE indication

If the association is assigned to a dialogue and the SACF is in the BUSY state,

— pass the service primitive to the U-ASE;

otherwise

— continue.

10.5.59 Protocol error

If the SACF is not *attached* to a MACF,

— issue an AF-ABORT (provider, abortRI) request, with the Diagnostic parameter set to "protocol-error".

10.5.60 Other service primitives

If the SACF is in the BUSY state, the following service primitives are always passed through:

AF-END-DIALOGUE response
AF-U-ERROR response
AF-HANDSHAKE response
AF-HANDSHAKE-AND-GRANT-CONTROL response
AF-HEURISTIC-REPORT request
AF-ABORT-AND-HEURISTIC-REPORT request
AF-TOKEN-GIVE (regular) request
AF-TOKEN-GIVE (keep) request
C-BEGIN response
C-READY request
C-COMMIT request
C-COMMIT response
C-COMMIT+C-BEGIN request
C-ROLLBACK response
C-RECOVER response
P-TOKEN-GIVE (sync-minor) request
A-ABORT request

10.6 SACF internal events

10.6.1 Unsolicited BID reject

This procedure is entered by a contention-winner SACF that is in the BID INDICATION RECEIVED state and has not yet issued an AF-BID response.

NOTE — The SACF, originally intending to accept the BID, but not having the *token*, delayed responding until the *token* was received. In the meantime, some internal

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

condition has arisen that warrants rejecting the BID.

— Issue an AF-BID response with the value of the Result parameter set to "rejected";

— enter the FREE state.

10.7 Concatenation

10.7.1 Mapping precedence

This part of ISO/IEC 10026 defines the rules governing the concatenation of TP APDUs and their mapping onto other services. These concatenation rules do not affect lower layer concatenation mechanisms (e.g., the Session Layer concatenation). Alternative mappings to Presentation Services are sometimes used depending on the APDUs being concatenated.

The mapping of any concatenation sequence involving one or more CCR APDUs shall be as specified in ISO/IEC 9805. Concatenation sequences involving only TP APDUs shall be mapped to the User data parameter of the P-DATA service with the exception of those TP APDUs for which a mapping to a different Presentation or ACE service is specifically defined in the procedures in clause 9. Concatenations involving U-ASE APDUs and not involving any CCR APDUs shall be mapped to the User data parameter of the P-DATA service unless otherwise specified in the U-ASE specification. Any such other mapping shall not interfere with the operation or semantics of either CCR or TP.

Sublause 9.5 shows the TP mapping of TP APDUs onto underlying services, if the TP concatenation mechanism is not used (basic unconcatenated TP APDU mapping).

10.7.2 Concatenation rules

This subclause specifies the rules for determining valid concatenations of TP, U-ASE, CCR, and ACSE APDUs.

NOTE 1 This clause does not specify the complete set of rules for determining valid sequences of TP, U-ASE, CCR, and ACSE APDUs. The ACSE and CCR standards themselves impose constraints which are not repeated here. Also the procedures in clauses 9, 10, and 11 further restrict the valid sequences of TP, U-ASE, CCR, and ACSE APDUs.

a) The APDUs in table 33 shall not be concatenated with any APDU;

Table 33 — APDUs that can not be concatenated

TP-BID-RI
TP-BID-RC
TP-BEGIN-DIALOGUE-RC (rejected)
TP-PREPARE-RI
TP-RECOVER-RI
C-ROLLBACK-RI
C-ROLLBACK-RC

b) The APDUs in table 34 shall be first in a concatenation sequence. These APDUs must begin a concatenation sequence because no prior APDU is possible;

Table 34 — APDUs that begin a concatenation sequence

TP-BEGIN-DIALOGUE-RI
TP-BEGIN-DIALOGUE-RC (accepted)

c) The APDUs in table 35 shall be last in a concatenation sequence. These APDUs must end a concatenated sequence because they are part of a confirmed exchange or because no subsequent APDU is possible.

Table 35 — APDUs that end a concatenation sequence

TP-END-DIALOGUE-RI
TP-END-DIALOGUE-RC
TP-ABORT-RI
TP-GRANT-CONTROL-RI
TP-REQUEST-CONTROL-RI
TP-HANDSHAKE-RI
TP-HANDSHAKE-AND-GRANT-CONTROL-RI

Based on a local decision, each APDU in table 35 may be concatenated to an existing concatenation sequence;

d) If the preceding TP-HANDSHAKE-RI or TP-HANDSHAKE-AND-GRANT-CONTROL-RI had the confirmation-urgency field equal to "urgent", or the confirmation-urgency field was absent from the TP-HANDSHAKE-RI, then the corresponding APDU in table 36 shall be last in a concatenation sequence.

If the preceding TP-HANDSHAKE-RI or TP-HANDSHAKE-AND-GRANT-CONTROL-RI had the confirmation-urgency field equal to "normal", then the corresponding APDU in table 36 need not end a concatenation sequence.

Based on a local decision, each APDU in table 36 may be concatenated to an existing concatenation sequence;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 36 — APDUs that conditionally end a concatenation sequence

TP-HANDSHAKE-RC
TP-HANDSHAKE-AND-GRANT-CONTROL-RC

e) The APDUs in table 37 may be concatenated and need not begin nor end a concatenation sequence. Based on a local decision, each APDU in table 37 may be concatenated to an existing concatenation sequence;

Table 37 — APDUs that may be concatenated

TP-U-ERROR-RI
TP-U-ERROR-RC
TP-TOKEN-GIVE-RI
TP-TOKEN-PLEASE-RI
TP-DEFER-RI
TP-HEURISTIC-REPORT-RI
TP-INITIALIZE-RI
TP-INITIALIZE-RC
U-ASE APDU

NOTES

2 The TP-INITIALIZE-RI/RC are the only TP APDUs mapped to the A-ASSOCIATE services. The U-ASE is free to map its APDUs on A-ASSOCIATE services.

3 Some of the TP APDUs in table 37 are not (expected to be) concatenated because they are defined to be mapped into other service in isolation rather than because of any concatentation rule (e.g., TP-TOKEN-GIVE/PLEASE-RI is only ever carried on P-TOKEN-GIVE/PLEASE-RI in isolation).

10.8 Routeing

The router part of the SACF receives CCR, ACSE and Presentation indications and confirms and determines which part of the PM shall receive the service primitive. This ensures that each service primitive is seen by the MACF (and possibly the SACF procedures) only once and that the full combined semantics of any TP APDU carried on the user data of a service primitive is given to the MACF with a single indication or confirm.

If the user data parameter associated with the service primitive is not present, the service primitive will be given to the MACF through the appropriate SACF procedures.

11 MACF description

11.1 Introduction

This clause describes the PM MACF procedures related to the use of TP Service primitives by the TPSUI.

These procedures identify which actions are to be taken by the MACF. Main procedures and Internal Event procedures that pertain to both the TPPM and the CPM, and that pertain only to the CPM, are identified accordingly in the subclause title. Subclause titles without such identification pertain only to the TPPM.

Dashed lists are used exclusively to present the actions.

NOTE — This representation makes it easier for the reader to locate the actions when using the "Index of Actions and Events" which follows clause 15.

11.2 CAF service definition

11.2.1 CAF-PLEASE request

This service is used by the TPPM to request the assignment of a channel for the purpose of initiating recovery. The issuance of the CAF-PLEASE request always results in a subsequent CAF-GIVE indication or CAF-FAIL indication, provided the TPPM still exists.

This service does not relate to any particular channel.

This is an unconfirmed service.

Table 38 gives the parameters of this primitive.

Table 38 — CAF-PLEASE parameters

CAF-PLEASE	
Parameter	**req**
AE-Title	M
Atomic Action Identifier	M
Branch Identifier	M

a) **AE-Title:** Specifies the AE Title of the transaction to be recovered;

b) **Atomic Action Identifier:** Together with the Branch Identifier, specifies the transaction branch to be recovered. Its value can be in the range as specified in ISO/IEC 9805;

c) **Branch Identifier:** Together with the Atomic Action Identifier, specifies the transaction branch to be recovered. Its value can be in the range as specified in ISO/IEC 9805.

The parameter values for "Atomic Action Identifier" and "Branch Identifier" are obtained from the log-ready and log-commit records for the transaction and branch, respectively, being recovered. These parameters are not set explicitly in the procedures that follow, but are assumed to be set on each CAF-PLEASE request.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

11.2.2 CAF-GIVE indication

This service indicates that a *channel has been transferred* to the TPPM for the purpose of recovery. This *transfer of the channel* is made because of a previous CAF-PLEASE request.

Upon issuance of this service primitive, the channel has been completely established and a C-RECOVER request may be immediately issued by the MACF.

NOTE — The SACF may queue the C-RECOVER request if the *token* is not owned.

This service relates to the *channel which is to be transferred.*

This is an unconfirmed service.

Table 39 gives the parameters of this primitive.

Table 39 — CAF-GIVE parameters

CAF-GIVE	
Parameter	**ind**
Channel-Utilization	M

a) **Channel-Utilization:** Specifies the type of recovery to be performed on this channel. The values are:

"one-way-recovery";
"two-way-recovery".

11.2.3 CAF-FAIL indication

This service indicates that a request for the assignment of a channel for recovery can not be satisfied by the CPM. This indication is a response to a previous CAF-PLEASE request.

The TPPM is responsible for issuing a further CAF-PLEASE request if recovery is still required.

This service does not relate to any particular channel.

This is an unconfirmed service.

This service has no parameters.

11.2.4 CAF-DETACH request

This service indicates that the TPPM has no further use for the channel. Issuing a CAF-DETACH request results in the *transfer of the channel* to the CPM.

This service relates to the *channel which is to be transferred.*

This is an unconfirmed service.

Table 40 gives the parameters of this primitive.

Table 40 — CAF-DETACH parameters

CAF-DETACH	
Parameter	**req**
Type	M

a) **Type:** This parameter can take the value:

1) "clean-up", when the channel is being *detached* from the TPPM and a C-RECOVER indication or confirm may be subsequently received by the CPM;

2) "free", when the channel is available for another recovery exchange;

3) "not-used", when the TPPM did not use the channel and is returning it immediately after the issuance of the CAF-GIVE indication.

When this service primitive is referenced in the procedure clauses, the value listed in the parenthetical argument is the value of the Type parameter.

11.2.5 CAF-RECOVER indication

This service indicates that a *channel has been transferred* to the TPPM for the purpose of recovering a specific transaction branch. The indication identifies the transaction branch.

The channel has been completely established and the response to the recovery action can be sent immediately on the channel.

The service relates to a specific transaction branch. After the recovery action has concluded, the channel must be *detached* from the TPPM.

When this service primitive is referenced in the procedure clauses, the first value listed in the parenthetical argument is the value of the Recovery State parameter.

Table 41 gives the parameters of this primitive:

Table 41 — CAF-RECOVER parameters

CAF-RECOVER	
Parameter	**ind**
Recovery State	M
Atomic Action Identifier	M
Branch Identifier	M
Channel-Utilization	M

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

a) **Recovery State:** Specifies the peer's view of the state of the branch. The values are

"ready";
"commit";

b) **Atomic Action Identifier:** Together with the Branch Identifier, specifies the transaction branch to be recovered. Its value can be in the range as specified in ISO/IEC 9805;

c) **Branch Identifier:** Together with the Atomic Action Identifier, specifies the transaction branch to be recovered. Its value can be in the range as specified in ISO/IEC 9805;

d) **Channel-Utilization:** Specifies the type of recovery to be performed on this channel. The values are:

"one-way-recovery";
"two-way-recovery".

11.3 Main procedures

These procedures are invoked by TP, AF, SAF, CAF, and CCR services.

11.3.1 TP-BEGIN-DIALOGUE request

Depending on a local decision, either

— assign an association compatible with this dialogue;

NOTE 1 Whether this association is assigned from the pool of associations, or has been newly established for use with this dialogue is a local matter.

— issue an AF-BEGIN-DIALOGUE request;

— invoke the "Initiating a transaction branch" procedure (see 11.5.4), if the coordination level is "commitment";

or

— issue a TP-BEGIN-DIALOGUE (rejected(provider)) confirm with the Rollback parameter set to "false" and the Diagnostic parameter set to

a) "recipient-unknown", if this dialogue establishment request is being rejected as a result of an A-ASSOCIATE confirm with the Diagnostic parameter set to one of

1) "called AP title not recognized";
2) "called AE qualifier not recognized";
3) "called AP invocation-identifier not recognized";
4) "called AE invocation-identifier not recognized";

b) "no-reason-given" otherwise.

NOTE 2 The TPPM will normally only choose to issue the TP-BEGIN-DIALOGUE (rejected) confirm because it is unable to obtain a compatible association, either from the pool, or by establishing a new association. If a TP-BEGIN-DIALOGUE (rejected) confirm is issued because of a failed attempt to establish an association, it is a local matter to convey the parameter from the A-ASSOCIATE confirm (other than those enumerated above) to the TPSUI.

11.3.2 AF-BEGIN-DIALOGUE indication (TPPM and CPM)

If the dialogue or channel is not to be rejected,

a) if a dialogue, and the *dialogue will not have a coordination level of commitment*,

— create a TPSUI of the type specified by the Recipient-TPSU-Title parameter of the AF-BEGIN-DIALOGUE indication, or a TPSUI of a default type if the parameter is not present;

NOTE 1 From the OSIE perspective, whether a new TPSUI is "created" in a real open system or whether an old instance is reused, is a local matter.

— issue a TP-BEGIN-DIALOGUE indication;

b) if a dialogue, and the *dialogue will have a coordination level of commitment*,

— continue;

NOTE 2 If the *dialogue will have a coordination level of "commitment"*, a TP-BEGIN-DIALOGUE indication will be issued on receipt of a C-BEGIN indication (see 11.3.37).

c) if a channel

— issue an AF-BEGIN-DIALOGUE (accepted, dataRI) response.

If the dialogue or channel is to be rejected,

— issue an AF-BEGIN-DIALOGUE (rejected(provider), dataRI) response with the value of the

a) Diagnostic parameter, if a dialogue, set to, as appropriate, one of:

1) "recipient-tpsu-title-unknown";
2) "tpsu-not-available(permanent)";
3) "tpsu-not-available(transient)";
4) "recipient-tpsu-title-required";
5) "functional-unit-not-supported";
6) "functional-unit-combination-not-supported";

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

7) "no-reason-given";

b) Diagnostic parameter, if a channel, set to, as appropriate, one of:

1) "functional-unit-not-supported";
2) "tppm-recovery-not-available";
3) "two-way-recovery-not-supported";
4) "no-reason-given";

c) Functional-Units parameter set to the functional units which are supported, if the Diagnostic parameter is set to "functional-unit-not-supported";

— issue a SAF-DETACH-ASSOCIATION (free) request if the Unchained Transactions functional unit is not selected, or if a channel;

— issue a SAF-DETACH-ASSOCIATION (begin-fear) request if the Unchained Transactions functional unit is selected and the *dialogue will not have a coordination level of "commitment"*;

— issue a SAF-DETACH-ASSOCIATION (begin-indication-expected) request if the *dialogue will have a coordination level of "commitment"*.

11.3.3 TP-BEGIN-DIALOGUE response

If the coordination level is "none" and the Result parameter is

a) "accepted",

— invoke the "First request/response" procedure (see 11.5.3);

b) "rejected",

— issue an AF-BEGIN-DIALOGUE (rejected(user), dataRI) response;

— issue a SAF-DETACH-ASSOCIATION (begin-fear) request if the Unchained Transactions functional unit is selected;

— issue a SAF-DETACH-ASSOCIATION (free) request if the Unchained Transactions functional unit is not selected.

If the coordination level is "commitment" and the Result parameter is

a) "accepted", and the TPPM is in the

1) ACTIVE state,

— invoke the "First request/response" procedure (see 11.5.3);

2) DECIDED (rollback) state,

— continue;

NOTE 1 This situation occurs when the TPPM received the C-ROLLBACK indication before the TPSUI has issued TP-BEGIN-DIALOGUE (accepted) response. The TP-BEGIN-DIALOGUE-RC APDU will be sent as User Data of the *rollback response*, which will be issued upon the receipt of a TP-DONE request.

b) "rejected", and the TPPM is in the

1) ACTIVE state,

— issue an AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) response;

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request;

— cease to be part of the transaction;

2) DECIDED (rollback) state,

NOTE 2 The only time the TPPM could be in this state is if it is a leaf and it received a *rollback indication* from the superior.

— issue an AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) response;

— issue a SAF-DETACH-ASSOCIATION (free) request;

— cease to be part of the transaction.

11.3.4 AF-BEGIN-DIALOGUE (accepted) confirm on a Dialogue

If the value of the Mapping parameter is "rollbackRC", the TPPM shall be in the DECIDED (rollback) state.

If the value of the Mapping parameter is "dataRI" or "rollbackRC", do the following:

a) if the confirmation parameter of the TP-BEGIN-DIALOGUE request was "always" and a TP-U-ABORT request was not received,

— issue a TP-BEGIN-DIALOGUE confirm with the rollback parameter set to "false";

b) if the confirmation parameter of the TP-BEGIN-DIALOGUE request was "always" and the Mapping parameter is "dataRI" and a TP-U-ABORT request was received,

— continue;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

c) if the confirmation parameter of the TP-BEGIN-DIALOGUE request was "negative",

— continue;

d) if the value of the Mapping parameter is "rollbackRC",

1) if a TP-U-ABORT request was received and no AF-ABORT request was issued,

— issue an AF-ABORT (user, dataRI) request;

NOTE — If a TP-U-ABORT request was received, an AF-ABORT request could not have been issued if the Mapping parameter is "rollbackRC".

— issue a SAF-DETACH-ASSOCIATION (free) request;

2) if the *last rollback confirm was received* and this is an intermediate node and the superior *dialogue has not been detached,*

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

3) if the *last rollback confirm was received* and the *superior dialogue is not chaining* and *rollback reporting has completed,*

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

4) if the Unchained Transactions functional unit is selected on the dialogue and no TP-U-ABORT request was received,

— *close the PSAP.*

11.3.5 AF-BEGIN-DIALOGUE (rejected, dataRI) confirm on a Dialogue

If the Diagnostic parameter is set to "association-reserved", set the value of that parameter for the TP-BEGIN-DIALOGUE confirm to "no-reason-given".

If the coordination level is "none",

— issue a TP-BEGIN-DIALOGUE confirm with the value of the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (free) request.

If the coordination level is "commitment" and either

a) the Unchained Transactions functional unit is selected and the Type parameter is "rejected(user)"; or

b) the Type parameter is "rejected(provider)";

and the TPPM is in the

a) ACTIVE state and a TP-COMMIT request was received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "true";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

b) ACTIVE state and a TP-COMMIT request was not received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

c) DECIDED (rollback) state,

1) always

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request;

2) if a TP-U-ABORT request was not received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "false";

NOTE — *A TP-DONE request is now owed.*

3) if a TP-U-ABORT request was received and the *last rollback confirm was received,*

i) if this is an intermediate node and the superior *dialogue has not been detached,*

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

ii) if the superior *dialogue is not chaining* and *rollback reporting has completed,*

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

11.3.6 AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm

If the coordination level is "commitment" and if the TPPM is in the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

a) ACTIVE state and a TP-COMMIT request was received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "true";

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

— issue a SAF-DETACH-ASSOCIATION (free) request;

b) ACTIVE state and a TP-COMMIT request was not received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "false";

— issue a C-ROLLBACK response;

— issue a SAF-DETACH-ASSOCIATION (free) request;

c) DECIDED (rollback) state,

1) always

— issue a C-ROLLBACK response;

— issue a SAF-DETACH-ASSOCIATION (free) request;

2) if a TP-U-ABORT request was not received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "false";

NOTE — *A TP-DONE request is now owed.*

3) if a TP-U-ABORT request was received and the *last rollback confirm was received,*

i) if this is an intermediate node and the superior *dialogue has not been detached,*

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

ii) if the superior *dialogue is not chaining* and *rollback reporting has completed,*

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

11.3.7 AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) confirm

If the coordination level is commitment, and the TPPM is in the DECIDED (rollback) state,

a) always

— issue a SAF-DETACH-ASSOCIATION (free) request;

b) if a TP-U-ABORT request was not received,

— issue a TP-BEGIN-DIALOGUE confirm with the Rollback parameter set to "false";

NOTE — *A TP-DONE request is now owed.*

c) if a TP-U-ABORT request was received and the *last rollback confirm was received,*

1) if this is an intermediate node and the superior *dialogue has not been detached,*

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

2) if the superior *dialogue is not chaining* and *rollback reporting has completed,*

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6).

11.3.8 AF-BEGIN-DIALOGUE confirm (CPM)

If the result parameter is "accepted" and a TPPM with a branch corresponding to the Atomic Action identifier and Branch Identifier specified in a previous CAF-PLEASE request is found,

— *transfer the channel* to the requesting TPPM;

— issue a CAF-GIVE indication to the TPPM with a branch corresponding to the Atomic Action identifier and Branch Identifier specified in a previous CAF-PLEASE request.

If the result parameter is "accepted" and a TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in a previous CAF-PLEASE request is not found,

— retain control of the channel.

NOTE — The CPM may make local decisions to allocate the channel to another TPPM in response to a subsequent CAF-PLEASE request, or to terminate the channel, or if the channel supports two-way-recovery to pass the *token* to the remote CPM.

If the result parameter is "rejected(provider)",

— issue a CAF-FAIL indication to the TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in the previous CAF-PLEASE

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

request;

— issue a SAF-DETACH-ASSOCIATION (free) request.

11.3.9 SAF-ASSOCIATION-LOST indication

NOTE 1 It is a local matter whether an implementation retries establishing a dialogue on another association.

If the coordination level is "none",

— issue a TP-BEGIN-DIALOGUE (rejected(provider)) confirm with the value of the Rollback parameter set to "false" and with the Diagnostic parameter set to "no-reason-given".

If the coordination level is "commitment" and if the TPPM is in the

a) ACTIVE state and a TP-COMMIT request was received,

— issue a TP-BEGIN-DIALOGUE (rejected(provider)) confirm with the Rollback parameter set to "true" and with the Diagnostic parameter set to "no-reason-given";

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

b) ACTIVE state and a TP-COMMIT request was not received,

— issue a TP-BEGIN-DIALOGUE (rejected(provider)) confirm with the Rollback parameter set to "false" and with the Diagnostic parameter set to "no-reason-given";

c) DECIDED (rollback) state,

1) if a TP-U-ABORT request was not received,

— issue a TP-BEGIN-DIALOGUE (rejected(provider)) confirm with the Rollback parameter set to "false" and with the Diagnostic parameter set to "no-reason-given";

NOTE 2 *A TP-DONE request is now owed.*

2) if a TP-U-ABORT request was received and the *last rollback confirm was received,*

i) if this is an intermediate node and the superior *dialogue has not been detached,*

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

ii) if the superior *dialogue is not chaining* and *rollback reporting has completed,*

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6).

11.3.10 SAF-ASSOCIATION-LOST indication (CPM)

— Issue a CAF-FAIL indication to the TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in the previous CAF-PLEASE request;

— continue.

11.3.11 TP-END-DIALOGUE request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-END-DIALOGUE request.

If the value of the confirmation parameter is "false",

— issue a SAF-DETACH-ASSOCIATION (begin-fear) request if the Unchained Transactions functional unit is selected and this is a superior dialogue;

— issue a SAF-DETACH-ASSOCIATION (free) request if the Unchained Transactions functional unit is not selected or this is a subordinate dialogue.

11.3.12 AF-END-DIALOGUE indication

If the coordination level is "none", and

a) the value of the Confirmation parameter is "false",

— issue a TP-END-DIALOGUE indication;

— issue a SAF-DETACH-ASSOCIATION (free) request;

b) the value of the Confirmation parameter is "true" and the dialogue is in a *user error purging period,*

1) if the Polarized Control functional unit is selected,

— issue a TP-END-DIALOGUE indication;

2) if the Shared Control functional unit is selected,

— continue;

c) the value of the Confirmation parameter is "true",

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

the dialogue is not in a *user error purging period*, and

1) a dialogue termination request is outstanding and the Shared Control functional unit is selected,

— issue a TP-P-ABORT indication with the value of the Diagnostic parameter set to "end-dialogue-collision" and the value of the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (free) request;

2) a dialogue termination request is not outstanding,

— issue a TP-END-DIALOGUE indication.

If the coordination level is "commitment", the indication was from a subordinate, a TP-BEGIN-TRANSACTION request has been received, a C-BEGIN confirm was not received, and the TPPM is in the

a) ACTIVE state, a TP-COMMIT request was not received, and the Confirmation parameter is set to "false",

— issue a TP-P-ABORT indication with the Diagnostic parameter set to "begin-transaction-end-dialogue-collision" and with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

b) ACTIVE state, a TP-COMMIT request was received, and the Confirmation parameter is set to "false",

— issue a TP-P-ABORT indication with the Diagnostic parameter set to "begin-transaction-end-dialogue-collision" and with the Rollback parameter set to "true";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

c) ACTIVE state, the Confirmation parameter is set to "true", and the dialogue is in the *transaction initiation purging period*,

— continue;

d) ACTIVE state, a TP-COMMIT request was not received, the Confirmation parameter is set to "true",, and the dialogue is not in the *transaction initiation purging period*,

— issue a TP-P-ABORT indication with the Diagnostic parameter set to "begin-transaction-end-dialogue-collision" and with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

e) ACTIVE state, a TP-COMMIT request was received, the Confirmation parameter is set to "true",, and the dialogue is not in the *transaction initiation purging period*,

— issue a TP-P-ABORT indication with the Diagnostic parameter set to "begin-transaction-end-dialogue-collision" and with the Rollback parameter set to "true";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

f) DECIDED (rollback) state and the dialogue is not in the *transaction initiation purging period*,

1) always

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request;

2) if a TP-U-ABORT request was not received,

— issue a TP-P-ABORT indication with the Diagnostic parameter set to "begin-transaction-end-dialogue-collision" and with the Rollback parameter set to "false";

NOTE — *A TP-DONE request is now owed.*

3) if a TP-U-ABORT request was received and the *last rollback confirm was received*,

i) if this is an intermediate node and the superior *dialogue has not been detached*,

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

ii) if the superior *dialogue is not chaining* and *rollback reporting has completed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

g) DECIDED (rollback) state and the dialogue is in the *transaction initiation purging period*,

— continue.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

11.3.13 AF-END-DIALOGUE indication (CPM)

— Issue a SAF-DETACH-ASSOCIATION (free) request.

11.3.14 TP-END-DIALOGUE response

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-END-DIALOGUE response;

— issue a SAF-DETACH-ASSOCIATION (free) request.

11.3.15 AF-END-DIALOGUE confirm

If the Confirmation parameter of the TP-END-DIALOGUE request previously issued was "true",

— issue a TP-END-DIALOGUE confirm;

— issue a SAF-DETACH-ASSOCIATION (free) request.

11.3.16 TP-U-ERROR request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-U-ERROR request.

11.3.17 AF-U-ERROR indication

If the TPPM is in the ACTIVE state and already issued an AF-PREPARE request on this dialogue,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5).

If the TPPM is in the ACTIVE state and no AF-PREPARE request has been issued, or if the coordination level is "none",

a) if the Shared Control functional unit was selected

— issue a TP-U-ERROR indication;

— issue an AF-U-ERROR response if both of the following conditions are met:

1) this is a subordinate dialogue or an AF-BEGIN-DIALOGUE response was issued; and

NOTE — The AF-U-ERROR response will be issued after the AF-BEGIN-DIALOGUE response.

2) there is no dialogue termination request outstanding and no handshake request outstanding;

b) if the Polarized Control functional unit was selected

— continue, if the TPPM is in a *user error purging period*;

— issue a TP-U-ERROR indication if the TPPM is not in a *user error purging period*.

If the TPPM is in the DECIDED (rollback) state,

— continue.

11.3.18 AF-U-ERROR confirm

If the Shared Control functional unit is selected,

— continue.

11.3.19 TP-U-ABORT request

NOTE 1 There are some cases where an AF-ABORT request will not be issued by this procedure. If the transaction branch has not completed the termination phase, the issuance of the AF-ABORT request is deferred until the necessary response/request is issued to complete the transaction. This is required because, according to CCR rules, nothing can be issued until this time. If the AF-ABORT request is to be deferred, this procedure will either continue or initiate rollback.

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received and the TPPM is not in the DECIDED (rollback) state.

If the coordination level is "none",

— issue an AF-ABORT (user, dataRI) request;

— issue a SAF-DETACH-ASSOCIATION (begin-fear) request if the Unchained Transactions functional unit is selected and this is a superior dialogue;

— issue a SAF-DETACH-ASSOCIATION (free) request if the Unchained Transactions functional unit is not selected or this is a subordinate dialogue.

If the coordination level is "commitment", is in the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

a) ACTIVE state,

— issue an AF-ABORT (user, rollbackRI) request if the dialogue is to a subordinate;

NOTE 2 The TPPM does not issue a C-ROLLBACK request to the superior at this time. It must wait for TP-DONE and all *rollback confirms*.

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

b) READY state,

— continue;

c) DECIDED (commit) state, and

1) the TP-U-ABORT request pertains to the superior dialogue, and the TPPM

i) has received a C-COMMIT+C-BEGIN indication

— invoke the "Rollback next transaction" procedure (see 11.5.11);

ii) has received a C-COMMIT indication

— continue;

2) the TP-U-ABORT request pertains to a subordinate dialogue, and the TPPM

i) has sent a C-COMMIT+C-BEGIN request,

— invoke the "Rollback next transaction" procedure (see 11.5.11);

ii) has sent a C-COMMIT request and has received a *commit confirm*

— issue an AF-ABORT (user, dataRI) request;

— issue a SAF-DETACH-ASSOCIATION (free) request;

iii) has sent a C-COMMIT request and has not received a *commit confirm*

— continue;

d) DECIDED (rollback) state and a *rollback response* was issued or a *rollback confirm* was received and received,
and the dia P-U-ABORT request pertains to the *purging perio* ogue and

i) a C-BEGIN indication was not received while in the DECIDED (rollback) state,

— issue an AF-ABORT (user, dataRI) request;

— issue a SAF-DETACH-ASSOCIATION (begin-fear) request if the Chained Transactions functional unit is selected;

NOTE 3 *A TP-DONE request is owed*, which will invoke the "Initiating Next Transaction after Rollback" procedure.

— *open the PSAP* if the Unchained Transactions functional unit is selected;

— issue a SAF-DETACH-ASSOCIATION (free) request if the Unchained Transactions functional unit is selected;

ii) a C-BEGIN indication was received while in the DECIDED (rollback) state,

— issue an AF-ABORT (user, rollbackRI) request on the superior dialogue;

— *open the PSAP*;

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request;

2) the TP-U-ABORT request pertains to a subordinate dialogue,

— issue an AF-ABORT (user, dataRI) request;

— *open the PSAP*, if the Unchained Transactions functional unit is selected;

— issue a SAF-DETACH-ASSOCIATION (free) request;

e) DECIDED (rollback) state and neither a *rollback response* was issued nor a *rollback confirm* was received,

— continue.

11.3.20 AF-ABORT (user, dataRI) indication

If the coordination level is "none",

— issue a TP-U-ABORT indication with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (free) request.

If the coordination level is "commitment", the indication was from a subordinate, and the TPPM is in the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

a) ACTIVE state, and

1) a TP-COMMIT request was not received;

2) a C-BEGIN confirm was not received;

then

— issue a TP-U-ABORT indication with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

b) ACTIVE state, and

1) a TP-COMMIT request was received;

2) a C-BEGIN confirm was not received;

then

— issue a TP-U-ABORT indication with the Rollback parameter set to "true";

— issue a SAF-DETACH-ASSOCIATION (rollback-indication-expected) request;

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

c) DECIDED (rollback) state, and

1) if a *rollback confirm* was received or a *rollback response* was issued, and the *dialogue is chaining*;

— issue a TP-U-ABORT indication with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (free) request;

NOTE 1 In this situation, a TP-U-ABORT request cannot have been previously received on this association because, in that case, the association would have been immediately or eventually *detached* and the event causing this procedure invocation could not have happened.

2) if a *rollback confirm* was not received and the Unchained Transactions functional unit is selected,

i) if a TP-U-ABORT request was not received,

— issue a TP-U-ABORT indication with the Rollback parameter set to "false";

ii) always

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request.

iii) if the *last rollback confirm was received* and this is an intermediate node and the superior *dialogue has not been detached*,

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

iv) if the *last rollback confirm was received*, the superior *dialogue is not chaining*, and *rollback reporting has completed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6).

If the coordination level is "commitment" and the indication is from the superior, and the TPPM is in the DECIDED (rollback) state and has issued a *rollback response* to or received a *rollback confirm* from the superior,

— issue a TP-U-ABORT indication with the Rollback parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (free) request.

NOTE 2 In this situation, a TP-U-ABORT request could not have been previously received on this dialogue because the association would have been *detached* since the rollback report to the superior occurred (and then the TPPM would not have received the AF-ABORT indication). Therefore, *a TP-DONE request must be owed*, preventing the transaction after the rollback from beginning. In addition, the *superior dialogue is* necessarily *chaining* and a C-BEGIN indication could not have been received in this state, since this would have led to *closing the PSAP*.

11.3.21 Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm on a dialogue

NOTE — The A-ABORT request in the title of this procedure refer to when these service are issued by the U-ASE or some part of the SACF to ACSE. The A-ABORT indication includes the case where the service primitive has the value of the Abort Source parameter equal to "ACSE service-user", and is therefore indicated by ACSE to the U-ASE.

If this is a protocol error or internal error,

— issue an AF-ABORT (provider, abortRI) request with the Diagnostic parameter set to

1) "protocol error", if this is a protocol error;

2) "transient-failure" or "permanent-failure", based on a local decision, if this is an internal error.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

If no TP-U-ABORT request was received,

— issue a TP-P-ABORT indication with the Rollback parameter set to

a) "true", if the TPPM is in the ACTIVE state, does not have a dialogue establishment indication outstanding, and the coordination level is "commitment";
b) "false", otherwise;

and the Diagnostic parameter set to

a) "protocol-error", if this is a protocol error;
b) "transient-failure" or "permanent-failure", based on a local decision, if this is an internal error;
c) "permanent-failure", if this is an A-RELEASE response, an A-RELEASE confirm, an A-[P-] ABORT indication, or an A-ABORT request; or
d) to the Diagnostic parameter on the AF-ABORT indication, if this is an AF-ABORT indication;

— cease to be part of the transaction, if there is a dialogue establishment indication outstanding.

If the dialogue is with the superior, the coordination level is "commitment", and the TPPM is in the

a) ACTIVE state and does not have a dialogue establishment indication outstanding,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

b) READY state

— issue a CAF-PLEASE request with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the superior, contained in the log-ready record;

c) DECIDED (commit) state

— invoke the "Rollback next transaction" procedure (see 11.5.11) if a C-COMMIT+C-BEGIN indication was received from the superior;

d) DECIDED (rollback) state

1) if a TP-U-ABORT request was received and the *last rollback confirm was received*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

2) otherwise,

— continue.

If the dialogue is with the subordinate, the coordination level is "commitment", and the TPPM is in the

a) ACTIVE state,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

b) READY state,

— continue;

c) DECIDED (commit) state,

— invoke the "Rollback next transaction" procedure (see 11.5.11) if a C-COMMIT+C-BEGIN request was issued;

— issue a CAF-PLEASE request if a *commit confirm* has not been received, with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the subordinate, contained in the log-commit record;

d) ACTIVE, or DECIDED (rollback) state, and both of the following are true:

1) an AF-PREPARE request has been issued to this subordinate; and

2) no *rollback confirm* or *rollback indication* has been received from this subordinate,

then

— issue a TP-HEURISTIC-REPORT indication with the Heuristic-Report parameter set to "heuristic-hazard";

— *write a log-damage record* with the value "heuristic-hazard" if no log-damage record exists;

e) DECIDED (rollback) state, neither a *rollback confirm* was received nor a *rollback response* was issued, the *last rollback confirm was received*, and

1) if this is an intermediate node and the superior *dialogue has not been detached*,

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

2) if the *superior dialogue is not chaining and rollback reporting has completed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

f) DECIDED (rollback) state,

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— continue.

11.3.22 Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm on a channel

If this is a protocol error or internal error and the association has not been aborted,

— issue an AF-ABORT (provider, abortRI) request with the Diagnostic parameter set to

1) "protocol error", if this is a protocol error;

2) "transient-failure" or "permanent-failure", based on a local decision, if this is an internal error.

If the channel was with the superior and the TPPM is in the

a) READY state,

— issue a CAF-PLEASE request with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the superior, contained in the log-ready record.

b) DECIDED (commit) state,

— continue.

If the channel was with the subordinate and the TPPM is in the

a) READY state,

— continue;

b) DECIDED (commit) state,

— issue a CAF-PLEASE request with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the superior, contained in the log-ready record.

11.3.23 Protocol error, internal error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-RELEASE (Result=affirmative) response, or A-RELEASE (Result=affirmative) confirm (CPM)

If this is a protocol error or internal error,

— issue an AF-ABORT (provider, abortRI) request with the Diagnostic parameter set to

1) "protocol error", if this is a protocol error;
2) "transient-failure" or "permanent-failure", based on a local decision, if this is an internal error.

Always

— issue a CAF-FAIL indication to the TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in the previous CAF-PLEASE request.

11.3.24 TP-GRANT-CONTROL request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-GRANT-CONTROL request.

11.3.25 AF-GRANT-CONTROL indication

If the coordination level is "none",

— issue a TP-GRANT-CONTROL indication.

If the coordination level is "commitment" and the TPPM is in the

a) ACTIVE state,

— issue a TP-GRANT-CONTROL indication;

b) DECIDED (rollback) state,

— continue.

11.3.26 TP-REQUEST-CONTROL request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-REQUEST-CONTROL request.

11.3.27 AF-REQUEST-CONTROL indication

— Continue if one of the following conditions is met:

a) the TPPM does not have the control of the dialogue;

b) the TPPM has an outstanding dialogue termination request;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

c) the TPPM has already invoked an AF-PREPARE request for the current provider-supported transaction;

d) the TPPM is in the DECIDED (rollback) state.

Otherwise,

— issue a TP-REQUEST-CONTROL indication.

11.3.28 TP-HANDSHAKE request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-HANDSHAKE request.

11.3.29 AF-HANDSHAKE indication

Do only the first applicable action of the following:

a) if the TPPM is in the DECIDED (rollback) state and the coordination level is "commitment",

— continue;

b) if the Shared Control functional unit is selected and the TPPM is in a *user error purging period*,

— continue;

c) if the TPPM is in the ACTIVE state, a C-READY indication was not received, the Shared Control functional unit is selected and an AF-PREPARE request had already been issued on this dialogue,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

d) if the TPPM is not in a *user error purging period*,

— issue a TP-HANDSHAKE indication;

e) if the Polarized Control functional unit was selected and the TPPM is in a *user error purging period*,

— issue a TP-HANDSHAKE indication.

11.3.30 TP-HANDSHAKE response

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-HANDSHAKE response.

11.3.31 AF-HANDSHAKE confirm

If the coordination level is "none",

— issue a TP-HANDSHAKE confirm.

If the coordination level is "commitment" and the TPPM is in the

a) ACTIVE state,

— issue a TP-HANDSHAKE confirm;

b) DECIDED (rollback) state,

— continue.

11.3.32 TP-HANDSHAKE-AND-GRANT-CONTROL request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue an AF-HANDSHAKE-AND-GRANT-CONTROL request.

11.3.33 AF-HANDSHAKE-AND-GRANT-CONTROL indication

If the coordination level is "none",

— issue a TP-HANDSHAKE-AND-GRANT-CONTROL indication.

If the coordination level is "commitment" and the TPPM is in the

a) ACTIVE state,

— issue a TP-HANDSHAKE-AND-GRANT-CONTROL indication;

b) DECIDED (rollback) state,

— continue.

11.3.34 TP-HANDSHAKE-AND-GRANT-CONTROL response

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

been received;

— issue an AF-HANDSHAKE-AND-GRANT-CONTROL response.

11.3.35 AF-HANDSHAKE-AND-GRANT-CONTROL confirm

If the coordination level is "none",

— issue a TP-HANDSHAKE-AND-GRANT-CONTROL confirm.

If the coordination level is "commitment" and the TPPM is in the

a) ACTIVE state,

— issue a TP-HANDSHAKE-AND-GRANT-CONTROL confirm;

b) DECIDED (rollback) state,

— continue.

11.3.36 TP-BEGIN-TRANSACTION request

NOTE — The Session Layer synchronize-minor token is needed in order to issue this request. The TPPM guarantees that this token will be positioned with the issuer unless the TPSUI or U-ASE have moved it, in which case it is the responsibility of the TPSUI to get the *token* before issuing this request. This is a constraint on the movement of the *token* by the U-ASE, described in annex B.

— Invoke the "Initiating a transaction branch" procedure (see 11.5.4).

11.3.37 C-BEGIN indication

The indication shall be from the superior in the dialogue tree. The Commit functional unit shall be selected. Either the coordination level shall not be "commitment", or the TPPM shall be in the DECIDED (rollback) state and the *dialogue shall be chaining.*

If the Shared Control functional unit is selected and a dialogue termination request is outstanding,

— issue a TP-P-ABORT indication with the Diagnostic parameter set to "begin-transaction-end-dialogue-collision" and the Rollback parameter set to "false";

— issue a C-ROLLBACK request;

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request.

If the TPPM is a root in a transaction tree and a dialogue termination request is not outstanding,

— issue an AF-ABORT (provider, rollbackRI) request with the Diagnostic parameter set to "begin-transaction-reject";

— issue a TP-P-ABORT indication with the value of the Rollback parameter set to "false"; and with the Diagnostic parameter set to "begin-transaction-reject";

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request.

NOTE 1 An implementation may choose to wait for the current transaction to complete and then accept the C-BEGIN indication. From a conformance testing view, this is equivalent to not being able to process the C-BEGIN indication until the current transaction has completed (when it can then be successfully processed, instead of rejected). The TPPM procedures and state tables do not provide procedures to allow waiting for the transaction to complete.

If the TPPM is not a root in the transaction tree and a dialogue termination request is not outstanding,

a) if a TP-BEGIN-DIALOGUE indication has been issued and the TPPM is not in the DECIDED (rollback) state,

— issue a TP-BEGIN-TRANSACTION indication;

— become a leaf node;

— enter the ACTIVE state.

b) if a TP-BEGIN-DIALOGUE indication has not been issued,

— create a TPSUI of the type specified by the Recipient-TPSU-Title parameter of the AF-BEGIN-DIALOGUE indication, or a TPSUI of a default type if the parameter is not present;

NOTE 2 From the OSIE perspective, whether a new TPSUI is "created" in a real open system or whether an old instance is reused, is a local matter.

— issue a TP-BEGIN-DIALOGUE indication with the parameters as specified in the previously received AF-BEGIN-DIALOGUE indication;

NOTE 3 If the *dialogue will have a coordination level of "commitment"*, the TP-BEGIN-DIALOGUE indication is issued as above since the C-BEGIN indication has arrived.

— become a leaf node;

— enter the ACTIVE state.

c) if an AF-BEGIN-DIALOGUE response was issued,

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— issue a C-BEGIN response;

d) if the TPPM is in the DECIDED (rollback) state and *a TP-DONE request is owed*,

— *close the PSAP*;

e) if the TPPM is in the DECIDED (rollback) state, and *a TP-DONE request is not owed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6).

11.3.38 C-BEGIN confirm

— Continue.

11.3.39 TP-DATA request

— Invoke the "First request/response" procedure (see 11.5.3), if this is a superior dialogue and if no AF-BEGIN-DIALOGUE response has been issued on the dialogue since an AF-BEGIN-DIALOGUE indication has been received;

— issue a U-ASE request.

11.3.40 U-ASE indication

If the TPPM is in a *user error purging period*,

— continue;

otherwise,

a) if the coordination level is "none"

— issue a TP-DATA indication;

b) if the coordination level is "commitment" and the TPPM is in the

1) ACTIVE state

i) and a TP-COMMIT request has been received, and this indication occurred on a dialogue with the Shared Control functional unit selected or on a dialogue with the Polarized Control functional unit selected after an AF-PREPARE request with the Data-Permitted parameter set to "true" has been issued,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

ii) otherwise,

— issue a TP-DATA indication;

2) DECIDED (rollback) state

— continue.

11.3.41 TP-DEFERRED-END-DIALOGUE request

— Optionally issue an AF-DEFER (end-dialogue) request according to a local decision.

NOTE — It is an implementation option to either invoke the AF-DEFER request immediately or to delay it until the AF-PREPARE request is invoked.

11.3.42 TP-DEFERRED-GRANT-CONTROL request

— Optionally issue an AF-DEFER (grant-control) request according to a local decision.

NOTE — It is an implementation option to either invoke the AF-DEFER request immediately or to delay it until the AF-PREPARE request is invoked.

11.3.43 AF-DEFER indication

If the coordination level is "commitment", and the TPPM is in the

a) ACTIVE state, either

— issue a TP-DEFERRED-END-DIALOGUE indication if the Type parameter is "end-dialogue"; or

— issue a TP-DEFERRED-GRANT-CONTROL indication if the Type parameter is "grant-control";

b) DECIDED (rollback) state,

— continue.

11.3.44 TP-PREPARE request

If a TP-DEFERRED-END-DIALOGUE request was received and no AF-DEFER (end-dialogue) request was issued,

— issue an AF-DEFER (end-dialogue) request.

If no TP-DEFERRED-END-DIALOGUE request was received, a TP-DEFERRED-GRANT-CONTROL request was received and no AF-DEFER (grant-control) request was issued,

— issue an AF-DEFER (grant-control) request.

Always

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— issue an AF-PREPARE request.

11.3.45 TP-COMMIT request

For each subordinate to which an AF-PREPARE request has not already been issued,

a) if a TP-DEFERRED-END-DIALOGUE request was received and no AF-DEFER (end-dialogue) request was issued,

— issue an AF-DEFER (end-dialogue) request;

b) if no TP-DEFERRED-END-DIALOGUE request was received, a TP-DEFERRED-GRANT-CONTROL request was received and no AF-DEFER (grant-control) request was issued,

— issue an AF-DEFER (grant-control) request;

c) always

— issue an AF-PREPARE request with the Data-Permitted parameter

1) absent, if the Shared Control functional unit is selected;

2) set to "false", if the Polarized Control functional unit is selected.

If the TPPM is the root,

— invoke the "Making commitment decision" procedure (see 11.5.7) if the *last ready was received.*

If the TPPM is an intermediate or leaf,

— invoke the "Entering READY state" procedure (see 11.5.2), if the *last ready was received.*

11.3.46 AF-PREPARE indication

If the coordination level is "commitment" and TPPM is in the

a) ACTIVE state, and

1) the TPPM is in a *user error purging period* or a handshake request is outstanding,

NOTE — a handshake request outstanding may occur only if the Shared Control functional unit was selected.

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

2) otherwise,

— issue a TP-PREPARE indication;

b) DECIDED (rollback) state

— continue.

11.3.47 C-READY indication

The TPPM shall be in the ACTIVE state and an AF-PREPARE request shall have been issued.

If the *last ready was not received* and

a) a TP-COMMIT request was not received,

NOTE — In this case a TP-PREPARE request was received.

— issue a TP-READY indication;

b) a TP-COMMIT request was received,

— continue.

If the *last ready was received* and

a) the TPPM is a root node,

— invoke the "Making commitment decision" procedure (see 11.5.7);

b) the TPPM is an intermediate node,

— invoke the "Entering READY state" procedure (see 11.5.2).

11.3.48 C-COMMIT indication or C-COMMIT+C-BEGIN indication

The TPPM shall be in the READY state.

— Invoke the "Receiving commit order" procedure (see 11.5.8).

If the TPPM is an intermediate node,

— invoke the "Sending commit order" procedure (see 11.5.12).

11.3.49 AF-ABORT (user, commitRI) indication

The TPPM shall be in the READY state.

— Invoke the "Receiving commit order" procedure (see 11.5.8);

— issue a TP-U-ABORT indication with the value of the Rollback parameter set to "false" if no TP-U-

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

ABORT request was received.

If the TPPM is an intermediate,

— invoke the "Sending commit order" procedure (see 11.5.12).

11.3.50 TP-DONE request

If the Heuristic-Report parameter is specified,

— invoke the "Recording the heuristic condition" procedure (see 11.5.9).

If the TPPM is in the DECIDED (commit) state and

a) the *last commit confirm was received*,

— invoke the "Completing commitment" procedure (see 11.5.1);

b) the *last commit confirm was not received*,

— continue.

If the TPPM is in the DECIDED (rollback) state and

a) the *last rollback confirm was received*,

1) if the TPPM is an intermediate or a leaf and the superior *dialogue has not been detached* and the TPPM has not issued a *rollback request* or a *rollback response* to the superior,

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

2) if the *superior dialogue is not chaining* and *rollback reporting has completed*,

— invoke the "Initiating transaction After rollback" procedure (see 11.5.6);

3) if the TPPM is an intermediate and a C-BEGIN indication has been received and the *dialogue has not been detached*,

— invoke the "Initiating transaction After rollback" procedure (see 11.5.6);

4) otherwise,

— continue;

b) the *last rollback confirm was not received*,

— continue.

11.3.51 C-COMMIT confirm or AF-HEURISTIC-REPORT (commitRC) indication

If this is an AF-HEURISTIC-REPORT indication,

— issue a TP-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition" procedure (see 11.5.9).

If either

a) a TP-DEFERRED-END-DIALOGUE request was received and no TP-U-ABORT request was received, or

b) an AF-ABORT (commitRI) request was issued,

then

— issue a SAF-DETACH-ASSOCIATION (free) request.

If a TP-U-ABORT request was received,

a) if a C-COMMIT+C-BEGIN request was issued,

— issue an AF-ABORT (user, rollbackRI) request;

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request;

b) if no C-COMMIT+C-BEGIN request was issued,

— issue an AF-ABORT (user, dataRI) request;

— issue a SAF-DETACH-ASSOCIATION (free) request.

If no TP-U-ABORT request was received and no TP-DEFERRED-END-DIALOGUE request was received,

— issue a C-ROLLBACK request, if a C-COMMIT+C-BEGIN request was issued and a TP-ROLLBACK indication is pending;

— *close the PSAP*.

NOTE — Any further events to/from the subordinate are to be handled either as part of the next transaction branch, or when this transaction branch is completed and the dialogue returns to coordination level of "none". In this way, the events can be handled as part of the normal procedures (e.g. the TPPM is in the ACTIVE state), rather than within the scope of the transaction termination procedures.

If the *last commit confirm was received*,

— invoke the "Completing commitment" procedure (see 11.5.1).

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

11.3.52 AF-ABORT (user, commitRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication

If a C-COMMIT request was issued and the Unchained Transactions functional unit was selected,

— issue a SAF-DETACH-ASSOCIATION (free) request;

— issue a TP-U-ABORT indication with the value of the Rollback parameter set to "false", if no TP-U-ABORT request was received;

— issue a TP-HEURISTIC-REPORT indication if this is an AF-ABORT-AND-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition" procedure (see 11.5.9) if this is an AF-ABORT-AND-HEURISTIC-REPORT indication;

— invoke the "Completing commitment" procedure (see 11.5.1), if the *last commit confirm was received.*

11.3.53 TP-ROLLBACK request

— Invoke the "Initiating rollback at TPPM" procedure (see 11.5.5).

11.3.54 C-ROLLBACK indication or AF-HEURISTIC-REPORT (rollbackRI) indication

If this is an AF-HEURISTIC-REPORT indication and the TPPM is in the ACTIVE state, then an AF-PREPARE request shall have been issued.

If the TPPM is in the

a) ACTIVE or READY state,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

b) ACTIVE state,

— *close the PSAP*, if this is a subordinate dialogue and the Unchained Transactions functional unit is selected on this dialogue;

c) any state, and this is an AF-HEURISTIC-REPORT indication,

— issue a TP-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition " procedure (see 11.5.9);

d) DECIDED (rollback) state

1) if the dialogue is with the subordinate and a TP-U-ABORT request was received,

— issue an AF-ABORT (user, rollbackRC) request;

— issue a SAF-DETACH-ASSOCIATION (free) request;

2) if the dialogue is with the subordinate and a TP-U-ABORT request was not received,

— issue a C-ROLLBACK response;

— *close the PSAP*, if the Unchained Transactions functional unit is selected on this dialogue;

3) if either of the following are true:

i) the dialogue is with the superior and a *rollback request* has been issued to the superior, or

ii) the dialogue is with a subordinate, the *last rollback confirm was received*, this is an intermediate node, and the superior *dialogue has not been detached,*

then

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

4) if the *last rollback confirm was received* and the *superior dialogue is not chaining* and *rollback reporting has completed,*

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

5) if the dialogue is with the superior and the *last rollback confirm was not received,*

— continue.

11.3.55 AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication

If this is an AF-ABORT (provider, rollbackRI) indication, then the Diagnostic parameter shall be set to "begin-transaction-reject", this shall be a subordinate dialogue, the Unchained Transactions functional unit shall have been selected, and a C-BEGIN confirm shall not have been received.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

If the coordination level is "commitment" and the TPPM is in the

a) ACTIVE state and has a *dialogue establishment indication outstanding* and an AF-ABORT (user/provider, rollbackRI) indication is received,

— issue a TP-U-ABORT indication with the Rollback parameter set to "false" if the Type parameter is "user";

— issue a TP-P-ABORT indication with the Rollback parameter set to "false" if the Type parameter is "provider";

— issue a C-ROLLBACK response;

— issue a SAF-DETACH-ASSOCIATION (free) request;

NOTE — The procedure "Initiating rollback at TPPM" (see 11.5.5) is not triggered in this case because the transaction cannot have started at the subordinate's node.

b) ACTIVE or READY state and does not have a *dialogue establishment indication outstanding*,

— issue a TP-U-ABORT indication with the Rollback parameter set to "true" if the Type parameter is "user" or this is an AF-ABORT-AND-HEURISTIC-REPORT indication, and no TP-U-ABORT request was received;

— issue a TP-P-ABORT indication with the Rollback parameter set to "true" if the Type parameter is "provider" and no TP-U-ABORT request was received;

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5);

— issue a SAF-DETACH-ASSOCIATION (free) request if an AF-ABORT (user/provider, rollbackRI) indication was received from a subordinate;

c) DECIDED (rollback) state,

1) if a TP-U-ABORT request was received,

— continue;

2) if no TP-U-ABORT request was received,

— issue a TP-P-ABORT indication with the Rollback parameter set to "false" if the type is "provider";

— issue a TP-U-ABORT indication with the Rollback parameter set to "false" if the type is "user" or this is an AF-ABORT-AND-HEURISTIC-REPORT indication;

3) if the dialogue is with a subordinate,

— issue a C-ROLLBACK response;

— issue a SAF-DETACH-ASSOCIATION (free) request;

d) any state, and this is an AF-ABORT-AND-HEURISTIC-REPORT indication,

— issue a TP-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition" procedure (see 11.5.9);

e) DECIDED (rollback) state,

1) if either of the following are true:

i) the dialogue is with the superior and a *rollback request* has been issued to the superior; or

ii) the dialogue is with a subordinate, the *last rollback confirm was received*, this is an intermediate node, and the superior *dialogue has not been detached*,

then

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

2) if the *last rollback confirm was received* and the *superior dialogue is not chaining* and *rollback reporting has completed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6).

11.3.56 C-ROLLBACK confirm or AF-HEURISTIC-REPORT (rollbackRC) indication

The TPPM shall be in the DECIDED (rollback) state.

If this was an AF-HEURISTIC-REPORT indication,

— issue a TP-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition" procedure (see 11.5.9).

If the *rollback confirm* was from the subordinate,

a) if a TP-U-ABORT request has been received and no AF-ABORT request was issued,

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— issue an AF-ABORT (user, dataRI) request;

b) if the Unchained Transactions functional unit is selected on this dialogue and no TP-U-ABORT request was received,

— *close the PSAP*;

c) if the *last rollback confirm was received* and this is an intermediate node and the superior *dialogue has not been detached*,

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

d) if the *last rollback confirm was received* and the *superior dialogue is not chaining* and *rollback reporting has completed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

e) if a TP-U-ABORT request was received or a TP-U-ABORT indication was issued,

— issue a SAF-DETACH-ASSOCIATION (free) request;

f) if none of the above conditions is satisfied,

— continue.

If the *rollback confirm* was from the superior,

a) if the superior *dialogue is not chaining* and *a TP-DONE request is not owed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

b) if a TP-U-ABORT request was received,

— issue an AF-ABORT (user, dataRI) request if an AF-ABORT (user, rollbackRI) request has not been issued;

— issue a SAF-DETACH-ASSOCIATION (begin-fear) request if no AF-ABORT (user, rollbackRI) request was issued;

— issue a SAF-DETACH-ASSOCIATION (free) request if an AF-ABORT (user, rollbackRI) request was issued;

c) if the superior *dialogue has not been detached*, the Unchained Transactions functional unit is selected, and *a TP-DONE request is owed*,

— *close the PSAP*;

d) if none of the above conditions is satisfied,

— continue.

11.3.57 AF-ABORT (user/provider, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication

The TPPM shall be in the DECIDED (rollback) state and no AF-ABORT (user, rollbackRI) request shall have been issued on this dialogue.

If the coordination level is "commitment" and one of the following is true:

a) this is an AF-ABORT indication on a subordinate dialogue for which the Unchained Transactions functional unit has been selected, the Diagnostic parameter is "Begin-transaction-reject", the Type parameter is set to "provider", and a C-BEGIN confirm has not been received; or

b) this is an AF-ABORT indication and the Type parameter is set to "user"; or

c) this is an AF-HEURISTIC-REPORT-AND-ABORT indication;

then

a) always

— issue a SAF-DETACH-ASSOCIATION (free) request;

b) if this is an AF-ABORT-AND-HEURISTIC-REPORT indication,

— issue a TP-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition" procedure (see 11.5.9);

c) if a TP-U-ABORT request was received,

— continue;

d) if no TP-U-ABORT request was received,

— issue a TP-U-ABORT indication with the Rollback parameter set to "false" if the value of the Type parameter is "user" or this is an AF-ABORT-AND-HEURISTIC-REPORT indication;

— issue a TP-P-ABORT indication with the Rollback parameter set to "false" if the value of the Type parameter is "provider";

e) if the *last rollback confirm was received* and this is

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

an intermediate node and the superior *dialogue has not been detached*,

— invoke the "Reporting rollback to superior" procedure (see 11.5.10);

f) if the indication was received from a subordinate, the *last rollback confirm was received*, and the *superior dialogue is not chaining* and *rollback reporting has completed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6);

g) if the indication was received from the superior and *a TP-DONE request is not owed*,

— invoke the "Initiating transaction after rollback" procedure (see 11.5.6).

11.3.58 CAF-RECOVER (ready) indication

If the TPPM is in the

a) READY state, and

1) if no other channel for this branch identifier already exists, by a local decision, either

— issue a C-RECOVER (retry-later) response;

— issue a CAF-DETACH (free) request;

or

— continue;

NOTE 1 The C-RECOVER (retry-later) response may be issued to release the channel while waiting for the *commit indication* from the superior.

2) if another channel for this branch identifier already exists, by a local decision, do either or both of

— issue a C-RECOVER (retry-later) response on the previously existing channel; and

— issue a CAF-DETACH (free) request on the previously existing channel;

or

— issue a C-RECOVER (retry-later) response on the channel from which this indication was received; and

— issue a CAF-DETACH (free) request on the channel from which this indication was received;

NOTE 2 This situation will arise when one of the channels has been aborted, but the abort has not yet been signaled to the TPPM. This part of ISO/IEC 10026 does not provide enough information to determine which channel has been aborted, so it is left to an implementation to make the choice of the channel to be released since the implementation may have local information to help determine which channel has been aborted. In addition, a local decision may be made to issue C-RECOVER (retry-later) responses and CAF-DETACH (free) requests on both channels while waiting for the *commit indication* from the superior.

b) DECIDED (commit) state

1) if no channel for this branch identifier already exists,

— issue a C-RECOVER (commit) request;

2) if a channel for this branch identifier already exists,

— issue a C-RECOVER (retry-later) response on the channel from which this indication was received;

— issue a CAF-DETACH (free) request on the channel from which this indication was received;

c) DECIDED (rollback) state

— issue a C-RECOVER (unknown) response;

— issue a CAF-DETACH (free) request.

If the *dialogue* corresponding to the value contained in the Branch Identifier parameter *has not been detached*, and the TPPM is in the

a) ACTIVE state and an AF-PREPARE request was sent,

— issue a TP-P-ABORT indication with the

1) Diagnostic parameter set to "permanent-failure"; and
2) Rollback parameter set to "true";

— issue an AF-ABORT (provider, abortRI) request on the dialogue, with the Diagnostic parameter set to "permanent-failure";

— issue a C-RECOVER (unknown) response on the channel;

— issue a CAF-DETACH (free) request on the channel;

— invoke the "Initiating rollback at TPPM"

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

procedure (see 11.5.5);

— issue a TP-HEURISTIC-REPORT indication with the Heuristic-Report parameter set to "heuristic-hazard";

— invoke the "Recording the heuristic condition" procedure (see 11.5.9);

b) READY, DECIDED (commit), or DECIDED (rollback) state,

— issue a TP-P-ABORT indication if no TP-U-ABORT request was received, with the

1) Diagnostic parameter set to "permanent-failure"; and
2) Rollback parameter set to "false";

— issue an AF-ABORT (provider, abortRI) request on the dialogue;

c) DECIDED (rollback) state,

— issue a TP-HEURISTIC-REPORT indication with the Heuristic-Report parameter set to "heuristic-hazard";

— invoke the "Recording the heuristic condition" procedure (see 11.5.9);

d) DECIDED (commit) state,

— invoke the "Rollback next transaction" procedure (see 11.5.11) if a C-COMMIT+C-BEGIN request was issued.

11.3.59 C-RECOVER (ready) indication or AF-RECOVER (ready) indication (CPM)

If this is a two-way-recovery channel and an AF-TOKEN-PLEASE request was previously issued since the most recent CAF-PLEASE request was received,

— issue a CAF-FAIL indication to the TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in the previous CAF-PLEASE request.

If this is an AF-RECOVER (ready) indication, according to a local decision, either

— issue a C-RECOVER (retry-later) response; or

NOTE — A C-RECOVER (retry-later) response may be issued if it is currently impossible to determine whether the log record exists (e.g., the portion of the set of log records identified by the recovery-context-handle [see 7.6] is currently inaccessible).

— attempt to locate a TPPM with an atomic action identifier, branch identifier, and recovery-context-handle corresponding to the parameters of the AF-RECOVER (ready) indication.

If this is a C-RECOVER (ready) indication,

— attempt to locate a TPPM with an atomic action identifier and branch identifier, corresponding to the parameters of the C-RECOVER (ready) indication.

If a TPPM is found,

— issue a CAF-RECOVER (ready) indication to the TPPM that has been found. Set the corresponding parameters of the CAF-RECOVER (ready) indication from the C-RECOVER (ready) indication or AF-RECOVER (ready) indication and set the Channel-Utilization parameter to that of the most recent AF-BEGIN-DIALOGUE request or indication on this channel.

If no TPPM is found and a C-RECOVER (retry-later) response was not issued,

— issue a C-RECOVER (unknown) response.

11.3.60 CAF-RECOVER (commit) indication

If the TPPM is in the

a) READY state,

— invoke the "Receiving commit order" procedure (see 11.5.8);

— invoke the "Sending commit order" procedure (see 11.5.12) if this node is an intermediate;

b) READY state, or DECIDED(commit) state and the *last commit confirm was not received*,

1) if no channel for this branch identifier already exists and, by a local decision, either

— issue a C-RECOVER (retry-later) response; and

— issue a CAF-DETACH (free) request;

or

— continue;

NOTE 1 The C-RECOVER (retry-later) response may be issued to release the channel while waiting for the *commit confirms* from the subordinates.

2) if a channel for this branch identifier already

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

exists and the TPPM is in the READY state, by a local decision, either

— issue a CAF-DETACH (Clean-up) request on the previously existing channel;

— issue a C-RECOVER (retry-later) response on the channel from which this indication was received; and

— issue a CAF-DETACH (free) request on the channel from which this indication was received;

or

— issue a CAF-DETACH (Clean-up) request on the previously existing channel;

NOTE 2 The C-RECOVER (retry-later) response may be issued to release the channel while waiting for the *commit confirms* from the subordinates. In any case, the previously existing channel is *detached* because the TPPM now knows the outcome of the transaction is commit, so the reply from the previous channel is uninteresting.

3) if a channel for this branch identifier already exists and the TPPM is in the DECIDED (commit) state, by a local decision, do either or both of

— issue a C-RECOVER (retry-later) response on the previously existing channel; and

— issue a CAF-DETACH (free) request on the previously existing channel

or

— issue a C-RECOVER (retry-later) response on the channel from which this indication was received; and

— issue a CAF-DETACH (free) request on the channel from which this indication was received.

NOTE 3 This situation will arise when one of the channels has been aborted, but the abort has not yet been signaled to the TPPM. This part of ISO/IEC 10026 does not provide enough information to determine which channel has been aborted, so it is left to an implementation to make the choice of the channel to be released since the implementation may have local information to help determine which channel has been aborted. In addition, a local decision may be made to *detach* both channels while waiting for the *commit confirms* from the subordinates.

If the *dialogue* corresponding to the value contained in the Branch Identifier parameter *has not been detached,*

— issue a TP-P-ABORT indication if no TP-U-ABORT request was issued, with the

1) Diagnostic parameter set to "permanent-failure";

and

2) Rollback parameter set to "false";

— issue an AF-ABORT (provider, abortRI) request on the dialogue, with the Diagnostic parameter set to "permanent-failure";

— invoke the "Rollback next transaction" procedure (see 11.5.11) if a C-COMMIT+C-BEGIN request was received on the dialogue.

11.3.61 C-RECOVER (commit) indication

The TPPM shall be in the READY state.

— Invoke the "Receiving commit order" procedure (see 11.5.8);

— invoke the "Sending commit order" procedure (see 11.5.12) if this node is an intermediate.

By a local decision, optionally

— issue a C-RECOVER (retry-later) response; and

— issue a CAF-DETACH (free) request.

NOTE — The C-RECOVER (retry-later) response may be issued to release the channel while waiting for the *commit confirms* from the subordinates.

11.3.62 C-RECOVER (commit) indication or AF-RECOVER (commit) indication (CPM)

If this is a two-way-recovery channel and an AF-TOKEN-PLEASE request was previously issued since the most recent CAF-PLEASE request was received,

— issue a CAF-FAIL indication to the TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in the previous CAF-PLEASE request.

If the most recent request from the TPPM on this channel was a CAF-DETACH (Clean-up) request and this is a C-RECOVER (commit) indication,

— issue a C-RECOVER (retry-later) response;

If the most recent request from the TPPM on this channel was not a CAF-DETACH (Clean-up) request

a) if this is an AF-RECOVER (commit) indication, according to a local decision, either

— issue a C-RECOVER (retry-later) response;

NOTE — A C-RECOVER (retry-later) response may be issued if it is currently impossible to determine whether the log record exists (e.g., the portion of the set of log records

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

identified by the recovery-context-handle [see 7.5] is currently inaccessible).

or

— attempt to locate a TPPM with an atomic action identifier, branch identifier, and recovery-context-handle corresponding to the parameters of the AF-RECOVER (commit) indication;

b) if this is a C-RECOVER (commit) indication,

— attempt to locate a TPPM with an atomic action identifier and branch identifier, corresponding to the parameters of the C-RECOVER (commit) indication;

c) if a TPPM is found,

— issue a CAF-RECOVER (commit) indication to the TPPM that has been found. Set the corresponding parameters of the CAF-RECOVER (commit) indication from the C-RECOVER (commit) indication or AF-RECOVER (commit) indication and set the channel-utilization parameter to that of the most recent AF-BEGIN-DIALOGUE request or indication on this channel;

d) if no TPPM is found and a C-RECOVER (retry-later) response was not issued,

— issue a C-RECOVER (done) response if no log-damage record exists;

— issue an AF-HEURISTIC-REPORT (recoverDoneRC) request if a log-damage record exists; in this case, the Heuristic-Report parameter is set to the current value of the log-damage record.

11.3.63 C-RECOVER (done) confirm or AF-HEURISTIC-REPORT (recoverDoneRC) indication

If this is an AF-HEURISTIC-REPORT indication,

— issue a TP-HEURISTIC-REPORT indication;

— invoke the "Recording the heuristic condition" procedure (see 11.5.9).

For either service primitive

— issue a CAF-DETACH (free) request;

— invoke the "Completing commitment" procedure (see 11.5.1) if the *last commit confirm was received.*

11.3.64 C-RECOVER (unknown) confirm

— Issue a CAF-DETACH (free) request;

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5) if the TPPM is in the READY state.

11.3.65 C-RECOVER (unknown) confirm (CPM)

If the most recent request from the TPPM on this channel was a CAF-DETACH (Clean-up) request,

— continue.

11.3.66 C-RECOVER (retry-later) confirm

— Issue a CAF-DETACH (free) request.

NOTE — The requirement to perform recovery is not absolved. The internal event "Retry recovery" (see 11.4.4) is used to initiate recovery, or recovery could be initiated by the partner, or both.

11.3.67 C-RECOVER (retry-later) confirm (CPM)

If the most recent request from the TPPM on this channel was a CAF-DETACH (clean-up) request,

— continue.

11.3.68 AF-TOKEN-GIVE (two-way-recovery) indication on a channel (TPPM)

If this is a two-way-recovery channel and the last service primitive on that channel was a C-RECOVER (commit) request,

— issue an AF-TOKEN-GIVE (two-way-recovery) request.

If this is a two-way-recovery channel and the last service primitive on that channel was a C-RECOVER indication or AF-RECOVER indication,

— continue.

11.3.69 AF-TOKEN-GIVE indication (CPM)

If there is an outstanding CAF-PLEASE request matching a channel owned by the CPM to the appropriate AEI, and two-way-recovery has been selected on the channel,

— issue a CAF-GIVE indication to the TPPM with a branch corresponding to the Atomic Action Identifier and Branch Identifier specified in the previous CAF-PLEASE request. Set the Channel-Utilization parameter to "two-way-recovery";

otherwise, if this is a two-way-recovery channel,

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— continue.

11.3.70 AF-TOKEN-PLEASE indication on a channel (TPPM)

If this is a two-way-recovery channel,

— continue.

NOTE — The receipt of an AF-TOKEN-PLEASE indication by the TPPM will always be the result of a collision of an AF-TOKEN-PLEASE request issued by a CPM and a C-RECOVER request or AF-RECOVER request issued by the TPPM.

11.3.71 AF-TOKEN-PLEASE indication (CPM)

If this is a two-way-recovery channel,

— issue an AF-TOKEN-GIVE (two-way-recovery) request.

11.3.72 CAF-PLEASE request (CPM)

If there is already an established channel owned by the CPM to the appropriate AEI, which is a two-way-recovery channel or a one-way-recovery channel initiated by the CPM, and

a) if this is a

1) one-way-recovery channel;

2) two-way-recovery channel and the *token* is owned; or

3) two-way-recovery channel and the last request received on the channel was a CAF-DETACH (not-used) request,

then

— *transfer the channel* to the requesting TPPM;

— issue a CAF-GIVE indication to the requesting TPPM;

b) if this is a two-way-recovery channel, the *token* is not owned, and the last request received on the channel was not a CAF-DETACH (not-used) request,

— issue an AF-TOKEN-PLEASE request.

Otherwise,

— assign an association compatible with the AEI requirements for a channel and to the AE-Title as specified on the CAF-PLEASE request;

— issue an AF-BEGIN-DIALOGUE request with the value of the

a) Functional-Units-Selected parameter set to "Recovery";

b) Channel-Utilization parameter set to either "one-way-recovery" or "two-way-recovery" based on a local decision.

11.3.73 CAF-GIVE indication

If the TPPM is in the

a) READY state, the recovery is to the superior, and no channel is *attached*,

— issue a C-RECOVER (ready) request if no recovery-context-handle is present in the log-ready record;

— issue an AF-RECOVER (ready) request if a recovery-context-handle is present in the log-ready record;

— issue an AF-TOKEN-GIVE (two-way-recovery) request if this is a two-way-recovery channel;

b) DECIDED (commit) state and the recovery is to the superior,

— issue a CAF-DETACH (not-used) request;

c) DECIDED (commit) state, the recovery is to a subordinate, no channel is *attached*, and a C-RECOVER (done) confirm was not received,

— issue a C-RECOVER (commit) request if no recovery-context-handle is present in the log-commit record;

— issue an AF-RECOVER (commit) request if a recovery-context-handle is present in the log-commit record;

— issue an AF-TOKEN-GIVE (two-way-recovery) request if this is a two-way-recovery channel;

d) DECIDED (commit) state, the recovery is to a subordinate, and either a channel is *attached* or a C-RECOVER (done) was received,

— issue a CAF-DETACH (not-used) request.

NOTE — Between the time the TPPM requested the channel for recovering a particular branch, an incoming recovery indication for that branch might have occurred, removing the need for a channel.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

11.3.74 CAF-FAIL indication

If the TPPM is in the

a) READY state, the recovery is to the superior, and no channel is *attached*,

— issue a CAF-PLEASE request with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the superior, contained in the log-ready record;

b) DECIDED (commit) state and the indication is from the superior,

— continue;

c) DECIDED (commit) state, the recovery is to a subordinate, no channel is *attached*, and a C-RECOVER (done) confirm was not received,

— issue a CAF-PLEASE request, with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the superior, contained in the log-ready record;

d) DECIDED (commit) state, the recovery is to a subordinate, and either a channel is *attached* or a C-RECOVER (done) confirm was received,

— continue.

11.3.75 CAF-DETACH request (CPM)

— *transfer the channel* to the CPM.

11.4 Internal event procedures

These procedures are invoked upon the receipt of an event local to a PM. The first paragraph of each procedure describes the conditions under which the procedure is invoked.

11.4.1 Delay recovery

This procedure is invoked when the TPPM will release the channel on which an AF-RECOVER indication or a C-RECOVER indication but no corresponding AF-RECOVER response or C-RECOVER response was issued.

— Issue a C-RECOVER (retry-later) response;

— issue a CAF-DETACH (free) request.

NOTES

1 In the absence of a TPPM state change, it is the partner's responsibility to initiate recovery. In the situation where the TPPM state changes from READY state to DECIDED (commit) state, the TPPM will initiate recovery.

2 This is used when it is undesirable for the TPPM to hold the channel while waiting for the conditions necessary to issue the C-RECOVER response.

11.4.2 Heuristic damage compensation for subtree

This procedure is invoked when it is known that heuristic damage compensation has occurred for the subtree. It causes the log-damage and log-heuristic records to be deleted since the damage was corrected. This prevents the reporting of this damage to the superior.

— Remove the log-damage record from secure storage.

If a log-heuristic record exists,

— remove the log-heuristic record from secure storage.

11.4.3 Restart after node crash (CPM)

The PM procedures assume that this procedure is entered before any other procedure after a node crash.

The MACF of a TPPM which cannot continue normal operation is deleted. The SAOs which were part of this TPPM have disappeared by an association abort or they are in a state such that arriving APDUs cannot interfere with the operation of the TPPM re-established after the node crash.

The CPM instantiates a TPPM for each log-ready record or log-commit record found, provided a TPPM within the AEI does not already exist for the log record, as follows:

NOTE — For each TPPM, the internal event "TPPM creation after node crash" (see 11.4.7) will occur.

a) if a log-commit record was found,

— create the TPPM in the DECIDED (commit) state;

b) if a log-ready record was found,

— create the TPPM in the READY state.

For all *TPPM bound data* for which no log record is found,

— set the *TPPM bound data* to the initial state.

11.4.4 Retry recovery

This procedure may be invoked at any time while the TPPM's responsibility to perform recovery is not yet absolved, and the TPPM has not issued a CAF-PLEASE

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

request nor is it processing recovery of the branch.

NOTE — A timer is a typical mechanism used to invoke this procedure.

— Issue a CAF-PLEASE request with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the superior, contained in the log-ready record, if this is a branch to the superior, or set to the value of the AE-Title taken from the branch identifier for the subordinate, contained in the log-commit record, if this is a branch to a subordinate.

11.4.5 Taking a heuristic decision

This procedure is entered when the TPPM takes a heuristic decision.

A TPPM may take a heuristic decision only while in the ACTIVE state if an AF-PREPARE indication has been received, or while in the READY state.

— Part or all of the bound data is set to the initial, or final state.

— secure the heuristic decision by *writing the log-heuristic record.*

11.4.6 Terminating a channel (CPM)

This procedure is entered when the CPM terminates a channel according to a local decision.

If the *token* is owned and the channel is *attached* to the CPM,

— issue an AF-END-DIALOGUE request with the Confirmation parameter set to "false";

— issue a SAF-DETACH-ASSOCIATION (free) request.

11.4.7 TPPM creation after node crash

This procedure is invoked as the first one in a newly created TPPM after a node crash. This procedure is executed as an integral part of the creation of the TPPM.

NOTE — This internal event procedure is entered immediately after completion of the internal event procedure "Restart after node crash (CPM)", if a TPPM was created.

— Issue a TP-COMMIT indication if this is a root TPPM in the DECIDED (commit) state;

— issue a CAF-PLEASE request

a) for each subordinate in the log-commit record, if the TPPM is in the DECIDED (commit) state; the AE-title parameter of the CAF-PLEASE request is the AE-title of the subordinate in the log-commit record;

b) for the superior in the log-ready record, if the TPPM is in the READY state; the AE-title parameter of the CAF-PLEASE request is the AE-title of the superior's branch identifier in the log-ready record.

11.4.8 TPPM-initiated rollback

This procedure may be invoked without any preconditions in ACTIVE state except when a dialogue establishment indication is outstanding.

— Invoke the "Initiating rollback at TPPM" procedure (see 11.5.5).

11.5 Common Procedures

These procedures are referenced by the main or internal event procedures.

Each procedure begins with a table that indicates the procedures that invoke the given procedure.

11.5.1 Completing commitment

Invoking Procedure Name	Clause
AF-ABORT (user, commitRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication	11.3.52
TP-DONE request	11.3.50
C-COMMIT confirm or AF-HEURISTIC-REPORT (commitRC) indication	11.3.51
C-RECOVER (done) confirm or AF-HEURISTIC-REPORT (recoverDoneRC) indication	11.3.63

— *Forget the transaction*;

NOTE 1 Before the transaction is forgotten, it is assumed that either the *TPPM bound data* are in the final state, or provisions have been made by the local system to achieve setting the *TPPM bound data* to the final state at a later time.

— issue a TP-COMMIT-COMPLETE indication.

Additionally,

a) if no TP-ROLLBACK indication is pending and

1) any *dialogue is chaining*,

— enter the ACTIVE state;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

2) no dialogue is *chaining*,

— cease to be part of the transaction;

3) the TPPM is an intermediate and the *dialogue* with the superior *is not chaining*,

— become a root node;

4) the TPPM is an intermediate and there are no *chaining subordinate dialogues*,

— become a leaf node;

b) if a TP-ROLLBACK indication is pending,

— issue a TP-ROLLBACK indication;

— make the TP-ROLLBACK indication no longer pending;

— enter the DECIDED (rollback) state;

c) for each subordinate dialogue that has *closed the PSAP*,

— *open the PSAP*;

If the TPPM is an intermediate or leaf node, and

a) if the superior *dialogue has not been detached*, take the first applicable actions of the following in the scope of the dialogue with the superior:

1) if a TP-U-ABORT request has been received, no AF-ABORT indication was received, the Unchained Transactions functional unit is selected, and a log-damage record exists,

NOTE 2 An AF-ABORT (user) request cannot have been previously issued (in response to a TP-U-ABORT request after the ACTIVE state) because there was no opportunity to issue a request to the superior after the TPPM entered the READY state.

— issue an AF-ABORT-AND-HEURISTIC-REPORT (commitRC) request. When issued, the AF-ABORT-AND-HEURISTIC-REPORT request carries the Heuristic-Report parameter set to the current value of the log-damage record;

— issue a SAF-DETACH-ASSOCIATION (free) request;

2) if a TP-U-ABORT request has been received, no AF-ABORT indication was received, the Unchained Transactions functional unit is selected, and no log-damage record exists,

NOTE 3 An AF-ABORT (user) request cannot have been previously issued (in response to a TP-U-ABORT request after the ACTIVE state) because there was no opportunity to issue a request to the superior after the TPPM entered the READY state.

— issue an AF-ABORT (user, commitRC) request;

— issue a SAF-DETACH-ASSOCIATION (free) request;

NOTE 4 The case of TP-U-ABORT request to the superior with the Chained Transactions functional unit selected is handled as part of the rollback procedures since the only way a TP-U-ABORT request could be issued to the superior is if there was a failure causing a rollback of the next transaction. In this way, the AF-ABORT indication to the superior will be carried by the appropriate CCR service.

3) if a log-damage record exists,

— issue an AF-HEURISTIC-REPORT (commitRC) request with the Heuristic-Report parameter set to the current value of the log-damage record;

— issue a SAF-DETACH-ASSOCIATION (free) request if an AF-DEFER (end-dialogue) indication was received;

4) if none of the above conditions was met,

— issue a C-COMMIT response;

— issue a SAF-DETACH-ASSOCIATION (free) request if an AF-DEFER (end-dialogue) indication or AF-ABORT (user, commitRI) indication was received;

b) if the superior *dialogue has been detached*, and

1) a channel with the superior is *attached*,

— issue a C-RECOVER (done) response if no log-damage record exists;

— issue an AF-HEURISTIC-REPORT (recoverDoneRC) request, if a log-damage record exists. When issued, the AF-HEURISTIC-REPORT request carries the Heuristic-Report parameter set to the current value of the log-damage record;

— issue a CAF-DETACH (free) request;

2) no channel with the superior is *attached*,

— continue.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

NOTE 5 In this case, when the superior subsequently establishes a channel and the CPM receives the C-RECOVER (commit) or AF-RECOVER (commit) indication, it will issue a C-RECOVER (done) response.

11.5.2 Entering READY state

Invoking Procedure Name	Clause
TP-COMMIT request	11.3.45
C-READY indication	11.3.47

If the TPPM is able to set the *TPPM bound data* in the ready-to-commit state,

— *write a log-ready record* atomically setting the *TPPM bound data* to the ready-to-commit state, if they are not already set to the ready-to-commit state;

— *write a log-ready record* if the *TPPM bound data* are already set to the ready-to-commit state;

NOTE 1 An implementation may choose to set the *TPPM bound data* to the ready-to-commit state atomically with writing the log-ready record.

— enter the READY state;

— invoke the "First request/response" procedure (see 11.5.3), if an AF-BEGIN-DIALOGUE response has not been issued;

— issue a P-TOKEN-GIVE (sync-minor) request, if the *token* is owned by the node on the dialogue with the superior;

NOTE 2 If the *token* was moved by the User-ASE to the subordinate, issuance of a P-TOKEN-GIVE (sync-minor) request is necessary to allow the superior to issue a *commit request*, if required.

— issue a C-READY request on the dialogue with the superior;

otherwise,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5).

11.5.3 First request/response

Invoking Procedure Name	Clause
TP-BEGIN-DIALOGUE response	11.3.3
TP-END-DIALOGUE request	11.3.11
TP-END-DIALOGUE response	11.3.14
TP-U-ERROR request	11.3.16
TP-U-ABORT request	11.3.19
TP-GRANT-CONTROL request	11.3.24
TP-REQUEST-CONTROL request	11.3.26
TP-HANDSHAKE request	11.3.28
TP-HANDSHAKE response	11.3.30
TP-HANDSHAKE-AND-GRANT-CONTROL request	11.3.32
TP-HANDSHAKE-AND-GRANT-CONTROL response	11.3.34
TP-DATA request	11.3.39
Entering READY state	11.5.2

NOTE — This procedure is not used for the DECIDED (rollback) state.

— Issue an AF-BEGIN-DIALOGUE (accepted, dataRI) response;

— issue a C-BEGIN response if a C-BEGIN indication was received;

— issue as many AF-U-ERROR responses as the number of AF-U-ERROR indications that have been received, if the Shared Control functional unit is selected.

11.5.4 Initiating a transaction branch

Invoking Procedure Name	Clause
TP-BEGIN-DIALOGUE request	11.3.1
TP-BEGIN-TRANSACTION request	11.3.36

— Issue a C-BEGIN request with the value of the

a) Atomic Action Identifier parameter set to

1) a newly created value, if the TPPM is not in the ACTIVE state;

2) the value of the atomic action identifier of the most recent C-BEGIN indication, if this is an intermediate or leaf node in the ACTIVE state; or

3) the value of the atomic action identifier of the transaction, if this is a root node in the ACTIVE state;

b) Branch Identifier parameter set to a value that uniquely identifies the transaction branch within the scope of the atomic action identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

If the TPPM is a leaf node in the ACTIVE state,

— become an intermediate node.

If the TPPM is not in the ACTIVE state,

— become a root node;

— enter the ACTIVE state.

11.5.5 Initiating rollback at TPPM

Invoking Procedure Name	Clause
AF-BEGIN-DIALOGUE (rejected, dataRI) confirm on a Dialogue	11.3.5
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm	11.3.6
SAF-ASSOCIATION-LOST indication	11.3.9
AF-END-DIALOGUE indication	11.3.12
AF-U-ERROR indication	11.3.17
TP-U-ABORT request	11.3.19
AF-ABORT (user, dataRI) indication	11.3.20
Protocol Error, Internal Error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result= affirmative) response, or A-RELEASE (Result=affirmative) confirm on a Dialogue	11.3.21
AF-HANDSHAKE indication	11.3.29
U-ASE indication	11.3.40
AF-PREPARE indication	11.3.46
TP-ROLLBACK request	11.3.53
C-ROLLBACK indication or AF-HEURISTIC-REPORT (rollbackRI) indication	11.3.54
AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication	11.3.55
CAF-RECOVER (ready) indication	11.3.58
C-RECOVER (unknown) confirm	11.3.64
TPPM-initiated rollback	11.4.8
Making commitment decision	11.5.7
Entering READY state	11.5.2

NOTE 1 A *rollback request* or *rollback response* is not sent to the superior until a TP-DONE request has been issued by the TPSUI and a rollback indication or confirm has been received from each subordinate. This allows for the propagation of any heuristic condition information to the root of the transaction tree. Sending the *rollback request* or *rollback response* to the superior is done in the "Reporting rollback to superior" procedure (see 11.5.10).

If the TPPM is in the ACTIVE state

— enter the DECIDED (rollback) state;

— issue a TP-ROLLBACK indication, if no rollback initiating request has been received nor rollback initiating indication has been issued (by the calling procedure);

NOTE 2 The terms "rollback initiating request" and "rollback initiating indication" are defined in the TP Service, ISO/IEC 10026-2.

— set the *TPPM bound data*, if any, to the initial state;

— issue a C-ROLLBACK response if this is a *rollback indication* from a subordinate;

— issue a C-ROLLBACK request to each subordinate except the one, if any, to which a *rollback request* was issued or from which a *rollback indication* has been received or whose *dialogue has been detached.*

NOTE 3 The case of the *rollback request* being issued would occur only when the rollback is issued in the same action sequence by the calling procedure.

If the TPPM is in the READY state, and the *rollback indication* was from the superior,

— enter the DECIDED (rollback) state;

— issue a TP-ROLLBACK indication, if no rollback initiating indication has been issued (by the calling procedure);

NOTE 4 The term "rollback initiating indication" is defined in the TP Service, ISO/IEC 10026-2.

— set the *TPPM bound data*, if any, to the initial state, unless a heuristic decision has been taken;

— issue a C-ROLLBACK request to each subordinate for which no TP-U-ABORT request was received and whose *dialogue has not been detached*;

— issue an AF-ABORT (user, rollbackRI) request to each subordinate for which a TP-U-ABORT request was received and whose *dialogue has not been detached*;

— issue a C-RECOVER (unknown) response to each subordinate from which a CAF-RECOVER (ready) indication was received and to which no C-RECOVER (retry-later) response was issued;

— issue a CAF-DETACH (free) request to each subordinate from which a CAF-RECOVER (ready) indication was received and to which no C-RECOVER (retry-later) response was issued and whose channel is still *attached*;

— *write a log-damage record* with the value "heuristic-hazard" if one does not exist and a subordinate *dialogue has been detached*;

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

— *write a log-damage record* with a value of "heuristic-mix" if the bound data are set to a state other than the initial state as a result of a previously taken heuristic decision by the TPPM;

— *forget the transaction.*

11.5.6 Initiating transaction after rollback

Invoking Procedure Name	Clause
AF-BEGIN-DIALOGUE (accepted) confirm on a Dialogue	11.3.4
AF-BEGIN-DIALOGUE (rejected, dataRI) confirm on a Dialogue	11.3.5
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm	11.3.6
AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) confirm	11.3.7
SAF-ASSOCIATION-LOST indication	11.3.9
AF-END-DIALOGUE indication	11.3.12
AF-ABORT (user, dataRI) indication	11.3.20
AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication	11.3.55
AF-ABORT (user, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication	11.3.57
Protocol Error, Internal Error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result= affirmative) response, or A-RELEASE (Result=affirmative) confirm on a Dialogue	11.3.21
C-BEGIN indication	11.3.37
TP-DONE request	11.3.50
C-ROLLBACK indication or AF-HEURISTIC-REPORT (rollbackRI) indication	11.3.54
C-ROLLBACK confirm or AF-HEURISTIC-REPORT (rollbackRC) indication	11.3.56

— Issue a TP-ROLLBACK-COMPLETE indication;

— *open the PSAP* for all subordinate dialogues for which the Unchained Transactions functional unit is selected;

If the *dialogue* with the superior *is not chaining*,

a) if the dialogue with the superior *has not been detached*,

— *open the PSAP*, if it is closed;

b) if any subordinate *dialogue is chaining*,

— become a root node;

— issue a C-BEGIN request to each *chaining subordinate dialogue* with the value of the

1) Atomic Action Identifier parameter set to a newly created value;

2) Branch Identifier parameter set to a value that uniquely identifies the transaction branch within the scope of the atomic action identifier;

— enter the ACTIVE state;

c) if no subordinate *dialogue is chaining*

— cease to be part of the transaction.

If the superior *dialogue* is *chaining*,

a) always

— *open the PSAP* with the superior, if closed;

b) if any subordinate *dialogue is chaining*

— issue a C-BEGIN request to each *chaining subordinate dialogue* with the value of the

1) Atomic Action Identifier parameter set to the value of the atomic action identifier received on the most recent C-BEGIN indication;

2) Branch Identifier parameter set to a value that uniquely identifies the transaction branch within the scope of the atomic action identifier;

— enter the ACTIVE state;

c) if no subordinate *dialogue is chaining*

— become a leaf node;

— enter the ACTIVE state.

11.5.7 Making commitment decision

Invoking Procedure Name	Clause
TP-COMMIT request	11.3.45
C-READY indication	11.3.47

If the TPPM is able to set the *TPPM bound data* in the final state,

— *write a log-commit record*;

NOTE — If the log-commit record cannot be written, a rollback is triggered. This implies that the action of

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

writing a log-commit record in the above paragraph may fail. That is why there is a check for the success or failure of writing the log-commit record in the following paragraphs.

— begin setting the *TPPM bound data* to the final state, unless a heuristic decision has been taken. The *TPPM bound data* shall eventually be set to the final state; when this occurs is a local matter.

If the TPPM is unable to set the *TPPM bound data* in the final state or *write the log-commit record*,

— invoke the "Initiating rollback at TPPM" procedure (see 11.5.5).

If the log-commit record was written,

— issue a TP-COMMIT indication;

— enter the DECIDED (commit) state;

— invoke the "Sending commit order" procedure (see 11.5.12).

11.5.8 Receiving commit order

Invoking Procedure Name	Clause
AF-ABORT (user, commitRI) indication	11.3.49
C-COMMIT indication or C-COMMIT+C-BEGIN indication	11.3.48
CAF-RECOVER (commit) indication	11.3.60
C-RECOVER (commit) indication	11.3.61

— Issue a TP-COMMIT indication;

— begin setting the *TPPM bound data* to the final state, unless a heuristic decision has been taken. The *TPPM bound data* shall eventually be set to the final state; when this occurs is a local matter;

— *write a log-damage record* with the value "Heuristic-mix", if the node took a heuristic decision to set its *TPPM bound data* in a state other than the final state;

NOTE — the TPPM of an intermediate or leaf node may or may not *write the log-commit record* at this time. Upon node crash recovery, ("Restart after node crash", see 11.4.3) if the log-commit record has not been written, the node will be in the READY state and will take the appropriate recovery actions.

— enter the DECIDED (commit) state.

11.5.9 Recording the heuristic condition

Invoking Procedure Name	Clause
AF-ABORT (user, commitRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication	11.3.52
AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication	11.3.55
AF-ABORT (user, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication	11.3.57
TP-DONE request	11.3.50
C-COMMIT confirm or AF-HEURISTIC-REPORT (commitRC) indication	11.3.51
C-ROLLBACK indication or AF-HEURISTIC-REPORT (rollbackRI) indication	11.3.54
C-ROLLBACK confirm or AF-HEURISTIC-REPORT (rollbackRC) indication	11.3.56
CAF-RECOVER (ready) indication	11.3.58
C-RECOVER (done) confirm or AF-HEURISTIC-REPORT (recoverDoneRC) indication	11.3.63

If the Heuristic-report parameter is set to "heuristic-hazard",

— *write a log-damage record* with the value "heuristic-hazard" if it has not already been created.

If the Heuristic-report parameter is set to "heuristic-mix",

— *write log-damage record* with the value "heuristic-mix" if it has not already been created, or update it to "heuristic-mix" if its current value is "heuristic-hazard".

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

11.5.10 Reporting rollback to superior

Invoking Procedure Name	Clause
AF-BEGIN-DIALOGUE (accepted) confirm on a Dialogue	11.3.4
AF-BEGIN-DIALOGUE (rejected, dataRI) confirm on a Dialogue	11.3.5
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm	11.3.6
AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) confirm	11.3.7
SAF-ASSOCIATION-LOST indication	11.3.9
AF-END-DIALOGUE indication	11.3.12
AF-ABORT (user, dataRI) indication	11.3.20
AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication	11.3.55
AF-ABORT (user/provider, rollbackRC) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication	11.3.57
Protocol Error, Internal Error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result= affirmative) response, or A-RELEASE (Result=affirmative) confirm on a Dialogue	11.3.21
TP-DONE request	11.3.50
C-ROLLBACK indication or AF-HEURISTIC-REPORT (rollbackRI) indication	11.3.54
C-ROLLBACK confirm or AF-HEURISTIC-REPORT (rollbackRC) indication	11.3.56

If a log-ready record exists,

— *forget the transaction.*

If a *rollback indication* was received from the superior,

a) take the first applicable set of actions on the dialogue to the superior:

1) if a TP-U-ABORT request has been received for the superior and no AF-ABORT indication was received from the superior and a log-damage record exists,

— issue an AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) request. When issued, the AF-ABORT-AND-HEURISTIC-REPORT request carries the Heuristic-Report parameter set to the current value of the log-damage record;

— issue a SAF-DETACH-ASSOCIATION (free) request;

2) if a TP-U-ABORT request has been received for the superior and no AF-ABORT indication was received from the superior,

— issue an AF-ABORT (user, rollbackRC) request;

— issue a SAF-DETACH-ASSOCIATION (free) request;

3) if a log-damage record exists,

— issue an AF-HEURISTIC-REPORT (rollbackRC) request. When issued, the AF-HEURISTIC-REPORT request carries the Heuristic-Report parameter set to the current value of the log-damage record;

— issue a SAF-DETACH-ASSOCIATION (free) request if an AF-ABORT indication was received;

4) if no AF-ABORT indication was received from the superior and the Confirmation parameter of the AF-BEGIN-DIALOGUE indication was set to "always" and no AF-BEGIN-DIALOGUE response has been issued,

NOTE — The service definition "dialogue establishment indication outstanding" does not apply here.

— issue an AF-BEGIN-DIALOGUE (accepted, rollbackRC) response;

5) if none of the above conditions could be met,

— issue a C-ROLLBACK response;

— issue a SAF-DETACH-ASSOCIATION (free) request if an AF-ABORT indication was received;

b) if the Unchained Transactions functional unit is selected on this dialogue and *a TP-DONE request is owed*,

— *close the PSAP.*

If no *rollback indication* was received from the superior, issue the first applicable request of the following to the superior:

a) if a TP-U-ABORT request has been received for the superior and a log-damage record exists,

— issue an AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) request. When issued, the AF-ABORT-AND-HEURISTIC-REPORT request carries the Heuristic-Report parameter set to the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

current value of the log-damage record;

b) if a TP-U-ABORT request has been received for the superior,

— issue an AF-ABORT (user, rollbackRI) request;

c) if a log-damage record exists,

— issue an AF-HEURISTIC-REPORT (rollbackRI) request. When issued, the AF-HEURISTIC-REPORT request carries the Heuristic-Report parameter set to the current value of the log-damage record;

d) if none of the above conditions could be met,

— issue a C-ROLLBACK request.

11.5.11 Rollback next transaction

Invoking Procedure Name	Clause
TP-U-ABORT request	11.3.19
Protocol Error, Internal Error, A[-P]-ABORT indication, AF-ABORT (provider, abortRI) indication, A-ABORT request, A-RELEASE (Result= affirmative) response, or A-RELEASE (Result=affirmative) confirm on a Dialogue	11.3.21
CAF-RECOVER (ready) indication	11.3.58
CAF-RECOVER (commit) indication	11.3.60

If a TP-ROLLBACK indication is not pending,

— make a TP-ROLLBACK indication pending.

NOTE — The pending TP-ROLLBACK indication is issued at the completion of this transaction in the "Completing commitment" procedure (see 11.5.1).

If the TPPM has no subordinates,

— continue.

For each subordinate *dialogue* that *has not been detached* that meets the following conditions:

a) a C-COMMIT+C-BEGIN request was issued and a *commit confirm* was received; and

b) no C-ROLLBACK request or AF-ABORT request was issued;

then

a) if no TP-U-ABORT request was received,

— issue a C-ROLLBACK request;

b) if a TP-U-ABORT request was received,

— issue an AF-ABORT (user, rollbackRI) request;

— issue a SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request.

11.5.12 Sending commit order

Invoking Procedure Name	Clause
AF-ABORT (user, commitRI) indication	11.3.49
C-COMMIT indication or C-COMMIT+C-BEGIN indication	11.3.48
CAF-RECOVER (commit) indication	11.3.60
C-RECOVER (commit) indication	11.3.61
Making commitment decision	11.5.7

For each subordinate dialogue:

a) if the subordinate *dialogue is chaining*, and

1) if this is a root node or the superior *dialogue is not chaining*,

— issue a C-COMMIT+C-BEGIN request with the value of the

i) Atomic Action Identifier parameter set to a newly created value;

ii) Branch Identifier parameter set to a value that uniquely identifies the transaction branch within the scope of the atomic action identifier;

2) if the superior *dialogue is chaining*,

— issue a C-COMMIT+C-BEGIN request with the value of the

i) Atomic Action Identifier parameter set to the value of the atomic action identifier of the most recent C-BEGIN indication;

ii) Branch Identifier parameter set to a value that uniquely identifies the transaction branch within the scope of the atomic action identifier;

b) if the subordinate *dialogue is not chaining* and

1) if the subordinate *dialogue has not been detached*,

i) if a TP-U-ABORT request has been received for this subordinate,

— issue an AF-ABORT (user, commitRI)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

request to the subordinate;

ii) if a TP-U-ABORT request has not been received for this subordinate,

— issue a C-COMMIT request to the subordinate;

2) if the subordinate *dialogue has been detached,*

i) if a channel is not *attached,*

— issue a CAF-PLEASE request with the AE-Title of the CAF-PLEASE request set to the value of the AE-Title taken from the branch identifier for the subordinate, contained in the log-commit record.

ii) if a channel is *attached,*

— issue a C-RECOVER (commit) request if no recovery-context-handle is present in the log-ready record;

NOTE — Since the channel is *attached,* a CAF-RECOVER (ready) indication will have been received and CCR permits the issuance of a C-RECOVER (commit) request without the *token* when following a C-RECOVER (ready) indication. If the *token* is owned and the channel is two-way-recovery, it will be returned after the C-RECOVER (commit) request, as specified below.

— issue an AF-TOKEN-GIVE (two-way-recovery) request if the *token* is owned on the channel and this is a two-way-recovery channel.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

12 Structure and encoding of TP APDUs

12.1 Abstract syntax of the TP-ASE APDUs

```
Transaction-Processing-APDUs

{joint-iso-ccitt transaction-processing(10) modules (1) apdus-abstract-syntax(1)
version1(0)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS
-- all definitions --

IMPORTS
      APPLICATION-SERVICE-ELEMENT
       FROM Remote-Operations-Notation-extension
       {joint-iso-ccitt remote-operation(4) notation-extension(2)}


-- object identifier assignments

id-as-tpase OBJECT IDENTIFIER ::=
      {joint-iso-ccitt transaction-processing(10) abstract-syntax(2) tp-apdus(1)}

            -- may be used to reference the abstract syntax of the
            -- transaction processing ASE APDUs

tpASE1 APPLICATION-SERVICE-ELEMENT ::=
      {joint-iso-ccitt transaction-processing(10) ase-id(0) tp-ase(1) version1(0)}

            -- may be used to identify the transaction processing ASE


-- top level APDU CHOICE

TPASE-APDU ::=  CHOICE
{
tp-begin-dialogue-ri              [1]  TP-BEGIN-DIALOGUE-RI,
tp-begin-dialogue-rc              [2]  TP-BEGIN-DIALOGUE-RC,
tp-bid-ri                         [3]  TP-BID-RI,
tp-bid-rc                         [4]  TP-BID-RC,
tp-end-dialogue-ri                [5]  TP-END-DIALOGUE-RI,
tp-end-dialogue-rc                [6]  TP-END-DIALOGUE-RC,
tp-u-error-ri                     [7]  TP-U-ERROR-RI,
tp-u-error-rc                     [8]  TP-U-ERROR-RC,
tp-abort-ri                       [9]  TP-ABORT-RI,
tp-grant-control-ri               [10] TP-GRANT-CONTROL-RI,
tp-request-control-ri             [11] TP-REQUEST-CONTROL-RI,
tp-handshake-ri                   [12] TP-HANDSHAKE-RI,
tp-handshake-rc                   [13] TP-HANDSHAKE-RC,
tp-handshake-and-grant-control-ri
                                  [14] TP-HANDSHAKE-AND-GRANT-CONTROL-RI,
tp-handshake-and-grant-control-rc
                                  [15] TP-HANDSHAKE-AND-GRANT-CONTROL-RC,
tp-defer-ri                       [16] TP-DEFER-RI,
tp-prepare-ri                     [17] TP-PREPARE-RI,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
tp-heuristic-report-ri          [18] TP-HEURISTIC-REPORT-RI,
tp-token-give-ri                [19] TP-TOKEN-GIVE-RI,
tp-token-please-ri              [20] TP-TOKEN-PLEASE-RI,
tp-recover-ri                   [21] TP-RECOVER-RI,
tp-initialize-ri
                                [22] TP-INITIALIZE-RI,
tp-initialize-rc
                                [23] TP-INITIALIZE-RC
}

-- individual APDU definitions.


TP-BEGIN-DIALOGUE-RI ::=  SEQUENCE
{           CHOICE
                {dialogue    [1]  SEQUENCE
                     {initiating-tpsu-title  [1]  TPSU-title OPTIONAL,
                      recipient-tpsu-title   [2]  TPSU-title OPTIONAL,
                      functional-units       [3]  FU-list DEFAULT {
                                                   shared-control,
                                                   commit-and-chained-transactions
                     --  dialogue is always selected.
                     --  Only one (or neither) of "commit-and-chained-transactions
                     --  "commit-and-unchained-transactions" shall be selected.
                     --  "recovery" shall not be selected.
                      begin-transaction      [4]  BOOLEAN OPTIONAL,
                      confirmation           [5]  ENUMERATED
                                                   {always      (1),
                                                    negative    (2)
                                                   } DEFAULT  negative,
                      correlator             [6]  Correlator,
                      last-partner-identifier [7] Correlator OPTIONAL,
                      user-data              [30] User-information OPTIONAL
                     },

                 channel     [2]  SEQUENCE
                     {functional-units       [1]  FU-list DEFAULT {recovery},
                          --  Only Recovery shall be selected.
                          --  Default is Recovery.
                      correlator             [2]  Correlator,
                      channel-utilization    [3]  ENUMERATED
                                                   {one-way-recovery (1),
                                                    two-way-recovery (2)}
                                                     DEFAULT one-way-recovery,
                      last-partner-identifier [4] Correlator OPTIONAL
                     }
                }
}


TP-BEGIN-DIALOGUE-RC ::=  SEQUENCE
{           CHOICE
                {dialogue    [1]  SEQUENCE
                     {functional-units       [1]  FU-list   OPTIONAL,
                          --  "recovery" shall not be selected.
                      result                 [2]  ENUMERATED
                                                   {accepted          (1),
                                                    rejected-provider (2),
                                                    rejected-user     (3)
                                                   }     DEFAULT  accepted,
                                diagnostic   [3]  ENUMERATED
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
                                        {recipient-tpsu-title-unknown   (1),
                                         tpsu-not-available-permanent   (2),
                                         tpsu-not-available-transient   (3),
                                         recipient-tpsu-title-required  (4),
                                         functional-unit-not-supported  (5),
                                         functional-unit-combination-not-supported
                                                                        (6),
                                         association-reserved           (7),
                                         no-reason-given                (8)
                                        }       OPTIONAL,
                                    correlator  [4]  Correlator,
                                    user-data   [30] User-information OPTIONAL

                            },

                        channel     [2]  SEQUENCE
                            {result                     [1]  ENUMERATED
                                                             {accepted          (1),
                                                              rejected-provider (2)
                                                             }      DEFAULT   accepted,
                                    diagnostic  [2]  ENUMERATED,
                                                {
                                                 functional-unit-not-supported (1),
                                                 association-reserved          (2),
                                                 tppm-recovery-not-available   (3),
                                                 two-way-recovery-not-supported (4),
                                                 no-reason-given               (5)
                                                }       OPTIONAL,
                                    correlator  [3]  Correlator
                            }
                    }
}


TP-BID-RI ::=  SEQUENCE
{                       ccr-token-requested     [1]  BOOLEAN  DEFAULT  FALSE,
                        last-partner-identifier [2]  Correlator OPTIONAL
}


TP-BID-RC ::=  SEQUENCE
{                       result                  [1]  ENUMERATED
                                                      {accepted   (1),
                                                       rejected   (2)
                                                      } DEFAULT accepted
}


TP-END-DIALOGUE-RI ::=  SEQUENCE
{                       confirmation            [1]  BOOLEAN DEFAULT FALSE
}


TP-END-DIALOGUE-RC ::=  SEQUENCE
{
}


TP-U-ERROR-RI ::=  SEQUENCE
{
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
TP-U-ERROR-RC ::=  SEQUENCE
{
}


TP-ABORT-RI ::=  SEQUENCE
{     type  CHOICE
                  {user        [1]  SEQUENCE
                                     {user-data  [30] User-information   OPTIONAL
                                     },
                   provider    [2]  SEQUENCE
                                     {diagnostic [1]  ENUMERATED
                                                      {permanent-failure      (1),
                                                       begin-transaction-reject (2),
                                                       transient-failure      (3),
                                                       protocol-error         (4)
                                                      }
                                     }
                  }
}


TP-GRANT-CONTROL-RI ::=  SEQUENCE
{
}


TP-REQUEST-CONTROL-RI ::=  SEQUENCE
{
}


TP-HANDSHAKE-RI ::=  SEQUENCE
{                      confirmation-urgency     [1]  Confirmation-urgency OPTIONAL
}


TP-HANDSHAKE-RC ::=  SEQUENCE
{
}


TP-HANDSHAKE-AND-GRANT-CONTROL-RI ::=  SEQUENCE
{                      confirmation-urgency     [1]  Confirmation-urgency
                                                     DEFAULT  urgent
}


TP-HANDSHAKE-AND-GRANT-CONTROL-RC ::=  SEQUENCE
{
}


TP-DEFER-RI ::=  SEQUENCE
{                        type               [1]  ENUMERATED
                                                 {end-dialogue     (1),
                                                  grant-control    (2)
                                                 }  DEFAULT  end-dialogue
}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
TP-PREPARE-RI ::=  SEQUENCE
{                       data-permitted     [1]  BOOLEAN  OPTIONAL
                                                 -- present if polarized-control
}


TP-HEURISTIC-REPORT-RI ::=  SEQUENCE
{                       heuristic-report         [1]  ENUMERATED
                                                      {heuristic-mix    (1),
                                                       heuristic-hazard (2)
                                                      }   DEFAULT   heuristic-mix
}


TP-TOKEN-GIVE-RI::=  SEQUENCE
{
                        reason                   [1]  ENUMERATED
                                                      { regular          (1),
                                                        keep             (2),
                                                        two-way-recovery (3)
                                                      }     DEFAULT  regular,
                        correlator               [2]  Correlator   OPTIONAL
}


TP-TOKEN-PLEASE-RI::=  SEQUENCE
{
}


TP-RECOVER-RI ::=  SEQUENCE
{                       recovery-context-handle [1]  Recovery-context-handle
}


TP-INITIALIZE-RI ::=  SEQUENCE
{                       protocol-version         [1]  Protocol-versions
                                                           DEFAULT {version1},
                  contention-winner-assignment   [2]  BOOLEAN DEFAULT TRUE,
                        -- The value 'TRUE' means that the association initiator is
                        -- the contention-winner.  The value 'FALSE' means that the
                        -- association acceptor is the contention-winner.

                        bid-mandatory            [3]  BOOLEAN DEFAULT TRUE,
                        -- The value TRUE means that the Bid mechanism must be used.
                        -- The value FALSE means that it may optionally be used.

                        recovery-context-handle  [4]  Recovery-context-handle
                                                           OPTIONAL
}


TP-INITIALIZE-RC ::= SEQUENCE
{                       protocol-version         [1]  Protocol-versions
                                                           DEFAULT {version1},
                        recovery-context-handle  [2]  Recovery-context-handle
                                                           OPTIONAL,
                        diagnostic               [3]  BIT STRING
                                                      {ccr-version-2-not-available   (0),
                                          tp-protocol-version-incompatibility        (1),
                                          contention-winner-assignment-rejected      (2),
                                                       bid-mandatory-value-rejected  (3),
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
                                                    no-reason-given              (4)}
                                                                 OPTIONAL
                        -- the field is not present if the BIT STRING value is empty
}


-- Supporting type definitions --

Confirmation-urgency    ::=  ENUMERATED
                              {urgent      (1),
                               normal      (2)
                              }

Correlator ::=  INTEGER
                        -- unique within the scope of the association

FU-list                 ::=   BIT STRING
                              {polarized-control                              (0),
                               shared-control                                 (1),
                               commit-and-chained-transactions                (2),
                               commit-and-unchained-transactions              (3),
                               handshake                                      (4),
                               recovery                                       (5)
                              }

Protocol-versions       ::=  BIT STRING    {version1(0)}

Recovery-context-handle ::=  OCTET STRING

TPSU-title              ::=  CHOICE
                              {T61String,          --  May be used as an AttributeValue
                               PrintableString,    --  for an RDN in a Directory Name.
                               INTEGER
                              }

User-information        ::=  SEQUENCE OF EXTERNAL

END -- of TP-ASE definitions
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

12.2 Rules of extensibility

To provide for future compatibility, for the TP-INITIALIZE-RI/RC and TP-BEGIN-DIALOGUE-RI/RC APDUs, a receiving TPPM shall ignore a received field or field value that is not defined within the APDU in the ASN.1 description of this version of this part of ISO/IEC 10026.

An APDU that is not defined in the ASN.1 description of the negotiated version of this part of ISO/IEC 10026 shall not be sent. If such an APDU is received, it shall be treated as a protocol error.

A received field or field value that is not defined within an APDU in the ASN.1 description of the negotiated version of this part of ISO/IEC 10026 (other than fields or field values of TP-INITIALIZE-RI/RC and TP-BEGIN-DIALOGUE-RI/RC APDUS) shall either be ignored or treated as a protocol error.

The abstract syntax name may be used when new fields or field values are defined within any TPASE-APDU.

13 Conformance

13.1 Static conformance requirements

13.1.1 Conformance classes

13.1.1.1 General requirements

The system shall support one or more of the following conformance classes:

a) Application Transaction Branches Class;

b) Unchained Provider-Supported Transaction Branches Class;

c) Chained Provider-Supported Transaction Branches Class.

Support for conformance classes is defined in terms of support for functional units in the subclauses of 13.1.1, "Conformance Classes". Support for each functional unit is defined in terms of support for sending and/or accepting particular APDUs as defined in 13.1.2, "Capabilities" and 13.1.3, "Functional Units".

13.1.1.2 Application transaction branches class

A system in conformance with the Application Transaction Branches Class shall support

a) the requirements of the Dialogue functional unit;

b) the Shared Control functional unit or the Polarized control functional unit, or both; and

c) optionally, the Handshake functional unit.

13.1.1.3 Chained provider-supported transaction branches class

A system in conformance with the Chained Provider-Supported Transaction Branches Class shall support

a) the requirements of the Dialogue functional unit;

b) the Shared Control functional unit, or the Polarized Control functional unit, or both;

c) the Commit functional unit;

d) the Chained Transactions functional unit;

e) the Recovery functional unit; and

f) optionally, the Handshake functional unit.

13.1.1.4 Unchained provider-supported transaction branches class

A system in conformance with the Unchained Provider-Supported Transaction Branches Class shall support

a) the requirements of the Dialogue functional unit;

b) the Shared Control functional unit, or the Polarized Control functional unit, or both;

c) the Commit functional unit;

d) the Unchained Transactions functional unit;

e) the Recovery functional unit; and

f) optionally, the Handshake functional unit.

13.1.2 Capabilities

13.1.2.1 General capabilities

The system shall be capable of

a) initiating the establishment of an application association (by sending a TP-INITIALIZE-RI APDU and receiving a TP-INITIALIZE-RC APDU) (role "Ai"), or

accepting the establishment of an association (by receiving a TP-INITIALIZE-RI APDU and sending a TP-INITIALIZE-RC APDU) (role "Aa"), or

both initiating and accepting the establishment of an

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

association (roles Ai and Aa);

b) functioning as a contention-winner of an association (role "Cw"), or

functioning as a contention-loser of an association (role "Cl"), or

functioning as both a contention-winner and contention-loser (roles "Cw" and "Cl");

c) initiating a TP dialogue (role "Di"), or

accepting a TP dialogue (role "Da"), or

both initiating and accepting a TP dialogue (roles "Di" and "Da");

d) when the Commit functional unit is supported,

initiating a transaction branch (role "Ti"), or

accepting a transaction branch (role "Ta"), or

both initiating and accepting a transaction branch (roles "Ti" and "Ta");

e) when the Recovery functional unit is supported, the system shall have both capabilities, role Ai and Aa, stated in 13.1.2 a) above for the purpose of recovery;

NOTE — The following are required capabilities of all systems regardless of role.

f) rejecting a TP dialogue;

g) supporting the rules of extensibility specified in the "Rules of Extensibility" (see 12.2).

13.1.2.2 Constraints

A system in conformance with the role of initiating a transaction branch shall support initiating a TP dialogue with the Commit functional unit selected.

A system in conformance with the role of accepting a transaction branch shall support accepting a TP dialogue with the Commit functional unit selected.

13.1.3 Functional units

13.1.3.1 Definition

TP functional units are logical groupings of related TP protocol elements.

The TP functional units comprise

a) the functional units visible to the TP service, as defined in ISO/IEC 10026-2;

b) the Recovery functional unit, as defined below.

13.1.3.2 Description of recovery functional unit

The Recovery functional unit is only used on a TP channel. It is internal to the TPPM and the CPM and therefore is not directly accessible to any TPSUI.

The Recovery functional unit provides the protocols necessary to allow a CPM to send a request for, accept a request for, reject a request for, normally terminate and abnormally terminate a TP channel.

The Recovery functional unit also provides the protocol necessary to allow a TPPM to effect the rollback or commitment of transactions that have been affected by a failure.

13.1.3.3 Requirements on TP APDUs

Table 42 shows whether support for sending/receipt of an APDU is necessary to support a given functional unit. When an APDU shall be supported, in the context of the functional unit where it appears, independently of any capability, then it is marked "M" (mandatory).

Where the requirements depend on the capabilities that are supported (see 13.1.2.1), the two-letter mnemonics identifying the roles (e.g., Ai) are used: the notation "Xy" means "The sending (or receipt) of the APDU shall be supported if the capability identified by Xy is supported."

More complicated conditions are spelled out beneath the table. They are referred to in the table using the notation "(Cn)".

A system that does not fulfill a condition expressed in the support column is not required to be capable of sending or receiving the corresponding TP APDUs.

Where a TP APDU appears in both the Dialogue and the Recovery functional units, a system that supports both functional units shall meet the requirements of both sets of conditions.

NOTE — Implementations that are restricted in that they do not send all APDUs required by this clause may be reasonable in specific application environments. Currently, these implementations are not conforming to this standard. However, it is under study within ISO as to whether such implementations in the future may claim conformance to the TP Protocol. This note does not modify any part of clause 13.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 42 - TP Functional Units and support for TP APDUs

Functional Units	TP APDU	Support	
		Sending	Receiving
Dialogue	TP-BEGIN-DIALOGUE-RI ("dialogue" structure)	Di	M
	TP-BEGIN-DIALOGUE-RC (accept)	Da	Di
	TP-BEGIN-DIALOGUE-RC (reject)	M	Di
	TP-END-DIALOGUE-RI	(C1)	(C1)
	TP-END-DIALOGUE-RC	(C1)	(C1)
	TP-U-ERROR-RI	M	M
	TP-ABORT-RI	M	M
	TP-BID-RI	(C5)	Cw
	TP-BID-RC	Cw	(C2)
	TP-INITIALIZE-RI	Ai	Aa
	TP-INITIALIZE-RC	Aa	Ai
Shared Control	TP-U-ERROR-RC	M	M
Polarized Control	TP-GRANT-CONTROL-RI	M	M
	TP-REQUEST-CONTROL-RI	M	M
Handshake	TP-HANDSHAKE-RI	M	M
	TP-HANDSHAKE-RC	M	M
Handshake and Polarized Control	TP-HANDSHAKE-AND-GRANT-CONTROL-RI	M	M
	TP-HANDSHAKE-AND-GRANT-CONTROL-RC	M	M
Commit	TP-PREPARE-RI	Ti	Ta
	TP-DEFER-RI	Ti	Ta
	TP-HEURISTIC-REPORT-RI	Ta	Ti
	TP-TOKEN-GIVE-RI	M	M
Unchained Transactions	(none)		
Chained Transactions	(none)		
Recovery	TP-BEGIN-DIALOGUE-RI ("channel" structure)	M	M
	TP-BEGIN-DIALOGUE-RC	M	M
	TP-BID-RI	(C5)	Cw
	TP-BID-RC	Cw	(C2)
	TP-RECOVER-RI	M	(C4)
	TP-TOKEN-PLEASE-RI	(C3)	(C3)
	TP-END-DIALOGUE-RI	M	M
	TP-INITIALIZE-RI	M	M
	TP-INITIALIZE-RC	M	M

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table 42 *(concluded)*

(C1) The sending and receipt of both the TP-END-DIALOGUE-RI and TP-END-DIALOGUE-RC shall be supported when one, or both, of the Application Transaction Branches class or the Unchained Provider-supported Transaction Branches class are supported.
(C2) The receipt of the TP-BID-RC APDU shall be supported when the system is capable of sending the TP-BID-RI APDU.
(C3) Both the sending and the receipt of the TP-TOKEN-PLEASE-RI APDU shall be supported when the two-way-recovery facility is used.
(C4) The receipt of the TP-RECOVER-RI APDU shall be supported when the system supplies a recovery-context-handle on associations which it will use for initiating or accepting provider-supported transactions.
(C5) Sending TP-BID-RI is optional. To use a given association for either a dialogue or a channel, it may be necessary to bid. See clauses 8 and 10 for the specific circumstances under which bidding is required to use a particular association.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

13.1.4 Dependencies on other standards

The system shall also implement the following standards:

a) The CCR protocol in conformance with ISO/IEC 9805/Amd 2, if the Commit functional unit is supported.

The system shall support the role of a CCR superior if the system supports role Ti, and the system shall support the role of a CCR subordinate if the system supports role Ta.

If the system supports both roles Ti and Ta, then the system shall support the combination of the roles of CCR superior and CCR subordinate as specified in the TP procedures (clauses 9, 10, and 11);

b) The ACSE protocol in conformance with ISO 8650.

The system shall support the normal mode.

The system shall support the role of association initiator if the system supports role Ai and shall support the role of association responder if the system supports role Aa;

c) The Presentation protocol in conformance with ISO 8823/Amd 5.

In addition to those services used by ACSE, the system shall support P-DATA service primitives.

If the Commit functional unit is supported, then, in addition to those services used by CCR, the system shall support the use of the P-TOKEN-GIVE (synchronize-minor) service primitives. In addition, when the system supports the Recovery functional unit and uses the two-way-recovery facility, the system shall also support the P-TOKEN-PLEASE (synchronize-minor) service primitives;

d) The ASN.1 basic encoding rules in conformance with ISO/IEC 8825 (even if the system supports other encodings); and

e) The Session protocol in conformance with ISO 8327/Amd 3.

The system shall support version 2.

The system shall support the Kernel and Duplex functional units.

13.2 Dynamic conformance requirements

13.2.1 General requirements

a) The system shall correctly generate, accept, and respond to all valid protocol elements that support each class to which conformance is claimed;

b) The system shall respond to all incorrect sequences of TP protocol elements.

13.2.2 Specific requirements

For each conformance class to which conformance is claimed and for each option of the static conformance requirements implemented, the system shall exhibit external behaviour consistent with having implemented the following:

a) a TP Protocol Machine as specified in the "TP-ASE Description", "SACF Description", and "MACF Description" (clauses 9, 10, and 11) interpreted in accordance with the "Execution Rules" (clause 7);

b) the association management functions defined in the "Association Management" clause (8.5);

c) the use of the Association Control Service Element and of the Presentation Layer, as specified in 8.2, "Use of ACSE Service Primitives", and 8.4, "Use of the Presentation Layer";

d) encoding of TP APDUs as specified in 12.1, "Abstract Syntax of the TP-ASE APDUs"; and

e) embedding of APDUs as described in 6.1.7, "Embedding".

13.3 Protocol Implementation Conformance Statement

The supplier of a protocol implementation for which conformance to ISO/IEC 10026 is claimed shall complete a copy of the PICS proforma provided in ISO/IEC 10026-4, annex A, and shall provide the information necessary to identify both the supplier and the implementation.

NOTE — The concepts of a PICS and a PICS proforma are defined in ISO/IEC 9646.

13.4 Receiving TP APDUs

The semantics of some optional fields of some TP APDUS only apply to certain conformance options.

The system shall accept the syntax of all validly formatted fields in received TP APDUs. However, the system may ignore the semantics of those fields referred to in the first sentence of this subclause, if the system does not support the corresponding static conformance options.

14 Compliance

This part of ISO/IEC 10026 complies with the service user rules of CCR, ISO/IEC 9804.

15 Precedence statement

The text of clauses 7 through 12 takes precedence over the description contained in annex A.

Predicates, variables, and states in the state tables reflect both the text procedures (defined in the previous clauses) and the sequencing rules defined in ISO/IEC 10026-2. The text procedures are augmented with certain Service rules (see 7.2) to provide the same detection of illegal behaviour.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

16 Index of Actions and Events

A-ABORT indication, receive an 31
A-ABORT request, receive an 56, 58
A-RELEASE (Result=affirmative) confirm, receive an 56, 58
A-RELEASE (Result=affirmative) response, receive an 56, 58
ACTIVE state, enter the 60, 73, 76, 77
AF-ABORT (provider, abortRI) indication, receive an 56, 58
AF-ABORT (provider, abortRI) request, issue an 44, 56, 58, 67, 68, 69
AF-ABORT (provider, rollbackRC) request, issue an 41
AF-ABORT (provider, rollbackRI) request, issue an 60
AF-ABORT (user, commitRC) indication, receive an 64
AF-ABORT (user, commitRC) request, issue an 74
AF-ABORT (user, commitRI) indication, receive an 62
AF-ABORT (user, commitRI) request, issue an 80
AF-ABORT (user, dataRI) indication, receive an 55
AF-ABORT (user, dataRI) request, issue an 50, 54, 55, 63, 66
AF-ABORT (user, rollbackRC) request, issue an 64, 79
AF-ABORT (user, rollbackRI) request, issue an 55, 63, 76, 80
AF-ABORT (user/provider, rollbackRC) indication, receive an 66
AF-ABORT (user/provider, rollbackRI) indication, receive an 64
AF-ABORT indication, issue an 30, 31
AF-ABORT request, receive an 30
AF-ABORT-AND-HEURISTIC-REPORT (commitRC) indication, receive an 64
AF-ABORT-AND-HEURISTIC-REPORT (commitRC) request, issue an 74
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) indication, receive an 66
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) request, issue an 79
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication, receive an 64
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) request, issue an 79
AF-ABORT-AND-HEURISTIC-REPORT indication, issue an 31
AF-ABORT-AND-HEURISTIC-REPORT request, receive an 30
AF-BEGIN-DIALOGUE (accepted) confirm, receive an 49
AF-BEGIN-DIALOGUE (accepted, dataRI) response, issue an 48, 75
AF-BEGIN-DIALOGUE (accepted, rollbackRC) response, issue an 79
AF-BEGIN-DIALOGUE (rejected(provider), dataRI) response, issue an 36, 48
AF-BEGIN-DIALOGUE (rejected(user), dataRI) response, issue an 49
AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) confirm, receive an 51
AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) response, issue an 41, 49
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm, receive an 50
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) response, issue an 49
AF-BEGIN-DIALOGUE (rejected, dataRI) confirm, receive an 50
AF-BEGIN-DIALOGUE confirm, issue an 29, 31
AF-BEGIN-DIALOGUE confirm, receive an 51
AF-BEGIN-DIALOGUE indication, issue an 29
AF-BEGIN-DIALOGUE indication, receive an 48
AF-BEGIN-DIALOGUE request, issue an 48, 71
AF-BEGIN-DIALOGUE request, receive an 29
AF-BEGIN-DIALOGUE response, receive an 29
AF-BID (accepted) response, issue an 43
AF-BID confirm, issue an 29
AF-BID indication, issue an 29
AF-BID request, issue an 35
AF-BID request, receive an 29
AF-BID response, issue an 37, 45
AF-BID response, receive an 29
AF-DEFER (end-dialogue) request, issue an 61, 62
AF-DEFER (grant-control) request, issue an 61, 62
AF-DEFER indication, issue an 30
AF-DEFER indication, receive an 61
AF-DEFER request, receive an 30
AF-END-DIALOGUE confirm, issue an 29
AF-END-DIALOGUE confirm, receive an 54
AF-END-DIALOGUE indication, issue an 29
AF-END-DIALOGUE indication, receive an 52, 54
AF-END-DIALOGUE request, issue an 52, 73
AF-END-DIALOGUE request, receive an 29
AF-END-DIALOGUE response, issue an 54
AF-END-DIALOGUE response, receive an 29
AF-GRANT-CONTROL indication, issue an 30
AF-GRANT-CONTROL indication, receive an 58
AF-GRANT-CONTROL request, issue an 58
AF-GRANT-CONTROL request, receive an 30
AF-HANDSHAKE confirm, issue an 30
AF-HANDSHAKE confirm, receive an 59
AF-HANDSHAKE indication, issue an 30
AF-HANDSHAKE indication, receive an 59
AF-HANDSHAKE request, issue an 59
AF-HANDSHAKE request, receive an 30
AF-HANDSHAKE response, issue an 59
AF-HANDSHAKE response, receive an 30
AF-HANDSHAKE-AND-GRANT-CONTROL confirm, issue an 30
AF-HANDSHAKE-AND-GRANT-CONTROL confirm, receive an 60
AF-HANDSHAKE-AND-GRANT-CONTROL indication, issue an 30
AF-HANDSHAKE-AND-GRANT-CONTROL indication, receive an 59
AF-HANDSHAKE-AND-GRANT-CONTROL request, issue an 59
AF-HANDSHAKE-AND-GRANT-CONTROL request, receive an 30
AF-HANDSHAKE-AND-GRANT-CONTROL response, issue an 60
AF-HANDSHAKE-AND-GRANT-CONTROL response, receive an 30
AF-HEURISTIC-REPORT (commitRC) indication, receive

ECHORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

an 63
AF-HEURISTIC-REPORT (commitRC) request, issue an 74
AF-HEURISTIC-REPORT (recoverDoneRC) indication, receive an 70
AF-HEURISTIC-REPORT (recoverDoneRC) request, issue an 70, 74
AF-HEURISTIC-REPORT (rollbackRC) indication, receive an 65
AF-HEURISTIC-REPORT (rollbackRC) request, issue an 79
AF-HEURISTIC-REPORT (rollbackRI) indication, receive an 64
AF-HEURISTIC-REPORT (rollbackRI) request, issue an 80
AF-HEURISTIC-REPORT indication, issue an 31
AF-HEURISTIC-REPORT request, receive an 30
AF-PREPARE indication, issue an 30
AF-PREPARE indication, receive an 62
AF-PREPARE request, issue an 62
AF-PREPARE request, receive an 30
AF-RECOVER (commit) indication, receive an 69
AF-RECOVER (commit) request, issue an 71
AF-RECOVER (ready) indication, receive an 68
AF-RECOVER (ready) request, issue an 71
AF-RECOVER indication, issue an 30
AF-RECOVER request, receive an 30
AF-REQUEST-CONTROL indication, issue an 30
AF-REQUEST-CONTROL indication, receive an 58
AF-REQUEST-CONTROL request, issue an 58
AF-REQUEST-CONTROL request, receive an 30
AF-TOKEN-GIVE (keep) request, issue an 36, 43
AF-TOKEN-GIVE (regular) request, issue an 34, 35, 37, 41, 42, 43, 44
AF-TOKEN-GIVE (two-way-recovery) indication, receive an 70
AF-TOKEN-GIVE (two-way-recovery) request, issue an 70, 71, 81
AF-TOKEN-GIVE indication, issue an 32
AF-TOKEN-GIVE indication, receive an 70
AF-TOKEN-PLEASE indication, issue an 32
AF-TOKEN-PLEASE indication, receive an 71
AF-TOKEN-PLEASE request, issue an 71
AF-U-ERROR confirm, issue an 29
AF-U-ERROR confirm, receive an 54
AF-U-ERROR indication, issue an 29
AF-U-ERROR indication, receive an 54
AF-U-ERROR request, issue an 54
AF-U-ERROR request, receive an 29
AF-U-ERROR response, issue an 54
AF-U-ERROR response, receive an 29
AF-U-ERROR responses, issue 75
association, assign an 48, 71
A[-P]-ABORT indication, receive an 56, 58
BID CONFIRM RECEIVED state, enter the 37
BID INDICATION RECEIVED state, enter the 37
BIDDING state, enter the 35
BUSY state, enter the 36, 41, 42
C-BEGIN confirm, receive a 61
C-BEGIN indication, receive a 60
C-BEGIN request, issue a 75, 77
C-BEGIN response, issue a 61, 75
C-COMMIT confirm, receive a 31, 63
C-COMMIT indication, receive a 31, 62
C-COMMIT request, issue a 81
C-COMMIT response, issue a 74
C-COMMIT+C-BEGIN indication, receive a 62
C-COMMIT+C-BEGIN request, issue a 80
C-PREPARE indication, receive a 30
C-READY indication, receive a 62
C-READY request, issue a 75
C-RECOVER (commit) indication, receive a 69
C-RECOVER (commit) request, issue a 67, 71, 81
C-RECOVER (done) confirm, receive a 70
C-RECOVER (done) response, issue a 70, 74
C-RECOVER (ready) indication, receive a 68
C-RECOVER (ready) request, issue a 71
C-RECOVER (retry-later) confirm (CPM), receive a 70
C-RECOVER (retry-later) confirm, receive a 70
C-RECOVER (retry-later) response, issue a 67, 68, 69, 72
C-RECOVER (unknown) confirm (CPM), receive a 70
C-RECOVER (unknown) confirm, receive a 70
C-RECOVER (unknown) response, issue a 67, 68, 76
C-RECOVER confirm, receive a 31
C-RECOVER indication, receive a 30
C-ROLLBACK confirm, receive a 31, 65
C-ROLLBACK indication, receive a 31, 64
C-ROLLBACK request, issue a 40, 60, 63, 76, 80
C-ROLLBACK response, issue a 41, 42, 51, 64, 65, 76, 79
CAF-DETACH (Clean-up) request, issue a 69
CAF-DETACH (free) request, issue a 67, 68, 69, 70, 72, 74, 76
CAF-DETACH (not-used) request, issue a 71
CAF-DETACH request, receive a 72
CAF-FAIL indication, issue a 51, 52, 58, 68, 69
CAF-FAIL indication, receive a 72
CAF-GIVE indication, issue a 51, 70, 71
CAF-GIVE indication, receive a 71
CAF-PLEASE request, issue a 57, 58, 72, 73, 81
CAF-PLEASE request, receive a 71
CAF-RECOVER (commit) indication, issue a 70
CAF-RECOVER (commit) indication, receive a 68
CAF-RECOVER (ready) indication, issue a 68
CAF-RECOVER (ready) indication, receive a 67
channel, retain control of the 51
channel, Terminating a, (Internal Event) 73
channel, transfer the 51, 71, 72
CLEANUP BEGIN INDICATION EXPECTED state, enter the 34
CLEANUP ROLLBACK CONFIRM EXPECTED state, enter the 34, 40
CLEANUP ROLLBACK INDICATION EXPECTED state, enter the 34
CPM, attach to the 36
DECIDED (commit) state, enter the 78
DECIDED (rollback) state, enter the 74, 76
delay recovery (Internal Event) 72
final state, begin setting the TPPM bound data to the 78
final state, set the bound data set to the 73
forget the transaction 77, 79
FREE state, enter the 34, 35, 37, 38, 41, 42, 43, 44
Heuristic damage compensation for subtree (Internal Event) 72

ECTDOM.COM Click to view the full PDF of ISO/IEC 10026-3:1992

heuristic decision, Taking a, (Internal Event) 73
initial state, set the bound data set to the 73
initial state, set the TPPM bound data to the 72, 76
intermediate node, become an 76
internal error, receive an 56, 58
leaf node, become a 60, 74, 77
log-commit record, write a 77
log-damage record, remove the 72
log-damage record, write a 57, 76, 77, 78
log-heuristic record, remove the 72
log-heuristic record, write the 73
log-ready record, write a 75
MACF, create a new 36
MACF, detach the 35
node crash, Restart after, (CPM) (Internal Event) 72
node crash, TPPM creation after (Internal Event) 73
P-TOKEN-GIVE (sync-minor) indication, receive a 31
P-TOKEN-GIVE (sync-minor) request, issue a 75
pending, make a TP-ROLLBACK indication 80
pending, make the TP ROLLBACK indication no longer 74
protocol error, receive a 56, 58
PSAP, close the 50, 61, 63, 64, 66, 79
PSAP, open the 55, 74, 77
queue, discard the 34, 36, 37, 38, 40
queue, establish a 35, 40, 42, 43
queue, flush the 37, 43, 44
READY state, enter the 75
recovery, delay, (Internal Event) 72
recovery, Retry, (Internal Event) 72
Restart after node crash (CPM) (Internal Event) 72
Retry recovery (Internal Event) 72
rollback, TPPM-initiated, (Internal Event) 73
root node, become a 74, 76, 77
SAF-ASSOCIATION-LOST indication, issue a 36, 37, 40
SAF-ASSOCIATION-LOST indication, receive a 52
SAF-DETACH-ASSOCIATION (begin-fear) request, issue a 49, 52, 54, 55, 66
SAF-DETACH-ASSOCIATION (begin-indication-expected) request, issue a 49
SAF-DETACH-ASSOCIATION (free) request, issue a 49, 50, 51, 52, 53, 54, 55, 56, 63, 64, 65, 66, 73, 74, 79
SAF-DETACH-ASSOCIATION (rollback-confirm-expected) request, issue a 49, 50, 53, 55, 56, 60, 63, 80
SAF-DETACH-ASSOCIATION (rollback-indication-expected) request, issue a 50, 53, 56
STRAY state, enter the 35
Taking a heuristic decision (Internal Event) 73
Terminating a channel (CPM) (Internal Event) 73
token to the U-ASE, pass the 44
TP-ABORT-RI APDU, receive a 31
TP-ABORT-RI APDU, send a 30
TP-ABORT-RI TP APDU, receive a 30
TP-ABORT-RI TP APDU, send a 30
TP-BEGIN-DIALOGUE (rejected(provider)) confirm, issue a 48, 52
TP-BEGIN-DIALOGUE confirm, issue a 49, 50, 51
TP-BEGIN-DIALOGUE indication, issue a 48, 60
TP-BEGIN-DIALOGUE request, receive a 48
TP-BEGIN-DIALOGUE response, receive a 49
TP-BEGIN-DIALOGUE-RC APDU, receive a 31
TP-BEGIN-DIALOGUE-RC APDU, send a 29
TP-BEGIN-DIALOGUE-RC TP APDU, receive a 29
TP-BEGIN-DIALOGUE-RI APDU, send a 29
TP-BEGIN-DIALOGUE-RI TP APDU, receive a 29
TP-BEGIN-TRANSACTION indication, issue a 60
TP-BEGIN-TRANSACTION request, receive a 60
TP-BID-RC APDU, send a 29
TP-BID-RC TP APDU, receive a 29
TP-BID-RI APDU, send a 29
TP-BID-RI TP APDU, receive a 29
TP-COMMIT indication, issue a 73, 78
TP-COMMIT request, receive a 62
TP-COMMIT-COMPLETE indication, issue a 73
TP-DATA indication, issue a 61
TP-DATA request, receive a 61
TP-DEFER-RI APDU, send a 30
TP-DEFER-RI TP APDU, receive a 30
TP-DEFERRED-END-DIALOGUE indication, issue a 61
TP-DEFERRED-END-DIALOGUE request, receive a 61
TP-DEFERRED-GRANT-CONTROL indication, issue a 61
TP-DEFERRED-GRANT-CONTROL request, receive a 61
TP-DONE request, receive a 63
TP-END-DIALOGUE confirm, issue a 54
TP-END-DIALOGUE indication, issue a 52, 53
TP-END-DIALOGUE request, receive a 52
TP-END-DIALOGUE response, receive a 54
TP-END-DIALOGUE-RC APDU, send a 29
TP-END-DIALOGUE-RC TP APDU, receive a 29
TP-END-DIALOGUE-RI APDU, send a 29
TP-END-DIALOGUE-RI TP APDU, receive a 29
TP-GRANT-CONTROL indication, issue a 58
TP-GRANT-CONTROL request, receive a 58
TP-GRANT-CONTROL-RI APDU, send a 30
TP-GRANT-CONTROL-RI TP APDU, receive a 30
TP-HANDSHAKE confirm, issue a 59
TP-HANDSHAKE indication, issue a 59
TP-HANDSHAKE request, receive a 59
TP-HANDSHAKE response, receive a 59
TP-HANDSHAKE-AND-GRANT-CONTROL confirm, issue a 60
TP-HANDSHAKE-AND-GRANT-CONTROL indication, issue a 59
TP-HANDSHAKE-AND-GRANT-CONTROL request, receive a 59
TP-HANDSHAKE-AND-GRANT-CONTROL response, receive a 59
TP-HANDSHAKE-AND-GRANT-CONTROL-RC APDU, send a 30
TP-HANDSHAKE-AND-GRANT-CONTROL-RC TP APDU, receive a 30
TP-HANDSHAKE-AND-GRANT-CONTROL-RI APDU, send a 30
TP-HANDSHAKE-AND-GRANT-CONTROL-RI TP APDU, receive a 30
TP-HANDSHAKE-RC APDU, send a 30
TP-HANDSHAKE-RC TP APDU, receive a 30
TP-HANDSHAKE-RI APDU, send a 30
TP-HANDSHAKE-RI TP APDU, receive a 30
TP-HEURISTIC-REPORT indication, issue a 57, 63, 64, 65, 66, 68, 70
TP-HEURISTIC-REPORT-RI APDU, receive a 31
TP-HEURISTIC-REPORT-RI APDU, send a 30
TP-INITIALIZE-RC APDU, receive a 20

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

TP-INITIALIZE-RC APDU, send a 19
TP-INITIALIZE-RI APDU, receive a 19
TP-INITIALIZE-RI APDU, send a 18
TP-P-ABORT indication, issue a 53, 57, 60, 65, 66, 67, 68, 69
TP-PREPARE indication, issue a 62
TP-PREPARE request, receive a 61
TP-PREPARE-RI APDU, receive a 30
TP-PREPARE-RI APDU, send a 30
TP-READY indication, issue a 62
TP-RECOVER-RI APDU, receive a 30
TP-RECOVER-RI APDU, send a 30
TP-REQUEST-CONTROL indication, issue a 59
TP-REQUEST-CONTROL request, receive a 58
TP-REQUEST-CONTROL-RI APDU, send a 30
TP-REQUEST-CONTROL-RI TP APDU, receive a 30
TP-ROLLBACK indication no longer pending, make the 74
TP-ROLLBACK indication pending, make a 80
TP-ROLLBACK indication, issue a 74, 76
TP-ROLLBACK request, receive a 64
TP-ROLLBACK-COMPLETE indication, issue a 77
TP-TOKEN-GIVE-RI APDU, receive a 31
TP-TOKEN-GIVE-RI APDU, send a 32
TP-TOKEN-PLEASE-RI APDU, receive a 32
TP-TOKEN-PLEASE-RI APDU, send a 32
TP-U-ABORT indication, issue a 55, 56, 62, 64, 65, 66
TP-U-ABORT request, receive a 54
TP-U-ERROR indication, issue a 54
TP-U-ERROR request, receive a 54
TP-U-ERROR-RC APDU, send a 29
TP-U-ERROR-RC TP APDU, receive a 29
TP-U-ERROR-RI APDU, send a 29
TP-U-ERROR-RI TP APDU, receive a 29
TPPM creation after node crash, (Internal Event) 73
TPPM in the DECIDED (commit) state, create the 72
TPPM in the READY state, create the 72
TPPM, attempt to locate a 68, 70
TPPM-initiated rollback (Internal Event) 73
TPSUI, create a 48, 60
transaction, cease to be part of the 49, 57, 74, 77
transaction, forget the 73
U-ASE indication, receive a 61
U-ASE request, issue a 61

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex A
(normative)

OSI TP protocol — State tables

A.1 General

This annex describes the OSI TP Protocol in terms of state tables. The state tables show the state of both the TPPM and the CPM, the events that occur in the OSI TP Protocol, the actions taken, and the resulting states.

A.2 Introduction

A.2.1 States tables

The TP state tables used in the description of the OSI TP Protocol comprise:

a) the TPPM MACF state table, presented as four tables (tables A.13 to A.16);

NOTE 1 The TPPM MACF state table is presented as four tables because of the difficulty of presenting all the states on a page.

b) the CPM MACF state table (table A.17);

c) the SACF state table (table A.18); and

d) the TPASE table (table A.8).

NOTE 2 The TPASE (table A.8) encodes and decodes TP APDUs.

A.2.2 PM state machines

A.2.2.1 PM instance

A state machine is an instance of a state table. An instance of a PM consists of several instances of the state machines described in the following subclauses. The state machines that comprise a PM instance cooperate by the exchange of events (see A.2.3), the use of shared variables, and the counting mechanism (see A.3 rule f).

A.2.2.2 TPPM

The state of a TPPM at any particular time is represented by the state in each of the

a) TPPM MACF state machines. There is one state machine per dialogue branch or transaction branch, depending on the coordination level; and

b) SAO state machines. There is one SAO state machine per SAO in use by the TPPM, comprising:

1) a SACF state machine;

2) a TPASE state machine;

3) a CCRPM (refer to ISO/IEC 9805), if CCR is in the application context;

4) an ACPM (refer to ISO 8650); and

5) one or more U-ASE state machines.

The TPPM state consists of the combined set of states of each of the above state machines as well as additional context related to the entire TPPM.

NOTE — An example of this context is the node variables defined in A.2.5.

The TPPM MACF states are described in A.4.1.1. The SACF states are described in A.6.1.

A.2.2.3 CPM

The CPM MACF as described in 6.2.1 is represented by a CPM MACF state machine for each channel.

The state of a channel within the CPM at any particular time is represented by the state in each of the

a) CPM MACF state machine; and

b) SAO state machine. There is one SAO state machine per SAO in use by the CPM, comprising:

1) a SACF state machine;

2) a TPASE state machine;

3) a CCRPM (refer to ISO/IEC 9805); and

4) an ACPM (refer to ISO 8650)

The CPM MACF states are described in A.4.1.2. The SACF states are described in A.6.1.

A.2.3 Events

PM input events are defined in 7.1.2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

State machines are affected by **incoming events** which comprise PM input events, internal events and synchronizing events. As a result of processing a single incoming event, a state machine may generate zero or more **outgoing events**, some of which may become incoming events to another state machine.

NOTE 1 The following examples illustrate how events are received and generated by the state machines. Refer to Figure 1.

Example 1: the state of the TPPM MACF state machine is assumed to be in state 1. Input event TP-BEGIN-DIALOGUE request is received by the TPPM MACF state machine (P1). The event is received according to table A.13, state 1, and is processed by taking actions, among which outgoing event AF-BEGIN-DIALOGUE request is generated to the SACF state machine (P4).

Incoming event AF-BEGIN-DIALOGUE request is received by the SACF state machine (P4) according to table A.18, state 1. As a result, the SACF state machine issues an outgoing event which is either an AF-BEGIN DIALOGUE request or an AF-BID request to the TPASE (P5). The TPASE encodes the AF-Service request into a TP APDU, according to table A.8, and generates an outgoing event: it issues (or the SACF may possibly concatenate with other TP APDUs—see 6.1.5) the corresponding Presentation-service request with the TP APDU as user-data (P9).

Example 2: input event P-DATA indication with a TP-BID-RI APDU as user-data is received from the PSAP (P3). The APDU is decoded by the TPASE according to table A.8 and outgoing event AF-BID indication is generated to the SACF state machine (P10).

Incoming event AF-BID indication is received by the SACF state machine according to table A.18. Assuming that it is accepted, the SACF machine generates outgoing event AF-BID response to the TPASE (P5). The TPASE encodes the AF-Service request into a TP APDU, according to table A.8, and generates an outgoing event: it issues (or the SACF may possibly concatenate with other TP APDUs—see 6.1.5) the corresponding Presentation-service request with the TP APDU as User data (P9).

This second example shows an input event being completely handled by the TPASE and the SACF state machine without the involvement of the TPPM MACF state machine.

In addition, this annex also uses the following event types.

An **internal event** is an event that is created as a result of some internal decision or occurrence (including internal or protocol errors). Internal events are described in A.4.3.1 and A.6.3.1.

A **synchronizing event** is an event used to convey node-related information across all MACF state machines with coordination level "commitment". A synchronizing event is generated as a result of processing a node-related state machine incoming event on a single branch (a single MACF state machine), and is a state machine incoming event to *all* MACF state machines (with coordination level "commitment"), including the state machine that generated the synchronizing event.

NOTE 2 For example, upon the arrival of a *commit indication* on the superior dialogue, a *commit request* must be issued to each subordinate in the transaction tree. This is done by generating a synchronizing event to *all* TPPM MACF state tables with coordination level of "commitment"; where applicable, the TPPM MACF state machines (the ones representing subordinate dialogues or channels in this case) will take actions that include issuing the *commit request*.

A synchronizing event is generated only after the subcell (see A.2.8, Conventions) is completely processed, i.e., all actions are complete and the transition is made to the next state. If multiple synchronizing events are generated, they are generated (and processed) in sequence in the order requested and in the same action sequence (see A.3 rule b).

A further distinction of events is whether the event is global or not. A **global event** is an event that is applied to all MACF state machines. The global events are TP-COMMIT request, TP-ROLLBACK request, TP-DONE request, all synchronizing events, and some internal events (see A.4.3.1 for the global internal events).
The order of processing by the affected state tables of a global event is arbitrary, but the rule of atomicity holds (see A.3).

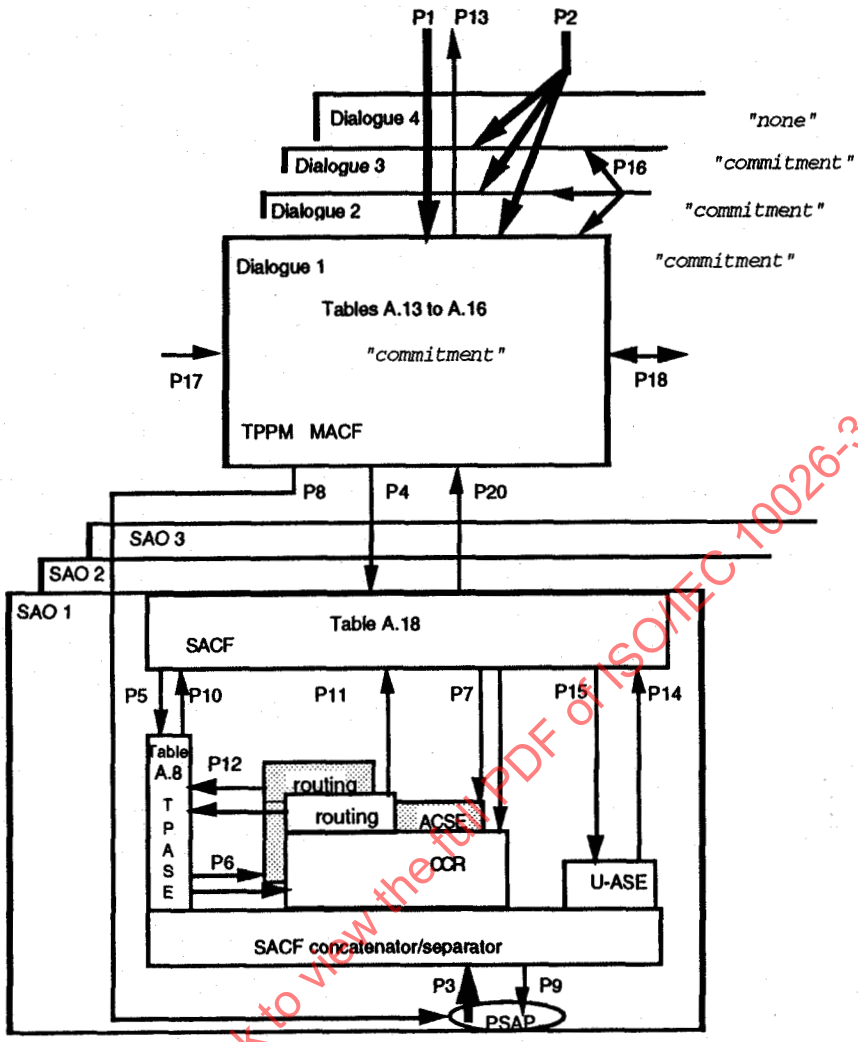
Internal and synchronizing events are described in A.4.3 and A.6.3. All other state machine input events are TP-, ACSE-, CCR-, AF-, SAF-, CAF-, or U-ASE-Services.

A.2.4 States

A state machine is in one state at any given time. Upon initial creation, the state machines are all in state 1, except upon creation after node crash, in which case they are created in the appropriate state (as defined by A.4.4.5, "Actions after node crash").

MACF states are described in A.4.1 and SACF states in A.6.1.

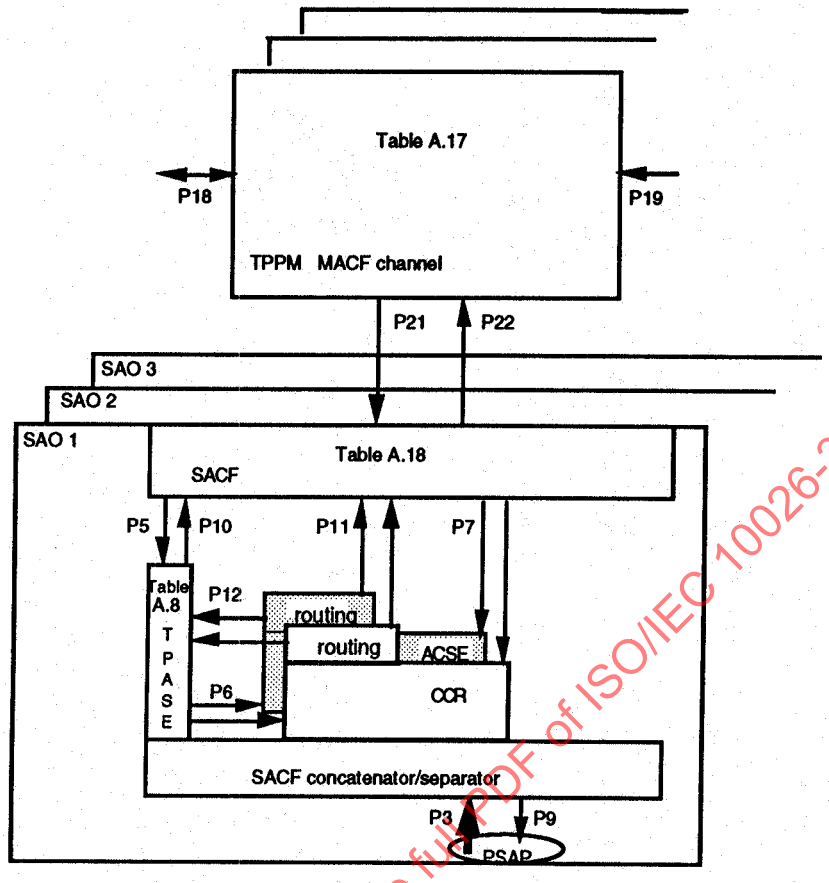
IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



P1 All TP- requests and responses except P2
P2 Global events: TP-COMMIT req, TP-ROLLBACK req, TP-DONE req
P3 All incoming APDUs
P4 All AF- requests and responses (except AF-BID request and response), C- requests and responses, U-ASE request and all SAF-DETACH requests
P5 All AF- requests and responses
P6 All C- and A- requests and responses carrying a TP APDU as user data
P7 All C- and A- requests and responses without a TP APDU as user data
P8 Open/close PSAP
P9 All outgoing APDUs
P10 All AF- indications and confirms
P11 All C-and A- indications and confirms without a TP APDU as user data
P12 All A- and C- indications and confirms which contain TP APDUs
P13 All TP- indications and confirms
P14 U-ASE indication
P15 U-ASE request
P16 All synchronizing events and global internal events
P17 TPPM internal events
P18 CAF- services
P20 All AF-, C-, A-, and U-ASE indications and confirms, and SAF-ASSOCIATION-LOST indication

Figure A.1 - Flow of events through the TPPM state machines

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



P3	All incoming APDUs
P5	All AF- requests and responses
P6	All C- and A- requests and responses carrying a TP APDU as user data
P7	All C- and A- requests and responses without a TP APDU as user data
P9	All outgoing APDUs
P10	All AF- indications and confirms
P11	All C- and A- indications and confirms without a TP APDU as user data
P12	All A- and C- indications and confirms which contain TP APDUs
P18	CAF services
P19	CPM internal events
P21	AF-BEGIN-DIALOGUE and C-RECOVER requests and responses; AF-END-DIALOGUE, AF-RECOVER, AF-TOKEN-PLEASE, and AF-TOKEN-GIVE requests
P22	AF-BEGIN-DIALOGUE, and C-RECOVER indications and confirms; AF-END-DIALOGUE, AF-RECOVER, AF-TOKEN-PLEASE, and AF-TOKEN-GIVE indications

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure A.2 - Flow of events through the CPM state machines

A.2.5 Variables and predicates

Each state machine uses variables for keeping track of certain information, and uses variables and predicates as conditions and predicate expressions. The variables are of type Boolean, Integer, Octet String, and Record Types (whose names begin with "T", and are described in A.4.2.2).

There are six categories of variables:

a) dialogue variables, (whose names begin with "D"), which are specific to each TPPM MACF state machines;

b) channel variables (whose names begin with "C"), which are specific to each CPM MACF state machine;

c) node variables (whose names begin with "N"), which are specific to the TPPM MACF state machines. These variables are shared between all TPPM MACF state machines for a node;

d) system variables (whose names begin with "S"), which are specific to the TPPM and CPM MACF state machines. These variables are accessible to any TPPM or CPM MACF state machine for the system. These variables also retain their value in the event of a node

crash;

e) local decision variables (whose names begin with "Ld"), which represent local decisions and options. Successive evaluations of local decision variables may yield different values. This non-deterministic behavior models potential changes in system resources and local strategies; and

f) association variables (whose names begin with "A"), which are related to a particular association. These variables are used by the SACF, however, some of these variables are also shared with the TPPM or CPM MACF while the MACF is attached.

Predicates (whose names begin with "P"), are inspected by any state machine and represent conditions outside of the TPPM.

Boolean functions which operate over sets of records are used to update and test membership in the system variables. These are described in A.4.4.1.

The Dialogue, Channel, Node, System, and some Local decision variables are described in A.4.2. The Association variables, and some Local decision variables are described in A.6.2. The Predicates are described in A.7.

A.2.6 Actions

Actions are presented in the state tables' cells between brackets ("[]"). These actions are described in A.4.4, A.5.4, and A.6.4.

For each valid incoming event (see A.2.8, "Conventions" below), all applicable actions are taken. Actions with free-form names often have conditions embedded in them.

A.2.7 Notation

Incoming events are represented by their name, with one or more attributes, when required. Some specific attributes are represented by a predicate, as follows:

AAI is a predicate which is the value of the *atomic-action identifier* parameter of the received service primitive; and

BI is a predicate which is the value of the *atomic-action-branch identifier* parameter of the received service primitive.

States are represented by a number. The integer part of the state number refers to the corresponding TP-Service state.

Predicate expressions are noted in the form of a list of variable values and/or predicates separated by commas.

"^" means "not", and is applied to variables of type Boolean and to Predicates.

"=" means "equal to", and is applied to variables of type Integer and Octet String.

"^=" means "not equal to", and is applied to variables of type Integer and Octet String.

">" means "greater than", and is applied to variables of type Integer.

A.2.8 Conventions

In the state tables, the intersection of an incoming event (row) and a state (column) forms a cell.

A subcell is a subset of a cell enclosed in a box.

The elements of a subcell are the following (given in their order of appearance in the subcell):

a) optionally, a predicate expression;

b) zero or more actions; and

c) a resultant state.

When a predicate expression holds for all the subcells of a same column, it is indicated at the top of the column, and not repeated in the subcells of that column.

A blank cell, a non-existent cell for an event, or a cell with no subcells for which the evaluation of the predicate expressions is true, represents an invalid event (see A.2.9.3) for that state.

A cell with a subcell for which the evaluation of the predicate expressions is true, represents a valid event (see A.2.9.2) for that state.

Predicate expressions in a cell are such that only one or zero subcells in a cell applies.

When a service primitive contains parenthetical arguments, these are as described in 9.2, 10.3, and 11.2, augmented by the following additional arguments:

a) a service parameter (left argument) and its value (right argument), separated by an equals sign (=);

b) the words "transaction branch" and "no transaction branch", which indicate that the TP-BEGIN-DIALOGUE request has been specified with the Chained Transactions funtional unit selected or with the Begin-Transaction parameter set to "true", or with the Begin-Transaction parameter either absent or set to "false", respectively;

c) the words "one-way-recovery" or "two-way-recovery", which indicate the value of the Channel-Utilization parameter;

d) the name of a functional unit selected on the dialogue or the channel, followed by the words "fu selected";

e) the words "sync-minor", which indicate the Session

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

token value.

These arguments may appear in any location inside the parentheses.

A.2.9 Processing of events

A.2.9.1 Evaluating predicate expressions

An event is processed by evaluating predicate expressions in all the subcells of the cell for the current state. If any subcell's predicate expression evaluates true (e.g., "^Aw, Ldres" is the predicate expression and Aw is FALSE and Ldres is TRUE), or no predicate expression exists for the subcell, the event is valid for the event/state combination, and the actions are taken and the transition made.

The evaluation of predicate expressions does not have any side effect; in particular, local decision variables retain their value during the evaluation of predicate expressions appearing in subcells of a same cell.

A.2.9.2 Processing valid events

For valid events, if the predicate expression (if any) is true, the following actions are taken:

a) the state machine performs the actions (if any) as shown in the cell; and

b) the state is changed to the specified resultant state (See also A.3, "Processing Rules", for additional rules in processing valid events).

A.2.9.3 Processing invalid events

Depending on the nature of the input events or state machine incoming events, one of the following actions is taken:

a) if the input event corresponds to the receipt of an invalid OSI TP Service primitive from the TPSUI, either an internal error or node crash is triggered, depending on a local decision as described in 7.1.6; or

b) if the state machine incoming event corresponds to receipt an invalid APDU from the partner TPPM, a protocol error is triggered as described in 7.1.6.

A.3 Processing rules

The following rules complement the rules of normal processing of events described in A.2.9:

a) atomicity;

An input event is processed completely before any other input event is accepted. This means that any outgoing events created by actions that are state machine incoming events to other state machines are processed by those state machines, and so on, until the only unprocessed events are outgoing events which are not state machine incoming events (that is, they are events at the PSAP or TPSUI).

When processing a given input event, state machines may either execute in parallel provided exclusive access to variables which are shared between state machines is maintained, or they shall be executed serially.

b) routeing;

When a service primitive received from the separator contains a TP APDU as user-data (or User Information in the case of ACSE), the service primitive becomes an event to the TPASE state machine.

When a service primitive received from the separator does not contain a TP APDU as user-data (or User Information in the case of ACSE), the service primitive becomes an event to the SACF state machine.

c) service conditions assumption;

The TPSUI is assumed to have issued requests and responses in accordance with the TPSUI conditions specified in ISO/IEC 10026-2.

NOTE 1 The state tables enforce the TPSUI constraints as specified in ISO/IEC 10026-2.

d) context mechanism;

When a subcell is executed, the actions taken by the subcell are related either to the dialogue, if attached, or the channel if no dialogue is available and a channel is attached. In some cases, there is both a dialogue and a channel or two channels attached during a single action sequence. A context mechanism is provided to identify whether the actions should occur on the dialogue or on the previously existing channel. This mechanism is implemented by the actions DIALOGUE and OLDCHANNEL. If actions include the detaching of a channel, the subsequent actions will be taken on the remaining channel. This context switching mechanism works only within a single subcell.

e) channel assignment;

When a CAF-PLEASE request is issued by a TPPM state machine, a CPM state machine which can accept the event is either created in state 1 or found in another state if the AE-title of the channel is the same as the AE-Title parameter of the CAF-PLEASE request.

f) counting mechanism;

For all conditions that depend on a certain number of events occuring, a counting mechanism is used. The counter is set to the number of events which must occur to cause the node transition. Each time an event which is to be counted occurs, the counter is decremented. When the counter becomes zero, the one-time actions

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

associated with the node are done by the state machine in which the counter becomes zero (see COUNTRDY, COUNTCOM, and COUNTRB). These one-time actions include generation of the appropriate synchronizing events.

NOTE 2 For example, when an intermediate node is to complete phase I of commitment, a C-READY indication must have been received on each subordinate branch and a TP-COMMIT request must have been received on each branch of the node. When the last of these events occur, the "Enter-ready" synchronizing event is generated, requesting the superior branch to issue the C-READY request. The counter would be set to the number of subordinate branches (for the C-READY indications) plus the total number of branches (for the TP-COMMIT requests).

The counting mechanism is used in the following three cases:

a) counting the events necessary to complete the first phase of commitment. These events are:

1) a C-READY indication for each subordinate branch; and

2) a TP-COMMIT request for each branch;

b) counting the events necessary to complete the second phase of commitment. These events are:

1) a *commit confirm* for each subordinate branch; and

2) a TP-DONE request for each branch;

c) counting the events necessary to complete the rollback. These events are:

1) a *rollback confirm* for each subordinate branch (except the subordinate branch from which a *rollback indication* has been received, if any); and

2) a TP-DONE request for each branch.

g) Parameter inheritance.

As in the main text, the definition of parameter inheritance (see 7.2 a) applies to the actions in this annex.

A.4 MACF state tables

A.4.1 MACF states

A.4.1.1 TPPM States

States are numbered with the following conventions:

a) the integer part of the state numbers corresponds to the state defined in 10026-2;

b) states 2 through 8 and states 12 through 14 correspond to a node handling either an application supported transaction or a provider supported transaction in the ACTIVE state;

c) states 9 through 11 are specific to a node handling an application supported transaction;

d) states 15 through 20.2 are specific to a node handling a provider supported transaction in the ACTIVE state;

e) state 20.3 corresponds to a transaction node in the READY state;

f) states 21.x correspond to a transaction node in the DECIDED (commit) state; and

g) states 23.x correspond to a transaction node in the DECIDED (rollback) state;

The following states are defined for the TPPM:

State 1
Idle state. No dialogue exists.

State 1.1
An AF-BEGIN-DIALOGUE indication has been received and the dialogue will have a coordination level of commitment. A C-BEGIN indication is awaited.

State 2
The TPSUI has control of the dialogue.

State 3
This state is valid only when the Polarized Control functional unit is selected. The dialogue is established and the TPSUI does not have control of the dialogue.

State 4
An AF-U-ERROR request was issued with the Shared Control functional unit selected or an AF-U-ERROR request was issued while the TPSUI did not have control of the dialogue.

State 5
This state is valid only when the Polarized Control functional unit is selected. The dialogue is established, the TPSUI has control of the dialogue, and an AF-U-ERROR indication has been received.

State 6
An AF-HANDSHAKE request has been issued. An AF-HANDSHAKE confirm is awaited.

State 7
An AF-HANDSHAKE indication has been received. A TP-HANDSHAKE response is awaited.

State 8
This state is valid only when both the Handshake and the

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Shared Control functional units are selected. An AF-HANDSHAKE indication has been received after an AF-HANDSHAKE request has been issued, or an AF-HANDSHAKE request has been issued after an AF-HANDSHAKE indication has been received.

State 9
This state is valid only when both the Handshake and the Shared Control functional units are selected. An AF-END-DIALOGUE (*confirmation* = TRUE) indication has been received after an AF-HANDSHAKE request has been issued.

State 10
This state is valid only when both the Handshake and the Shared Control functional units are selected. An AF-HANDSHAKE indication has been received after an AF-END-DIALOGUE (*confirmation* = TRUE) request has been issued.

State 11
An AF-END-DIALOGUE (*confirmation* = TRUE) request has been issued. An AF-END-DIALOGUE confirm is awaited.

State 12
An AF-END-DIALOGUE (*confirmation* = TRUE) indication has been received. A TP-END-DIALOGUE response is awaited.

State 13
An AF-HANDSHAKE-AND-GRANT-CONTROL request has been issued. An AF-HANDSHAKE-AND-GRANT-CONTROL confirm is awaited.

State 14
An AF-HANDSHAKE-AND-GRANT-CONTROL indication has been received. A TP-HANDSHAKE-AND-GRANT-CONTROL response is awaited.

State 15
This state is valid only for a dialogue with a subordinate. A TP-PREPARE request has been issued. A C-READY indication is awaited.

State 17
This state is valid only for a dialogue with a subordinate. A C-READY indication has been received. A TP-COMMIT request is awaited.

State 18
This state is valid only for the dialogue with the superior. An AF-PREPARE indication has been received. A TP-COMMIT request is awaited.

State 20.1
This state is valid only for a dialogue with a subordinate. A TP-COMMIT request has been received. A C-READY indication is awaited.

State 20.2
A TP-COMMIT request and a C-READY indication have been received. Synchronizing events "Continue-commit" (if a root node) or "Enter-ready" (if intermediate or leaf nodes) are awaited.

State 20.3
This state is valid only for an intermediate or a leaf node. The node is in the READY state. Synchronizing event "Enter-ready" has been received. A *commit indication* is awaited on the dialogue with the superior. Synchronizing event "Continue-commit" is awaited on dialogues with subordinates. Recovery may be in progress on dialogues with subordinates.

State 21.1
This state is valid only for a dialogue with a subordinate. A commit request has been issued. A *commit confirm* is awaited.

State 21.2
This state is valid only for a dialogue with a subordinate. A *commit confirm* is awaited. The next branch will not be rolled back if a C-COMMIT confirm or an AF-HEURISTIC-REPORT (commitRC) indication is received.

NOTE – An AF-ABORT (commitRC) indication is received only if the Unchained Transactions functional unit is selected.

State 21.3
This state is valid only for a dialogue with a subordinate that is chaining. A *commit confirm* has been received. Synchronizing event "Complete-commit" is awaited.

State 21.4
This state is valid only for a dialogue with a subordinate that is chaining. A *commit confirm* has been received. Rollback has been initiated on this branch. Synchronizing event "Complete-commit" is awaited.

State 21.5
This state is valid only for the dialogue with the superior. Synchronizing event "Complete-commit" is awaited.

State 21.6
This state is valid only for the dialogue with the superior that is chaining. The next branch will be rolled back. Synchronizing event "Complete-commit" is awaited if the association has not aborted.

State 23.1
This state is valid only for a dialogue with a subordinate. A *rollback request* has been issued. A rollback confirm is awaited.

State 23.2
This state is valid only for a dialogue with a subordinate. A *rollback indication* or a *rollback confirm* has been received. Synchronizing events "Report-rollback" or "Complete-rollback" are awaited.

State 23.3
This state is valid only for the dialogue with the superior. A TP-ROLLBACK request, Internal event "Rollback-by

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

TPPM", or synchronizing event "Rollback-all" has been received. Synchronizing event "Report-rollback" is awaited.

State 23.4
This state is valid only for the dialogue with the superior. A *rollback indication* has been received. Synchronizing event "Report-rollback" is awaited.

State 23.5
This state is valid only for the dialogue with the superior. A rollback report has been issued to the superior. A *rollback confirm* or synchronizing event "Complete-rollback" is awaited.

State 23.6
This state is valid only for the dialogue with the superior. A rollback report has been issued to the superior, and confirmation has been received. The dialogue with the superior *is available for the next transaction.* A C-BEGIN indication or synchronizing event "Complete-rollback" is awaited.

State 23.7
This state is valid only for the dialogue with the superior. A rollback report has been issued to the superior, and confirmation has been received. The dialogue with the superior *is not available for the next transaction.* A TP-DONE request or synchronizing event "Complete-rollback" is awaited.

State 23.8
This state is valid only for the dialogue with the superior. The dialogue with the superior has failed and the conditions for reporting rollback have not been fulfilled. The synchronizing event "Complete-rollback" is awaited.

State 25
This state is valid for a dialogue with a subordinate. This dialogue, having a coordination level of "commitment", has been terminated during the active state of the transaction without causing a rollback. This "zombie" dialogue will participate in the termination of the transaction.

State 99
On the dialogue with the superior, the TPPM is in the READY state. A CAF-PLEASE request has been issued. A CAF-GIVE indication is awaited.

On a dialogue with a subordinate, the TPPM is in the DECIDED (commit) state. A CAF-PLEASE request has been issued. A CAF-GIVE indication is awaited.

A.4.1.2 CPM states

State 1
Idle state. No channel exists.

State 2
The channel is free and may be allocated to a TPPM. For a one-way-recovery channel, the channel was initiated by this CPM and the AF-BEGIN-DIALOGUE (accepted) confirm has been received. For a two-way-recovery channel, the token is owned (unless the token will arrive as part of the channel establishment procedures managed by the SACF).

State 3
The channel is not free and must not be allocated to a TPPM. For a one-way-recovery channel, the channel was not initiated by this CPM. For a two-way-recovery channel, the token is not owned (and is not expected to arrive as part of the channel establishment procedures managed by the SACF).

State 4
The channel is temporarily owned by a TPPM.

State 5
The channel is established in the two-way-recovery mode and the token is awaited to perform recovery.

State 6
A channel is being established. An AF-BEGIN-DIALOGUE confirm is awaited.

State 7
The channel has been detached by the TPPM while C-RECOVER (ready) request was outstanding.

A.4.2 MACF variables

A.4.2.1 Overview

Six categories of variables are defined for MACF:

a) variables that pertain to a dialogue. These variables are created at dialogue establishment time, and are destroyed at dialogue termination time, unless otherwise specified. Dialogue variables are prefixed by the letter "D". They are listed in table A.1;

b) variables that pertain to a channel. Channel variables are prefixed by the letter "C". They are listed in table A.2;

c) variables that pertain to a node. These variables are created at establishment time of the first dialogue that includes the node as part of the dialogue tree and are destroyed at termination time of the last dialogue, unless otherwise specified. Node variables are prefixed by the letter "N". They are listed in table A.3;

d) variables that model open system data. System variables are prefixed by the letter "S". They are listed in table A.4;

e) variables that model a decision local to the node, when there is a choice for the TPPM. Local decision variables are prefixed by the letter "L". Local decision variables reflect a local decision made at the time of reference of the value of the variable. Therefore, the values of these variables are determined newly each time they are referenced. Local decision variables are listed in

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

table A.5; and

f) variables that are owned by SACF and shared with MACF when MACF is attached to the association. These variables are listed in table A.6.

Table A.1 - Dialogue variables

Name	Meaning
Da	dialogue establishment accepted
Dah	dialogue establishment accepted and held
Danyb	any abort received
Db	dialogue aborted and not available
Dbcr	C-BEGIN confirm received
Dbegdi	AF-BEGIN-DIALOGUE indication
Dbpart	abort issued to/received from partner
Dbrid	current branch identifier
Dbridn	branch identifier for next transaction
Dc	control
Dch	chaining dialogue with a subordinate
Dchat	channel attached
Dcr	confirmation requested
Dd	TP-DONE request owed
Ddef	AF-DEFER to be sent on prepare
Ddp	data permitted (polarized control mode)
De	defer end-dialogue
Denb	number of outstanding TP-U-ERROR requests
Denbb	number of outstanding TP-U-ERROR requests before TP-BEGIN-TRANSACTION request
Depnb	AF-U-ERROR responses number
Dfdone	first TP-DONE request received
Dg	defer grant-control
Dh	Handshake functional unit
Dl	coordination level
Dps	prepare sent
Drbrep	rollback reported to superior
Drvyp	rccovery pending
Dsh	Shared Control functional unit
Dsup	dialogue with superior
Dtb	abort received from TPSUI
Du	Unchained Transactions functional unit
Dx	transaction extended

Table A.2 - Channel variables

Name	Meaning
Caaid	atomic-action identifier
Cbrid	atomic-action-branch identifier
Cinit	channel initiator

Table A.3 - Node variables

Name	Meaning
Naaid	current atomic-action identifier
Naaidn	atomic-action identifier for next transaction
Nbrid	superior branch identifier
Nbridn	superior branch identifier for next transaction
Nch	chaining dialogue with the superior
Ncnt	count of events
Ncr	confirmation requested with superior
Nfa	failure actions allowed
Nfrb	first rollback request
Ni	intermediate node
Nlf	leaf node
Np	prepare indication received
Nr	root node
Nrn	reject not allowed
Nrpend	rollback pending
Nsubnb	subordinate number
Nt	transaction termination
Ntpsui	TPSUI created

Table A.4 - System variables

Name	Meaning
SldD	log-damage data
SlhD	log-heuristic data
SnD	node data

Table A.5 - MACF local decisions variables

Name	Meaning
Lddef	decision to delay AF-DEFER request
Ldfail	local failure in writing a log record
Ldperm	decision of permanent failure
Ldrej	decision to reject the dialogue
Ldretry	decision to retry
Ldretryo	decision to retry on the old channel
Ldt	decision to terminate the channel
Ldtwr	decision to have two-way-recovery
Ldunk	decision to reject, recipient unknown

Table A.6 - Shared variables with SACF

Name	Meaning
Arrh	received recovery-context-handle
Atokx	token expected
Atppm	attached to a TPPM
Atwr	two-way-recovery

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

A.4.2.2 Definitions of MACF types

Certain MACF variables contain sets of data which are structured into record types. These types are defined here. They are referred to only in A.4.2.3 which defines the MACF variables. Each type definition may be used as a set. Therefore, the type definition specifies the field(s) that uniquely identify the member of the set.

When a variable refers to a type definition, a field in the variable is referenced by the name of the variable, the value of the identifier field(s) in parenthesis (if the type is used as a set), a period, and the name of the field in the type definition.

NOTE — For example, suppose sbbr is a set of Tbranch. To reference the rch field for a branch with the brid of Dbrid, specify "sbbr (Dbrid).rch".

Tbranch (branch): Tbranch is a record which contains all necessary information about a single transaction branch (which could be to the superior or a subordinate). This record contains the following fields:

brid: contains the branch identifier for the specified branch.

aet: contains the peer AE-title, if the branch is to a subordinate.

rch: contains the peer partner recovery-context-handle for the branch, if provided.

The brid field identifies the record.

Tnode (node record): Tnode is a record which contains all of the information required for a node record. Tnode applies to any type of node record. This record contains the following fields:

aaid: contains the atomic action identifier.

spbr: contains a Tbranch record for the superior.

type: contains the type of log record. Possible values are "heuristic-hazard", "heuristic-mixed", "heuristic-initial", "heuristic-final", "log-commit", "log-ready", or NULL.

sbbr: contains a set of Tbranch records, one for each subordinate.

The aaid and spbr fields identify the record.

A.4.2.3 Definitions of MACF variables

The following variables are defined for the MACF state table. MACF variables are Boolean variables, unless otherwise specified.

Caaid (atomic-action identifier on channel): the value of Caaid indicates the atomic-action identifier used for recovery of the branch on the channel.

Cbrid (atomic-action-branch identifier on channel): the value of Cbrid indicates the atomic-action-branch identifier used for recovery of the branch on the channel.

Cinit (channel initiator): set to TRUE when the channel is initiated by the CPM.

Da (dialogue establishment accepted): when related to a dialogue with the superior and set to TRUE, Da indicates that a first request or response has been issued to the superior. When related to a dialogue with a subordinate and set to TRUE, Da indicates that a first indication or confirm has been received.

Dah (dialogue establishment accepted and held): when set to TRUE, Dah indicates that an TP-BEGIN-DIALOGUE (accepted) response has been issued in the DECIDED (rollback) state, but its propagation has not occured yet, because a TP-DONE request is awaited.

Danyb (any abort received): when set to TRUE, Danyb indicates that the dialogue has been or will be detached, or that the SAO is no longer attached (a SAF-ASSOCIATION-LOST indication has been received, or a SAF-DETACH-ASSOCIATION request has been issued). Danyb is true when Db, Dbpart, or Dtb is true.

Db (dialogue aborted and not available): when set to TRUE, Db indicates that the dialogue has been detached. Db is true when an SAF-DETACH-ASSOCIATION request has been issued or an SAF-ASSOCIATION-LOST indication has been received. Only one of Db, Dtb, or Dbpart may be true.

Dbcr (C-BEGIN confirm received): when set to TRUE, Dbcr indicates that a C-BEGIN confirm has been received. Dbcr is used to check the validity of an AF-END-DIALOGUE indication and an AF-ABORT indication when the Unchained Transactions functional unit is selected.

Dbegdi (AF-BEGIN-DIALOGUE indication): Dbegdi is used to save the AF-BEGIN-DIALOGUE indication so that the parameters from this indication are available when a TP-BEGIN-DIALOGUE indication is issued after the C-BEGIN indication arrives.

Dbpart (abort issued to/received from partner): when set to TRUE, Dbpart indicates that an AF-ABORT (user) request/indication or an AF-ABORT-AND-HEURISTIC-REPORT request/indication has been issued to or received from the partner TPPM. Only one of Db, Dtb, or Dbpart may be true.

In the case of commitment Dbpart is used to detect protocol errors. In the case of rollback Dbpart is used to repeat the abort if necessary in the event of a rollback collision.

Dbrid: (current branch identifier): the value of Dbrid indicates the atomic-action-branch identifier to a subordinate for the current transaction.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Dbridn: (branch identifier for next transaction): the value of Dbridn indicates the atomic-action-branch identifier for the next transaction.

Dc (control): when set to TRUE, Dc indicates that the TPSUI had control at the beginning of the transaction branch. Dc denotes which TPSUI will acquire the control of the dialogue, upon completion of rollback, should rollback occur.

Dch (chaining dialogue with a subordinate): when set to TRUE, Dch indicates that the dialogue *is chaining*.

Dchat (channel attached): when set to TRUE, Dchat indicates that a channel is attached to the TPPM for recovery of a particular branch.

Dcr (confirmation requested): when set to TRUE, Dcr denotes on a dialogue with a subordinate that the TPPM has received a TP-BEGIN-DIALOGUE (*confirmation* = "always") request and has not yet received an AF-BEGIN-DIALOGUE confirm.

Dd (TP-DONE request owed): when set to TRUE, Dd indicates that a *TP-DONE request is owed*. When set to FALSE, Dd indicates that a *TP-DONE request is not owed* by the TPSUI.

Ddef (AF-DEFER to be sent on prepare): when set to TRUE, an AF-DEFER request is to be issued when the AF-PREPARE request is issued. The type of the AF-DEFER request is determined by the values of De and Dg. When set to FALSE, no AF-DEFER request is issued when the AF-PREPARE request is issued.

Ddp (data permitted): when set to TRUE and if the Polarized Control functional unit is selected, Ddp indicates that a TP-DATA indication may be received by the superior TPSUI, after it has issued a TP-PREPARE request.

De (defer end-dialogue): when set to TRUE, De indicates that either an AF-DEFER (end-dialogue) request has been issued or an AF-DEFER (end-dialogue) indication has been received.

Denb (number of outstanding TP-U-ERROR requests): Denb is a variable of type Integer, used in shared control mode only. Denb indicates the number of outstanding TP-U-ERROR requests.

Denb is incremented by 1 upon issuance of a TP-U-ERROR request. Denb is decremented by 1 upon receipt of an AF-U-ERROR confirm, AF-HANDSHAKE indication, or AF-END-DIALOGUE (*confirmation* = TRUE) indication. Denb is set to zero upon occurence of a rollback.

Denbb (number of outstanding TP-U-ERROR requests before TP-BEGIN-TRANSACTION request): Denbb is a variable of type Integer, used in shared control mode and Unchained Transactions only.

Denbb indicates the number of TP-U-ERROR requests that were outstanding when the TP-BEGIN-TRANSACTION request was received. Denbb is set to the value of Denb when a TP-BEGIN-TRANSACTION request is received. Denbb is decremented whenever Denb is decremented.

Depnb (AF-U-ERROR response number): Depnb is a variable of type Integer, used in shared control mode only. Depnb indicates the number of TP-U-ERROR responses that shall be issued after an AF-BEGIN-DIALOGUE response is issued.

Dfdone (first TP-DONE request received): when set to TRUE, Dfdone indicates that the first TP-DONE request after TP-COMMIT indication or after a rollback initiating indication has been received. When set to FALSE, Dfdone indicates that a TP-DONE request with a heuristic-report parameter may be received, subject to the value of Dd.

Dg (defer grant-control): when set to TRUE, Dg indicates that either an AF-DEFER (grant-control) request has been issued or an AF-DEFER (grant-control) indication has been received.

Dh (Handshake functional unit): when set to TRUE, Dh indicates that the Handshake functional unit is selected.

Dl (coordination level): Dl reflects the value of the coordination level. When set to TRUE, Dl indicates that the coordination level is "commitment"; when set to FALSE, Dl indicates that the coordination level is "none".

Dps (prepare sent): Dps is set to TRUE after an AF-PREPARE request is received. It is used for subordinate dialogues only.

Drbrep (rollback reported to superior): when set to TRUE, Drbrep indicates that rollback has been reported to the superior. Drbrep is used by all transaction branches to avoid resetting Ncnt once rollback has been reported to the superior and a TP-DONE request becomes owed.

Drvyp (recovery pending): when set to TRUE, Drvyp enables the occurence of internal event "Retry-recovery" for that branch.

Dsh (Shared Control functional unit): when set to TRUE, Dsh indicates that the Shared Control functional unit is selected. When set to FALSE, Dsh indicates that the Polarized Control functional unit is selected.

Dsup (dialogue with superior): when set to TRUE Dsup indicates that the dialogue or transaction branch is with the superior.

Dtb (abort received from TPSUI): when set to TRUE, Dtb indicates that the TPSUI has issued a TP-U-ABORT request, but the issuance of the AF- Service has not yet occured. Once the AF-ABORT request has been issued or an AF-ABORT indication is received, Dtb is set to FALSE, and Db or Dbpart is set as appropriate. Only one of Db, Dtb or Dbpart may be true.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Du (Unchained Transactions functional unit): when set to TRUE, Du indicates that the Unchained Transactions functional unit is selected.

Dx (transaction extended): Dx is a variable of type Boolean which is set to TRUE when a TP-BEGIN-TRANSACTION request is received and is set to FALSE when a C-BEGIN confirm is received. Dx is used to determine if an AF-END-DIALOGUE indication or an AF-ABORT (user, dataRI) indication is valid for a subordinate dialogue with a coordination level "none".

Lddef (decision to delay AF-DEFER request): when set to TRUE, an AF-DEFER request will be issued when an AF-PREPARE request is issued. When set to FALSE, an AF-DEFER request is issued immediately.

Ldfail (local failure in writing a log record): when set to TRUE, Ldfail indicates that the TPPM cannot write a log record or set the bound data to the proper state.

Ldperm (decision of permanent failure): when set to TRUE, Ldperm indicates that the *diagnostic* parameter shall be set to "permanent-failure". When set to FALSE, Ldperm indicates that the *diagnostic* parameter shall be set to "transient-failure".

Ldrej (decision to reject the dialogue): when set to TRUE, Ldrej indicates that the TPPM takes a local decision to reject the dialogue establishment.

NOTE 1 This definition applies to both the initiator side (local reject) and responder side.

Ldretry (decision to retry): when set to TRUE, Ldretry indicates:

a) for a TPPM that it may issue a C-RECOVER (retry-later) response when, either the transaction outcome is not yet known to respond to a CAF-RECOVER (ready) indication, or when all commit confirms have not yet been received to respond to a CAF-RECOVER (commit) indication; and

b) for a CPM that it may issue a C-RECOVER (retry-later) response when the value of the recovery-context-handle does not allow it to determine if a TPPM can be found.

Ldretryo (decision to retry on the old channel): when set to TRUE, Ldretryo indicates that a C-RECOVER (retry-later) rsp will be sent on the old channel. Since this variable is used in conjunction with Ldretry in cases where at least one C-RECOVER (retry-later) response shall be sent, this variable shall not be set to FALSE if Ldretry is set to FALSE in the same subcell.

Ldt (decision to terminate the channel): when set to TRUE, Ldt indicates that channel utilization is to be terminated.

Ldtwr (decision to have two-way-recovery): when set to TRUE, Ldtwr indicates that the channel shall be established in the two-way-recovery mode. When set to FALSE, Ldtwr indicates that the channel shall be established in the one-way-recovery mode.

Ldunk (decision to reject, recipient unknown): when set to TRUE, Ldunk indicates that the dialogue is to be rejected because an association could not be established and the diagnostic parameter to be issued on the TP-BEGIN-DIALOGUE confirm is to be "recipient-unknown". When set to FALSE, the dialogue is rejected for other reasons and the diagnostic parameter of the TP-BEGIN-DIALOGUE confirm is "no-reason-given".

Naaid: (current-atomic action identifier): the value of Naaid indicates the atomic action identifier for the current transaction.

Naaidn: (atomic-action identifier for next transaction): the value of Naaidn indicates the atomic action identifier for the next transaction.

Nbrid: (superior branch identifier): the value of Nbrid indicates the atomic-action-branch identifier to the superior.

Nbridn: (superior branch identifier for next transaction): the value of Nbridn indicates the atomic-action-branch identifier to the superior for the next transaction.

Nch (chaining dialogue with the superior): when set to TRUE, Nch indicates that the dialogue with the superior *is chaining*.

Ncnt (count of events): the number of events that must occur before the next node state transition can occur. Ncnt is decremented for each relevant event (see clause A.3, rule f, counting mechanism).

Ncr (confirmation requested with the superior): when set to TRUE, Ncr indicates that a dialogue establishment indication is outstanding on the dialogue with the superior and therefore that a TP-BEGIN-DIALOGUE response is awaited.

Nfa (failure actions allowed): when set to TRUE, Nfa indicates that TP-U-ABORT requests are authorized during transaction termination.

Nfrb (first rollback): Used to determine that a TP-ROLLBACK request or TPPM initiated rollback request is processed so specific actions may be taken only once. Set to TRUE when a TP-ROLLBACK request or TPPM initiated rollback request is received by the first branch state machine.

Ni (intermediate node): when set to TRUE, Ni indicates that the node is an intermediate node of the transaction tree.

Nlf (leaf node): when set to TRUE, Nlf indicates that the node is a leaf node of the transaction tree.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Np (prepare indication received): when set to TRUE, Np means that an AF-PREPARE indication is received from the superior.

Nr (root node): when set to TRUE, Nr indicates that the node is the root node of the transaction tree.

Nrn (reject not allowed): when set to TRUE, Nrn denotes that the TPSUI has issued a request or response on any dialogue, and thus indicates that the dialogue with the superior can no longer be rejected.

Nrpend (rollback pending): when set to TRUE, Nrpend indicates that the synchronizing event "Rollback-next-trans" has been received .

Nsubnb (subordinate number): Nsubnb is a variable of type Integer. Nsubnb indicates the number of subordinates.

Nt (transaction termination): when set to TRUE, Nt indicates that the transaction branch has entered the termination phase, and that the transaction tree can no longer grow.

Ntpsui (TPSUI created): when set to TRUE, Ntpsui indicates that the TPSUI corresponding to the TPPM has been created.

SldD (log-damage data): SldD is a variable of type Set Of Tnode. SldD represents the set of log-damage records that are kept by an open system for appropriate heuristic reporting.

NOTE 2 To reference the value of the log record for a member of this set with a "aaid" value of Naaid and a "spbr" value of Nbrid, specify "SldD (Naaid, Nbrid).type.

SlhD (log-heuristic data): SlhD is a variable of type Set Of Tnode. SlhD represents the set of log-heuristic records that are kept by an open system for appropriate heuristic reporting.

NOTE 3 To reference the value of the log record for a member of this set with a "aaid" value of Naaid and a "spbr" value of Nbrid, specify "SlhD (Naaid, Nbrid).type.

SnD (node data): SnD is a variable of type Set Of Tnode. SnD represents the system data attached to transactions that have been initiated and that are not yet complete.

NOTE 4 To reference the value of the log record for a member of this set with a "aaid" value of Naaid and a "spbr" value of Nbrid, specify "SnD (Naaid, Nbrid).type. To reference the value of the "aei" field for a subordinate branch identified by Dbrid, specify "Snd (Naaid, Nbrid).sbbr (Dbrid).aei".

A.4.2.4 Initialization of MACF variables

MACF variables are initialized as follows:

Danyb and Db: initialized to TRUE after a transaction node crash.

All other variables are initialized as follows:

a) variables of type Boolean are initialized to FALSE;

b) variables of type Integer are initialized to zero; and

c) variables of type Octet string are initialized to EMPTY.

A.4.3 MACF events

A.4.3.1 Internal events

The following internal events, defined for the MACF state table, occur within the scope of a single branch:

Delay-recovery
Abbreviation for "Delay Recovery" as defined in 11.4.1.

Internal error
Abbreviation for "Internal Error", as defined in 11.3.21, 11.3.22, and 11.3.23.

Protocol error
Abbreviation for "Protocol Error", as defined in 11.3.21, 11.3.22, and 11.3.23.

Retry-recovery
Abbreviation for "Retry recovery", as defined in 11.4.4

Terminate-channel
Abbreviation for "Terminating a Channel", as defined in 11.4.6.

The following internal events, defined for the MACF state table, occur on all branches of a TPPM of a transaction tree:

Heuristic-damage-comp
Abbreviation for "Heuristic damage compensation for subtree", as defined in 11.4.2.

Heuristic-decision
Abbreviation for "Taking a Heuristic Decision" as defined in 11.4.5. Heuristic decision carries one parameter, called "heuristic-report", which can take the value of either "heuristic-final", "heuristic-initial", "heuristic-hazard", or "heuristic-mix".

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Restart-TPPM
Abbreviation for "TPPM creation after node crash" as defined in 11.4.7.

Rollback-by-TPPM
Abbreviation for "TPPM initiated rollback" as defined in 11.4.8.

A.4.3.2 Synchronizing events

The following synchronizing events are defined for the MACF state table: .

Complete-commit
This event is generated when commitment is completed at a node. When the superior receives this, it issues a *commit response*. When the subordinates see this, they complete the transaction.

Complete-rollback
This event is generated when rollback is complete and the next transaction may begin (if the Chained Transactions functional unit is selected on any dialogue). C-BEGIN request is sent on any dialogue with a subordinate that is available for the next transaction. Other branches involved in the rollback transaction are deleted from the transaction tree.

Continue-commit
"Continue-commit" is used by each dialogue with a subordinate to issue a *commit request*.

Enter-ready
This event is used by the dialogue to the superior to issue a C-READY request.

Report-rollback
This event is generated when all the conditions required to report the rollback to the superior are fulfilled. The dialogue to the superior uses this event to issue a *rollback response* or *rollback request* to the superior.

Rollback-all
This event is generated when a rollback occurs at a node. It is used by dialogues with subordinates to issue a *rollback request*.

Rollback-next-trans
This event is generated by a subordinate during commitment upon either receipt of A(-P)-ABORT indication, TP-U-ABORT request when a C-COMMIT+C-BEGIN indication has previously been received. All dialogues with subordinates must issue a *rollback request* if the *commit confirm* has been received.

Set-done-true
Indicates to all dialogues that a *TP-DONE request is* now *owed*. Each dialogue sets Dd to TRUE.

A.4.4 MACF actions

A.4.4.1 Functions

Variables which contain sets are manipulated by the functions decribed below. These functions are used to add and delete members of the set and to determine if a given member of a set exists.

addBranch (variable, brid)
Adds a new member of a set of type Tbranch. The variable parameter specifies a variable (or field of a variable) which is a Set Of Tbranch. The brid parameter specifies the value of the "brid" field of Tbranch which identifies the branch in the set.

addNode (variable, aaid, spbrid)
Adds a new member of a set of type Tnode. The variable parameter specifies a variable (or field of a variable) which is a Set Of Tnode. The aaid parameter specifies the value of the "aaid" field of Tnode, and the value of spbrid specifies the value of the "spbr.brid" field of Tnode. The aaid and the spbrid parameters identify the node record.

delBranch (A, X)
Deletes a member of a set of type Tbranch. The variable parameter specifies a variable (or field of a variable) which is a Set Of Tbranch. The brid parameter specifies the value of the "brid" field of Tbranch which identifies the branch in the set.

delNode (variable, aaid, spbrid)
Deletes a member of a set of type Tnode. The variable parameter specifies a variable (or field of a variable) which is a Set Of Tnode. The aaid parameter specifies the value of the "aaid" field of Tnode, and the value of spbrid specifies the value of the "spbr.brid" field of Tnode. The aaid and the spbrid parameters identify the node record.

memsb (variable, aaid, sbbrid)
Determines if a node record in the specified variable exists which refers to the specified *atomic-action-identifier* and subordinate *atomic-action-branch-identifier*. The aaid parameter specifies the value of the "aaid" field of Tnode which identifies the node record in the set. The sbbrid parameter specifies the value of a "sbbr.brid" field which identifies a branch within the node record. If the specified record is found, TRUE is returned; otherwise, FALSE is returned.

memsp (variable, aaid, spbrid)
Determines if a node record in the specified variable exists which refers to the specified *atomic-action-identifier* and superior *atomic-action-branch-identifier*. The aaid parameter specifies the value of the "aaid" field of Tnode which identifies the node record in the set. The spbrid parameter specifies the value of a "spbr.brid" field which identifies the branch with the superior within the node record. If the specified record is found, TRUE is returned; otherwise, FALSE is returned.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

A.4.4.2 Actions on services

Table A.7 lists actions that issue service primitives. These actions are named according to the following convention:

First character

A	AF-
C	C- or CAF-
P	P-
S	SAF-
T	TP-
U	U-

Next characters

AB	ABORT
AHR	ABORT-AND-HEURISTIC-REPORT
ASE	ASE
BD	BEGIN-DIALOGUE
BE	BEGIN
BT	BEGIN-TRANSACTION
DE	DEFERRED (end-dialogue)
DET	DETACH
DG	DEFERRED (grant-control)
DT	DATA
ED	END-DIALOGUE
GC	GRANT-CONTROL
GIV	GIVE
HR	HEURISTIC-REPORT
HS	HANDSHAKE
HSGC	HANDSHAKE-AND-GRANT-CONTROL
PAB	P-ABORT
PL	PLEASE
RB	ROLLBACK
PR	PREPARE
RC	REQUEST-CONTROL
RE	RECOVER
RY	READY
TOKG	TOKEN-GIVE
TOKP	TOKEN-PLEASE
UAB	U-ABORT
UE	U-ERROR

Service Primitive type

rq	Request
i	Indication
rs	Response
c	Confirm

Miscellaneous parameter values

A	result = Accepted
F	confirmation = False
RU	result = Rejected (user)
RP	result = Rejected (provider)
SB	Subordinate
SP	Superior
X	Inherited parameter value
TWR	reason = Two-way-recovery

Mapping parameter values

a	abortRI
d	dataRI
r	rollbackRI
rbc	rollbackRC
rd	recoverDoneRC
c	commitRI
crc	commitRC

Source parameter

SAVE uses the parameters as specified in the service primitive contained in a variable

The remaining characters qualify the action in a manner specific to the service being issued.

NOTE — For example, TUABiR issues a TP-U-ABORT indication with the rollback parameter set to TRUE.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.7 (1 of 3) - MACF actions on services

Action name	Parameter settings/service primitive issued
AABrqPa	AF-ABORT (provider, abortRI) req on the dialogue
AABrqPrTR	- set the *diagnostic* parameter to "begin-transaction-reject" AF-ABORT (provider, rollbackRI) req
AABrqUd	AF-ABORT (user, dataRI) req
AABrqUr	AF-ABORT (user, rollbackRI) req
AABrqUrbc	AF-ABORT (user, rollbackRC) req
ABDrq	AF-BEGIN-DIALOGUE (Dialogue fu selected) req
ABDrqRO	AF-BEGIN-DIALOGUE (Recovery fu selected, one-way-recovery) req
ABDrqRT	AF-BEGIN-DIALOGUE (Recovery fu selected, two-way-recovery) req
ABDrsAd	AF-BEGIN-DIALOGUE (accepted, dataRI) rsp
ABDrsArbc	AF-BEGIN-DIALOGUE (accepted, rollbackRC) rsp
ABDrsRPd	AF-BEGIN-DIALOGUE (rejected(provider), dataRI) rsp
ABDrsRUd	AF-BEGIN-DIALOGUE (rejected(user), dataRI) rsp
ABDrsRUr	AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) rsp
ABDrsRUrbc	AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) rsp
ADErq	AF-DEFER (end-dialogue) req
ADGrq	AF-DEFER (grant-control) req
AEDrq	AF-END-DIALOGUE req
AEDrqF	AF-END-DIALOGUE (confirmation = FALSE) req
AEDrs	AF-END-DIALOGUE rsp
AGCrq	AF-GRANT-CONTROL req
AHRrqHrdC	- set the *atomic-action-identifier* parameter to AAI - set the *atomic-action-branch-identifier* parameter to BI - set the *heuristic-report* parameter to SIdD (AAI, BI).type AF-HEURISTIC-REPORT (recoverDoneRC) req
AHSrq	AF-HANDSHAKE req
AHSrs	AF-HANDSHAKE rsp
AHSGCrq	AF-HANDSHAKE-AND-GRANT-CONTROL req
AHSGCrs	AF-HANDSHAKE-AND-GRANT-CONTROL rsp
APRrq	AF-PREPARE req
ARCrq	AF-REQUEST-CONTROL req
ATOKGrqTWR	AF-TOKEN-GIVE (two-way-recovery) req
ATOKPrq	AF-TOKEN-PLEASE req
AUErq	AF-U-ERROR req
AUErs	AF-U-ERROR rsp
CAFDETrqCU	- set the *type* parameter to "clean-up" CAF-DETACH req.
CAFDETrqF	- set the *type* parameter to "free" CAF-DETACH req.
CAFDETrqNU	- set the *type* parameter to "not-used" CAF-DETACH req.
CAFFAILi	CAF-FAIL ind to the transaction branch identified by the *atomic-action-identifier* value of Caaid and the *atomic-action-branch-identifier* value of Cbrid.
CAFGIVi	CAF-GIVE ind to the transaction branch identified by the *atomic-action-identifier* value of Caaid and the *atomic-action-branch-identifier* value of Cbrid.
CAFPLrqSB	- set the *AE-title* parameter to SnD (Naaid, Nbrid).sbbr (Dbrid).aet - set the *atomic-action-identifier* parameter to Naaid - set the *atomic-action-branch-identifier* parameter to Nbrid CAF-PLEASE req
CAFPLrqSP	- set the *AE-title* parameter to the *AE-title* of the superior (extractable from Nbrid) - set the *atomic-action-identifier* parameter to Naaid - set the *atomic-action-branch-identifier* parameter to Nbrid CAF-PLEASE req

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.7 (Continued 2 of 3) - MACF actions on services

Action name	Parameter settings/service primitive issued
CAFREiC	CAF-RECOVER (commit) ind to the transaction branch identified by the *atomic-action-identifier* value of AAI and the *atomic-action-branch-identifier* value of BI.
CAFREiR	CAF-RECOVER (ready) ind to the transaction branch identified by the *atomic-action-identifier* value of AAI and the *atomic-action-branch-identifier* value of BI.
CBErq	- set the *atomic-action-identifier* parameter to Naaid - set the *atomic-action-branch-identifier* parameter to Dbrid C-BEGIN req
CRBrq	C-ROLLBACK req
CRBrs	C-ROLLBACK rsp
CRErsDC	- set the *atomic-action-identifier* parameter to AAI - set the *atomic-action-branch-identifier* parameter to BI C-RECOVER (done) rsp
CRErsRT	- set the *atomic-action-identifier* parameter to Naaid - set the *atomic-action-branch-identifier* parameter to Nbrid C-RECOVER (retry-later) rsp
CRErsRTC	- set the *atomic-action-identifier* parameter to AAI - set the *atomic-action-branch-identifier* parameter to BI C-RECOVER (retry-later) rsp
CRErsU	- set the *atomic-action-identifier* parameter to AAI - set the *atomic-action-branch-identifier* parameter to BI C-RECOVER (unknown) rsp
CRYrq	C-READY req
PTOKGrq	P-TOKEN-GIVE (sync-minor) req
SDETrqBF	SAF-DETACH-ASSOCIATION (begin-fear) req
SDETrqCB	SAF-DETACH-ASSOCIATION (begin-indication-expected) req
SDETrqF	SAF-DETACH-ASSOCIATION (free) req
SDETrqRB	SAF-DETACH-ASSOCIATION (rollback-indication-expected) req
SDETrqRBC	SAF-DETACH-ASSOCIATION (rollback-confirm-expected) req
TBDcRP	- set the *rollback* parameter to FALSE - set the *diagnostic* parameter to "no-reason-given" TP-BEGIN-DIALOGUE (reject (provider)) cnf
TBDcRPr	- set the *rollback* parameter to TRUE - set the *diagnostic* parameter to "no-reason-given" TP-BEGIN-DIALOGUE (reject (provider)) cnf
TBDcRPru	- set the *rollback* parameter to TRUE - set the *diagnostic* parameter to "recipient-unknown" TP-BEGIN-DIALOGUE (reject (provider)) cnf
TBDcRPu	- set the *rollback* parameter to FALSE - set the *diagnostic* parameter to "recipient-unknown" TP-BEGIN-DIALOGUE (reject (provider)) cnf
TBDcX	- set the *rollback* parameter to FALSE TP-BEGIN-DIALOGUE cnf
TBDcXr	- set the *rollback* parameter to TRUE TP-BEGIN-DIALOGUE cnf
TBDi	TP-BEGIN-DIALOGUE ind
TBDiSAVE	- set the parameters to those in the AF-BEGIN-DIALOGUE ind saved in Dbegdi TP-BEGIN-DIALOGUE ind
TBTi	TP-BEGIN-TRANSACTION ind
TDEi	TP-DEFERRED-END-DIALOGUE ind
TDGi	TP-DEFERRED-GRANT-CONTROL ind
TDTi	TP-DATA ind
TEDc	TP-END-DIALOGUE cnf
TEDi	TP-END-DIALOGUE ind
TGCi	TP-GRANT-CONTROL ind

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.7 (Concluded 3 of 3) - MACF actions on services

Action name	Parameter settings/service primitive issued
THRi	TP-HEURISTIC-REPORT ind
THRiH	- set the *heuristic-report* parameter = "heuristic-hazard" TP-HEURISTIC-REPORT ind
THSc	TP-HANDSHAKE cnf
THSi	TP-HANDSHAKE ind
THSGCc	TP-HANDSHAKE-AND-GRANT-CONTROL cnf
THSGCi	TP-HANDSHAKE-AND-GRANT-CONTROL ind
TPABi	- set the *rollback* parameter to FALSE TP-P-ABORT ind
TPABiBTED	- set the *rollback* parameter to FALSE - set the *diagnostic* parameter to "begin-transaction-end-dialogue-collision" TP-P-ABORT ind
TPABiBTEDr	- set the *rollback* parameter to TRUE - set the *diagnostic* parameter to "begin-transaction-end-dialogue-collision" TP-P-ABORT ind
TPABiBTR	- set *diagnostic* parameter to "begin-transaction-reject" - set the *rollback* parameter to FALSE TP-P-ABORT ind
TPABiED	- set the *rollback* parameter to FALSE - set the *diagnostic* parameter to "end-dialogue-collision" TP-P-ABORT ind
TPABiR	- set the *rollback* parameter to TRUE TP-P-ABORT ind
TPRi	TP-PREPARE ind
TRBi	TP-ROLLBACK ind
TRCi	TP-REQUEST-CONTROL ind
TRYi	TP-READY ind
TUABi	- set the *rollback* parameter to FALSE TP-U-ABORT ind
TUABiR	- set the *rollback* parameter to TRUE TP-U-ABORT ind
TUEi	TP-U-ERROR ind
UASErq	U-ASE req

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

A.4.4.3 Actions on variables

For actions that manipulate MACF variables, the following conventions apply:

First character: V

The name of the variable being set begins at the second character.

The final characters are either:

DEC (decrement by one);
F (set to FALSE);
INC (increment by one);
SAVE (save the most recently received or issued specified service primitive in the specified variable); or
T (set to TRUE);

NOTE — An example is "VdaT", for "set Da to TRUE".

A.4.4.4 Actions with free-form names

[ABDET] (association aborted or detached)
Invoked when the association has been aborted or detached.
- set Db and Danyb to TRUE.
- set Dtb and Dbpart to FALSE.

[ABPTNR] (aborted dialogue with partner)
Invoked either when notification of a dialogue abort is received from or issued to the partner TPPM.
- set Dbpart and Danyb to TRUE.
- set Dtb to FALSE.

[ABTPSUI] (aborted by TPSUI)
Invoked upon receipt of a TP-U-ABORT request from TPSUI.
- set Dtb and Danyb to TRUE.

[ADDBRSB] (add branch subordinate)
Invoked when a branch with a subordinate is added to a transaction. Adds one for the C-READY indication and one for the TP-COMMIT request.
- set Dl to TRUE.
- if ^Nr and ^Ni and ^Nlf,
  - set Nr to TRUE,
  - set Naaid to a new unique value,
  - set Nbrid to NULL,
  - addNode (SnD, Naaid, Nbrid).
- if Nlf,
  - set Nlf to FALSE,
  - set Ni to TRUE.
- add 2 to Ncnt.
- add one unit to Nsubnb.
- set Dc to TRUE.
- if ^Du,
  - set Dch to TRUE.
- set Dbrid to a new unique value.
- addBranch (SnD (Naaid, Nbrid).sbbr, Dbrid).
- set SnD (Naaid, Nbrid).sbbr (Dbrid).rch to Arrh.

[ADDBRSP] (add branch superior)
Invoked when a transaction is created upon indication from the superior. Sets the counter to account for a TP-COMMIT request.
- set Dl to TRUE.
- set Nlf to TRUE.
- set Ncnt to 1.
- set Naaid to the *atomic-action-identifier* parameter.
- set Nbrid to the *atomic-action-branch-identifier* parameter.
- addNode (SnD, Naaid, Nbrid).
- set SnD (Naaid, Nbrid).spbr.rch to Arrh.
- if ^Du,
  - set Nch to TRUE.
- if Da,
  - issue a C-BEGIN rsp.

[BEGTRANS] (TP-BEGIN-TRANSACTION)
Invoked to flag that a TP-BEGIN-TRANSACTION request has been issued and to record the number of TP-U-ERROR requests that are outstanding at that time.
- set Denbb to Denb.
- set Dx to TRUE.

[CBEAFTRB] (C-BEGIN after rollback)
Invoked upon receipt of a C-BEGIN indication.
- issue a C-BEGIN rsp.
- delNode (SnD, Naaid, Nbrid).
- set Naaid to the *atomic-action-identifier* parameter.
- set Nbrid to the *atomic-action-branch-identifier* parameter.
- addNode (SnD, Naaid, Nbrid).
- set SnD (Naaid, Nbrid).spbr.rch to Arrh.

[CMPCOMSB]
(completing commitment with subordinate)
Completes the commitment processing for a single branch with a subordinate.
- if Dg, set Dc to FALSE.
- if ^Dg, set Dc to TRUE.
- if Dch, set Dbrid to Dbridn.
- open the PSAP.

[CMPCOMSP]
(completing commitment with superior)
Completes the commitment processing for the branch with the superior.
- if Dg, set Dc to TRUE.
- if ^Dg, set Dc to FALSE.
- if Nch, set Nbrid to Nbridn.

[CMPRBSB]
(completing rollback with subordinate)
Completes the rollback processing for a single branch with a subordinate.
- set Dbrid to a new unique value.
- addBranch (SnD (Naaid, Nbrid).sbbr, Dbrid).
- set SnD (Naaid, Nbrid).sbbr (Dbrid).rch to Arrh.

[COMREQ] (issue commit request)
Invoked to issue the correct type of *commit request* to a single subordinate.
- if Dch,
  - set Dbridn to a unique value,
  - set the *atomic-action-identifier* parameter to Naaidn,
  - set the *atomic-action-branch-identifier* parameter to Dbridn,
  - issue a C-COMMIT+C-BEGIN req,
  - addBranch (SnD (Naaidn, Nbridn).sbbr, Dbridn),
  - set SnD (Naaidn, Nbridn).sbbr (Dbridn).rch to Arrh.
- if Dtb,
  - issue a AF-ABORT (user, commitRI) req,
  - set Dbpart to TRUE,
  - set Dtb to FALSE.
- if ^Dch and ^Dtb,
  - issue a C-COMMIT req.

[COMRSP] (issue commit response)
Invoked to issue the correct type of *commit response* to the superior.
- if ^memsp (SldD, Naaid, Nbrid),
  - if Dtb and Du,
    - issue an AF-ABORT (user, commitRC) req.
  - if ^Dtb or (Dtb and ^Du),
    - issue a C-COMMIT rsp.
- if memsp (SldD, Naaid, Nbrid),
  - if Dtb and Du,
    - set *heuristic-report* parameter to SldD (Naaid, Nbrid).type,
    - issue an AF-ABORT-AND-HEURISTIC-REPORT (commitRC) req.
  - if ^Dtb or (Dtb and ^Du),
    - set *heuristic-report* parameter to SldD (Naaid, Nbrid).type,
    - issue an AF-HEURISTIC-REPORT (commitRC) req.
- if Dtb and ^Du,
  - set Dbpart to TRUE,
  - set Dtb to FALSE.

ECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

[COUNTCOM]
(count commitment confirm event)
Counts an event to complete commitment phase 2 processing for the node. Used to count a *commit confirm* or a TP-DONE request. When all events have been received, completes the one-time actions associated with commitment processing for a node. Also handles starting the rollback of the next transaction if that is to occur.

If the next transaction rolls back and commitment is complete, the counter is set to the number of subordinates from which a *rollback confirm* will be received plus the number of branches from which a TP-DONE request must be received.

If the next transaction does not rollback and commitment is complete, the counter is set to the number of subordinates from which a C-READY indication will be received plus the number of branches from which a TP-COMMIT request must be received.

- subtract one unit from Ncnt.

Begin the next transaction
- if Ncnt = 0,
    - issue TP-COMMIT-COMPLETE ind,
    - delNode (SnD, Naaid, Nbrid),
    - set Nfa and Np to FALSE,
    - if Nr,
        - set Naaid to Naaidn,
        - set Nbrid to Nbridn.
    - generate "Complete-commit",
    - if ^Nrpend,
        - set Nt to FALSE,

        Set the counter for phase one of commitment for the next transaction; one C-READY ind from each subordinate; a TP-COMMIT req for each branch.
        - if Nr or Ni,
            - set Ncnt to Nsubnb + Nsubnb.
        - if Ni, add one unit to Ncnt,
        - if Nlf, set Ncnt to 1.
    - if Nrpend,
        - issue a TP-ROLLBACK ind,
        - set Nt and Nfa to TRUE,

        Set the counter to owe a TP-DONE req and a rollback confirm from each subordinate.
        - set Ncnt to Nsubnb,
        - if Nr or Ni, add Nsubnb to Ncnt,
        - if Ni or Nlf, add one unit to Ncnt,
        - generate "Set-done-true".

[COUNTRB] (count rollback confirm event)
Counts a *rollback confirm* event and performs the actions associated with completing rollback when all events have been received. When all events have arrived, the synchronizing event to report rollback to the superior is generated. If this is a root node, the one-time actions necessary to begin the next transaction are done including generating the synchronizing event.
- subtract one unit from Ncnt.

Begin the next transaction for the root
- if Ncnt = 0 and Nr,
    - issue a TP-ROLLBACK-COMPLETE ind,
    - delNode (SnD, Naaid, Nbrid),
    - set Nfa, Np, Nt, Nrpend, and Nfrb to FALSE,
    - set Ncnt to Nsubnb + Nsubnb,
    - set Naaid to a unique value,
    - addNode (SnD, Naaid, Nbrid),
    - generate "Complete-rollback".

Initiate reporting of rollback to the superior if not a root
- if Ncnt = 0 and ^Nr,
    - generate "Report-rollback".

[COUNTRDY] (count ready event)
Counts the events needed to complete phase I of commitment, this includes a TP-COMMIT request to each branch and a C-READY indication from each subordinate. When all events have been received, does the one-time actions associated with completing phase I. For the root, makes the commitment decision and sets the counter to account for receiving a TP-DONE request and a *commit confirm* from each subordinate. For the intermediate node, enters the READY state and generates the synchronizing event to propagate the ready.
- subtract one unit from Ncnt.

- if (Ncnt = 0) and Nr and ^Ldfail,

    Set the counter to owe a TP-DONE req and a *commit confirm* from each subordinate.
    - set Ncnt to Nsubnb + Nsubnb,
    - issue a TP-COMMIT ind,
    - begin setting the *TPPM bound data* to the final state, unless a heuristic decision was taken; the *TPPM bound data* shall eventually be set to the final state; when this occurs is a local matter,
    - set SnD (Naaid, Nbrid).type to "log-commit",
    - set Naaidn to a new unique value,
    - set Nbridn to NULL,
    - addNode (SnD, Naaidn, Nbridn),
    - generate "Set-done-true",
    - generate "Continue-commit".

- if (Ncnt = 0) and ^Nr and ^Ldfail,

    NOTE — Setting the *TPPM bound data* to the ready-to-commit state may be done atomically with writing the log-ready record

    - set the *TPPM bound data* to the ready-to-commit state,
    - set SnD (Naaid, Nbrid).type to "log-ready",
    - generate "Enter-ready".

- if (Ncnt = 0) and Ldfail,
    - issue a TP-ROLLBACK ind,

    Set the counter to *owe a TP-DONE request* for each

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

branch and owe a *rollback confirm* from each subordinate.
- if Nr or Ni,
  - set Ncnt to Nsubnb + Nsubnb.
- if Ni, add one unit to Ncnt,
- if Nlf, set Ncnt to 1,

- set Nfrb and Nfa to TRUE,
- generate "Set-done-true",
- generate "Rollback-all".

[CPSAP] (close the PSAP)
- close the PSAP.

[DECDENB] (Decrement Denb)
Invoked to decrement Denb and also Denbb, whenever applicable.
- subtract one unit from Denb.
- if Denbb > 0,
  - subtract one unit from Denbb.

[DEFREQ] (send AF-DEFER request)
Issues an AF-DEFER request that has been held pending receipt of a TP-PREPARE request or a TP-COMMIT request from the TPSUI.
- if De,
  - issue an AF-DEFER (end-dialogue) req.
- if ^De and Dg,
  - issue an AF-DEFER (grant-control) req.

[DELBR] (delete branch)
Removes a transaction branch and adjusts the node variables accordingly.
- set Dl to FALSE.
- if Dsup,
  - set Nch to FALSE,
  - if Ni,
    - set Ni to FALSE,
    - set Nr to TRUE.
  - if Nlf,
    - set Nlf to FALSE.
- if ^Dsup,
  - subtract one unit from Nsubnb,
  - set Dch to FALSE,
  - if Nsubnb = 0 and Ni,
    - set Ni to FALSE,
    - set Nlf to TRUE.
  - if Nsubnb = 0 and Nr,
    - set Nr to FALSE.

Reduce the counter for the C-READY ind
- subtract one unit from Ncnt.

Reduce the counter for the TP-COMMIT req
- subtract one unit from Ncnt.

[DELBRANCH] (delete branch in system variable)
Removes a transaction branch from sbbr for a dialogue which is terminated during the ACTIVE state without causing a rollback.
- delBranch (SnD (Naaid, Nbrid).sbbr, Dbrid).

[DELIMIT] (delimit dialogue)
Handles the AF-BEGIN-DIALOGUE (accepted, dataRI) response which occurs before any requests are issued by the subordinate.
- if Dsup and ^Da,
  - issue an AF-BEGIN-DIALOGUE (accepted, dataRI) rsp,
  - set Ncr to FALSE,
  - set Da and Nrn to TRUE,
  - if Dl,
    - issue a C-BEGIN rsp.
  - while (Depnb > 0),
    - issue AF-U-ERROR rsp,
    - subtract one unit from Depnb.

[DIALOGUE] (operate on dialogue)
Executes all subsequent actions on the dialogue.

[INITMACF] (initialize MACF)
Invoked upon the first TP-BEGIN-DIALOGUE request from the root node.
- set all node variables to their initial values, as specified in A.4.2.4.

[INITDIASB] (initialize dialogue with subordinate)
Invoked when an AF-BEGIN-DIALOGUE request is issued.
- if Handshake functional unit is selected,
  - set Dh to TRUE.
- if Unchained Transactions functional unit is selected,
  - set Du to TRUE.
- if Shared Control functional unit is selected,
  - set Dsh to TRUE.
- set Nrn to TRUE.
- if (*confirmation* = "always"),
  - set Dcr to TRUE.

[INITDIASP] (initialize dialogue with superior)
Invoked when an AF-BEGIN-DIALOGUE indication is received.
- if Handshake functional unit is selected,
  - set Dh to TRUE.
- if Unchained Transactions functional unit is selected,
  - set Du to TRUE.
- if Shared Control functional unit is selected,
  - set Dsh and Dc to TRUE.
- set Dsup to TRUE.
- if (*confirmation* = "always"),
  - set Ncr to TRUE.

[INITRB] (initiate rollback)
Initiates rollback at this node. Sets the counter to account for a *rollback confirm/indication* from each subordinate.
- set the bound data to the initial state.
- set Nfrb to TRUE.
- set Ncnt to Nsubnb.
- set Nt to TRUE.
- set SnD (Naaid, Nbrid).type to NULL.
- generate "Rollback-all".

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

[LOGDAM] (log damage)
Updates the log-damage record according to the value of the *heuristic-report* parameter of either a TP-DONE request or an AF-[ABORT-AND-]HEURISTIC-REPORT indication.
- if the value of the heuristic-report parameter is "heuristic-hazard",
  - if ^memsp (SldD, Naaid, Nbrid),
    - addNode (SldD, Naaid, Nbrid),
    - set SldD (Naaid, Nbrid).type to "heuristic-hazard".

- if the value of the heuristic-report parameter is "heuristic-mix",
  - if ^memsp (SldD, Naaid, Nbrid),
    - addNode (SldD, Naaid, Nbrid),
    - set SldD (Naaid, Nbrid).type to "heuristic-mix".
  - if memsp (SldD, Naaid, Nbrid),
    - if (SldD (Naaid, Nbrid).type = "heuristic-hazard"),
      - set SldD (Naaid, Nbrid).type to "heuristic-mix".

[LOGDAMH] (**log damage hazard**)
Creates a the log-damage record with the value of "heuristic-hazard".
- if ^memsp (SldD, Naaid, Nbrid),
  - addNode (SldD, Naaid, Nbrid),
  - set SldD (Naaid, Nbrid).type to "heuristic-hazard".

[LOGDAMRB] (**log damage rollback**)
Updates the log-damage as necessary in the event of a rollback.
- if memsp (SlhD, Naaid, Nbrid),
  - if SlhD (Naaid, Nbrid).type ^= "heuristic-initial",
    - addNode (SldD, Naaid, Nbrid),
    - set SldD (Naaid, Nbrid).type to "heuristic-mix".

[LOGHD] (**log heuristic decision**)
Updates the log-heuristic record according to the value of the *heuristic-report* parameter from the heuristic decision.
- if ^memsp (SlhD, Naaid, Nbrid),
  - addNode (SlhD, Naaid, Nbrid),
  - set SlhD (Naaid, Nbrid).type to the *heuristic-report* parameter.
- if memsp (SlhD, Naaid, Nbrid),
  - if (SlhD (Naaid, Nbrid).type ^= *heuristic-report*),
    - set SlhD (Naaid, Nbrid).type to "heuristic-mix".

[LOGREMOVE] (**log-heuristic,log-damage remove**)
Removes the log-damage and log-heuristic records.
- delNode (SldD, Naaid, Nbrid).
- if memsp (SlhD, Naaid, Nbrid),
  - delNode (SlhD, Naaid, Nbrid).

[NOTCHAIN] (**not chaining**)
Invoked when the dialogue is no longer *chaining transaction branches.*
- if Dsup, set Nch to FALSE.
- if ^Dsup, set Dch to FALSE.

[NEXTAAID] (**next atomic action identifier**)
Sets the atomic action identifier to the new value for the next transaction after commitment.
- set Naaid to Naaidn.
- set Nbrid to Nbridn.

[NXTTRAN] (**next transaction**)
Does the one time actions for a leaf or intermediate node when rollback is complete and the next transaction after a rollback is to begin.
- issue a TP-ROLLBACK-COMPLETE ind.
- set Nfa, Np, Nt, Nrpend and Nfrb to FALSE.
- if Ni,
  - set Ncnt to Nsubnb + Nsubnb,
  - add one unit to Ncnt.
- if Nlf, set Ncnt to 1.
- if ^Nch,
  - delNode (SnD, Naaid, Nbrid),
  - set Naaid to a new unique value,
  - set Nbrid to NULL,
  - addNode (SnD, Naaid, Nbrid).
- generate "Complete-rollback".

[OLDCHANNEL] (operate on **old channel**)
Executes all subsequent actions in this subcell on the channel that was already attached to the TPPM before a CAF-RECOVER indication was received on a different channel.

[OPSAP] (**open PSAP**)
- open the PSAP.

[OWEDONE] (**owe** a TP-**DONE** request).
Makes a *TP-DONE request owed* and allows failure related actions. If a *TP-DONE request is not owed,* adjust the count of events expected.
- set Nfa to TRUE.
- if ^Dd,
  - generate "Set-done-true",
  - if ^Drbrep,
    - if Nr or Ni, add Nsubnb to Ncnt,
    - if Ni or Nlf, add one unit to Ncnt.

[OWEDONECO] (**owe** a TP-**DONE** request after **commit** indication) and adjust the count of events expected.
Makes a *TP-DONE request owed .*
- generate "Set-done-true".
- if Nr or Ni, add Nsubnb to Ncnt.
- if Ni or Nlf, add one unit to Ncnt.

[PREPREQ] (issue AF-**PREP**ARE **req**)
- if Dsh,
  - issue an AF-PREPARE req.
- if ^Dsh,
  - issue an AF-PREPARE (data-permitted = FALSE) req.

ICONDM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

[RBNEXTSB] (rollback next transaction subordinate)
Invoked when it is determined that the next transaction will rollback and the dialogue is chaining.
- if ^Nrpend,
  - set Nrpend to TRUE,
  - generate "Rollback-next-trans".

[RBREQ] (issue rollback request)
Invoked to issue a *rollback request* of the correct form.
- if ^Dtb and (^Dsup or (^memsp (SldD, Naaid, Nbrid) and Dsup)),
  - issue C-ROLLBACK req.
- if Dtb and (^Dsup or (^memsp (SldD, Naaid, Nbrid) and Dsup)),
  - issue an AF-ABORT (user, rollbackRI) req.
- if Dsup and memsp (SldD, Naaid, Nbrid) and ^Dtb,
  - set *heuristic-report* parameter to SldD (Naaid, Nbrid).type,
  - issue an AF-HEURISTIC-REPORT (rollbackRI) req.
- if Dsup and memsp (SldD, Naaid, Nbrid) and Dtb,
  - set *heuristic-report* parameter to SldD (Naaid, Nbrid).type,
  - issue an AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) req.
- if Dtb,
  - set Dbpart to TRUE,
  - set Dtb to FALSE.

[RBRSPNOAB] (issue rollback response no abort)
Invoked to issue a *rollback response* if no abort is included. Danyb is FALSE when this procedure is invoked.
- if ^Dsup or (^memsp (SldD, Naaid, Nbrid) and Dsup),
  - issue C-ROLLBACK rsp.
- if Dsup and memsp (SldD, Naaid, Nbrid),
  - set *heuristic-report* parameter to SldD (Naaid, Nbrid).type,
  - issue an AF-HEURISTIC-REPORT (rollbackRC) req.

[RBRSPAB] (issue rollback response with abort)
Invoked to issue a *rollback response* together with a user abort. Either Dtb or Dbpart is TRUE when this procedure is invoked.
- if ^Dsup or (^memsp (SldD, Naaid, Nbrid) and Dsup),
  - issue an AF-ABORT (user, rollbackRC) req.
- if Dsup and memsp (SldD, Naaid, Nbrid),
  - set *heuristic-report* parameter to SldD (Naaid, Nbrid).type,
  - issue an AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) req.

[RECCOM] (receive commit indication)
Receives the *commit indication*. Sets the counter to account for receiving a *commit confirm* from each subordinate.
- set Ncnt to Nsubnb.
- issue a TP-COMMIT ind.
- begin setting the *TPPM bound data* to the final state, unless a heuristic decision has been taken; the *TPPM bound data* shall eventually be set to the final state; when this occurs is a local matter.
- if Nch,
  - set Naaidn to the *atomic-action-identifier* parameter,
  - set Nbridn to the *atomic-action-branch-identifier* parameter,
  - addNode (SnD, Naaidn, Nbridn),
  - set SnD (Naaidn, Nbridn).spbr.rch to Arrh.
- if ^Nch,
  - set Naaidn to a new unique value,
  - set Nbridn to NULL,
  - addNode (SnD, Naaidn, Nbridn).
- if memsp (SlhD, Naaid, Nbrid),
  - if (SlhD (Naaid, Nbrid).type ^= "heuristic-final"),
    - addNode (SldD, Naaid, Nbrid),
    - set SldD (Naaid, Nbrid).type to "heuristic-mix".
- generate "Continue-commit".

[RECVRCOMI]
(issue C-RECOVER (commit) request)
Invoked to issue the correct type of C-RECOVER (commit) request to the subordinate.
- set the *atomic-action-identifier* parameter to Naaid.
- set the *atomic-action-branch-identifier* parameter to Dbrid.
- if SnD (Naaid, Nbrid).sbbr (Dbrid).rch = NULL,
  - issue C-RECOVER (commit) req.
- if SnD (Naaid, Nbrid).sbbr (Dbrid).rch ^= NULL,
  - set the *recovery-context-handle* parameter to SnD (Naaid, Nbrid).sbbr (Dbrid).rch,
  - issue an AF-RECOVER (commit) req.
- if Atwr,
  - issue an AF-TOKEN-GIVE (two-way-recovery) req.

[RECVRCOMR]
(issue C-RECOVER (commit) request after CAF-RECOVER (ready) ind)
Invoked to issue C-RECOVER (commit) request to the subordinate when a CAF-RECOVER (ready) indication was received.
- set the *atomic-action-identifier* parameter to Naaid.
- set the *atomic-action-branch-identifier* parameter to Dbrid.
- issue a C-RECOVER (commit) req.
- if Atwr,
  - issue an AF-TOKEN-GIVE (two-way-recovery) req.

[RECVRDONE]
(issue C-RECOVER (done) response)
Invoked to issue the correct type of C-RECOVER (done) response to the superior.
- set the *atomic-action-identifier* parameter to Naaid.
- set the *atomic-action-branch-identifier* parameter to Nbrid.
- if ^memsp (SldD, Naaid, Nbrid),
  - issue C-RECOVER (done) rsp.
- if memsp (SldD, Naaid, Nbrid),
  - set the heuristic-report parameter to SldD (Naaid, Nbrid).type,
  - issue an AF-HEURISTIC-REPORT (recoverDoneRC) req.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

[RECVRRDY] (issue C-RECOVER (ready) request)
Invoked to issue the correct type of C-RECOVER (ready) request to the superior.
- set the *atomic-action-identifier* parameter to Naaid.
- set the *atomic-action-branch-identifier* parameter to Nbrid.
- if SnD (Naaid, Nbrid).spbr.rch = NULL,
  - issue C-RECOVER (ready) req.
- if SnD (Naaid, Nbrid).spbr.rch ^= NULL,
  - set the *recovery-context-handle* parameter to SnD (Naaid, Nbrid).spbr.rch,
  - issue an AF-RECOVER (ready) req.
- if Atwr,
  - issue an AF-TOKEN-GIVE (two-way-recovery) req.

[RESETD] (reset dialogue variables)
Resets dialogue variables for the next transaction.
- set Dfdone, Dd, Ddp, De, Dg, Ddef and Drbrep to FALSE.
- set Denb, Denbb and Depnb to zero.
- set Dps and Dx to FALSE.
- if Du,
  - set Dbcr to FALSE.

[SETAAID] (set TPPM atomic-action and atomic-action-branch identifiers)
Sets the *atomic-action identifier* and *atomic-action-branch identifier* required to find the TPPM which requested the channel.
- set Caaid to the *atomic-action-identifier* parameter.
- set Cbrid to the *atomic-action-branch-identifier* parameter.

[SETDIAG] (set diagnostic)
Sets the *diagnostic* parameter of the next AF- or TP-service primitive issued by the TPPM.
- if this is a protocol error,
  - set the *diagnostic* parameter to "protocol-error".
- if this is an internal error,
  - if Ldperm,
    - set the *diagnostic* parameter to "permanent-failure".
  - if ^Ldperm,
    - set the *diagnostic* parameter to "transient-failure".

[SETDIAGBD] (set diagnostic on AF-BEGIN-DIALOGUE response)
- if the Dialogue functional unit is selected, set the *diagnostic* parameter to, as appropriate, one of:
  - "recipient-tpsu-title-unknown"
  - "tpsu-not-available (permanent)"
  - "tpsu-not-available (transient)"
  - "recipient-tpsu-title-required"
  - "functional-unit-not-supported"
  - "functional-unit-combination-not-supported"
  - "no-reason-given"
- if the Recovery functional unit is selected, set the *diagnostic* parameter to, as appropriate, one of:
  - "functional-unit-not-supported"
  - "tppm-recovery-not-available"
  - "two-way-recovery-not-supported"
  - "no-reason-given"
- if the *diagnostic* parameter is set to "functional-unit-not-supported",
  - set the *functional-units* parameter to the functional units that are supported.

[SETDIAGTP] (set diagnostic on TP-P-ABORT-indication)
- if this is a Protocol error,
  - set the *diagnostic* parameter to "protocol-error".
- if this is an Internal error,
  - if Ldperm,
    - set the *diagnostic* parameter to "permanent-failure".
  - if ^Ldperm,
    - set the *diagnostic* parameter to "transient-failure".
- if this is an A-RELEASE rsp or A-RELEASE cnf,
  - set the *diagnostic* parameter to "permanent-failure".
- if this is an A-ABORT ind,
  - set the *diagnostic* parameter to "permanent-failure".
- if this is an A-P-ABORT ind,
  - set the *diagnostic* parameter to "permanent-failure".
- if this is an A-ABORT req,
  - set the *diagnostic* parameter to "permanent-failure".
- if this is a CAF-RECOVER (ready) req or a CAF-RECOVER (commit) req,
  - set the *diagnostic* parameter to "permanent-failure".

[SETTOKX] (set Atokx to TRUE)
Sets Atokx to TRUE after a C-RECOVER indication or a CAF-RECOVER indication has been received on a two-way-recovery channel.
- if Atwr,
  - set Atokx to TRUE.

A.4.4.5 Actions after node crash

After a node crash, action REBUILDTPPMS is executed.

[REBUILDTPPMS] (Rebuild TPPMs after node crash)
Creates one TPPM per each element included in SnD after a node crash has occured.
- for each element of SnD where type ^= NULL,
  - create a new MACF and for this new MACF do,
    - set Naaid to aaid,
    - set Nbrid to spbr.brid,
    - set Nt to TRUE,

  create branch to superior
  - if Nbrid ^= NULL,
    - add a new state machine and for this new branch do,
      - set Dsup to TRUE,
      - set Db, Dl, and Danyb to TRUE,
      - if type = "log-commit",
        - set Dd to TRUE,
        - set state to 21.5.
      - if type = "log-ready",
        - set state to 20.3.

  create branches to subordinates
  - for each element of sbbr,

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

- add one unit to Nsubnb,
- add a new state machine and for this new branch do,
  - set Dbrid to brid,
  - set Db, Dl, and Danyb to TRUE,
  - if type = "log-commit",
    - set Dd to TRUE,
    - set state to 21.1.
  - if type = "log-ready",
    - set state to 20.3.

- if Nbrid = NULL,
  - set Nr to TRUE.
- if Nbrid ^= NULL and Nsubnb = 0,
  - set Nlf to TRUE.
- if Nbrid ^= NULL and Nsubnb ^= 0,
  - set Ni to TRUE.
- if type = "log-commit",

  Count commit confirms from each subordinate
  - set Ncnt to Nsubnb,

  Count TP-DONE requests (like OWEDONE)
  - if Nr or Ni,
    - add Nsubnb to Ncnt.
  - if Ni or Nlf,
    - add one unit to Ncnt.
  - issue a TP-COMMIT ind.
- generate "Restart-TPPM".

A.5 TPASE

A.5.1 TPASE states

There is no state defined for the TPASE.

A.5.2 TPASE variables

There is no variable defined for the TPASE.

A.5.3 TPASE events

There is no Internal event and no synchronizing event defined for the TPASE.

A.5.4 TPASE actions

The following actions are defined for the TPASE:

Dec
- decode the TP APDU.
- issue the AF service primitive corresponding to the received TP APDU to the SACF.

MapA (Map to A-ABORT service)
- encode the TP APDU.
- issue an A-ABORT request, with the TP APDU carried by the *user data* parameter.

MapC (Map to CCR)
- encode the TP APDU.
- issue the CCR req/rsp specified by the *mapping* parameter, with the TP APDU carried by the *user data* parameter.

MapPd (Map to P-DATA service)
- encode the TP APDU.
- issue a P-DATA request, with the TP APDU carried by the *user data* parameter.

NOTE — The actual Presentation service which will be used to carry the APDU is determined according to the rules contained in 9.5 "Mapping" and 10.7 "Concatenation".

MapPp (Map to P-TOKEN-PLEASE service)
- encode the TP APDU.
- issue a P-TOKEN-PLEASE request, with the TP APDU carried by the *user data* parameter.

MapPg (Map to P-TOKEN-GIVE service)
- encode the TP APDU.
- issue a P-TOKEN-GIVE request, with the TP APDU carried by the *user data* parameter.

MapS (Map to SACF)
- decode the TP APDU carried by the *user data* parameter.
- set the *mapping* parameter to the value corresponding to the received CCR ind/cnf or A-ABORT ind.
- issue the AF service primitive corresponding to the received TP APDU to the SACF.

Table A.8 lists the actions taken for each event received by the TPASE.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.8 (1 of 2) - TPASE actions

Event	Action
AF-BEGIN-DIALOGUE req	MapPd
TP-BEGIN-DIALOGUE-RI	Dec
AF-BEGIN-DIALOGUE rsp	MapPd
TP-BEGIN-DIALOGUE-RC	Dec
C-ROLLBACK (TP-BEGIN-DIALOGUE-RC) cnf	MapS
AF-BID req	MapPd
TP-BID-RI	Dec
AF-BID rsp	MapPd
TP-BID-RC	Dec
AF-END-DIALOGUE req	MapPd
TP-END-DIALOGUE-RI	Dec
AF-END-DIALOGUE rsp	MapPd
TP-END-DIALOGUE-RC	Dec
AF-U-ERROR req	MapPd
TP-U-ERROR-RI	Dec
AF-U-ERROR rsp	MapPd
TP-U-ERROR-RC	Dec
AF-ABORT (user, dataRI) req	MapPd
TP-ABORT-RI	Dec
AF-ABORT (provider, abortRI) req	MapA
A-ABORT (TP-ABORT-RI) ind	MapS
AF-GRANT-CONTROLreq	MapPd
TP-GRANT-CONTROL-RI	Dec
AF-REQUEST-CONTROL req	MapPd
TP-REQUEST-CONTROL-RI	Dec
AF-HANDSHAKE req	MapPd
TP-HANDSHAKE-RI	Dec
AF-HANDSHAKE rsp	MapPd
TP-HANDSHAKE-RC	Dec
AF-HANDSHAKE-AND-GRANT-CONTROL req	MapPd
TP-HANDSHAKE-AND-GRANT-CONTROL-RI	Dec
AF-HANDSHAKE-AND-GRANT-CONTROL rsp	MapPd
TP-HANDSHAKE-AND-GRANT-CONTROL-RC	Dec
AF-DEFER req	MapPd
TP-DEFER-RI	Dec
AF-PREPARE req	MapC
C-PREPARE (TP-PREPARE-RI) ind	MapS
AF-ABORT (user, commitRI) req	MapC
C-COMMIT (TP-ABORT-RI) ind	MapS
AF-ABORT (user, commitRC) req	MapC
C-COMMIT (TP-ABORT-RI) cnf	MapS
AF-HEURISTIC-REPORT (commitRC) req	MapC
C-COMMIT (TP-HEURISTIC-REPORT-RI) cnf	MapS
AF-ABORT-AND-HEURISTIC-REPORT (commitRC) req	MapC
C-COMMIT (TP-ABORT-RI, TP-HEURISTIC-REPORT-RI) cnf	MapS
AF-ABORT (user, rollbackRI) req	MapC
C-ROLLBACK (TP-ABORT-RI) ind	MapS
AF-HEURISTIC-REPORT (rollbackRI) req	MapC
C-ROLLBACK (TP-HEURISTIC-REPORT-RI) ind	MapS

Table A.8 (Concluded 2 of 2) - TPASE actions

Event	Action
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) req	MapC
C-ROLLBACK (TP-ABORT-RI, TP-HEURISTIC-REPORT-RI) ind	MapS
AF-ABORT (user, rollbackRC) req	MapC
C-ROLLBACK (TP-ABORT-RI) cnf	MapS
AF-HEURISTIC-REPORT (rollbackRC) req	MapC
C-ROLLBACK (TP-HEURISTIC-REPORT-RI) cnf	MapS
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) req	MapC
C-ROLLBACK (TP-ABORT-RI, TP-HEURISTIC-REPORT-RI) cnf	MapS
AF-TOKEN-GIVE req	MapPg
TP-TOKEN-GIVE-RI	Dec
AF-TOKEN-PLEASE req	MapPp
TP-TOKEN-PLEASE-RI	Dec

A.6 SACF

A.6.1 SACF states

The names for SACF states in the main text are shown after the corresponding state number.

State1 (FREE)
The SAO is in the FREE state.

State 1.1
This state is valid only for a contention-winner. An AF-BID (*token-requested* = TRUE) indication has been received. The *token* is not available and the SACF waits for the *token* before accepting or rejecting the bid.

State 1.2
This state is valid only for a contention-winner. An AF-BID indication has been received, and the TPPM accepted the bid. The SACF waits for an AF-BEGIN-DIALOGUE indication from the contention-loser.

State 2 (STRAY)
AF-BEGIN-DIALOGUE req has been issued.

State 3 (BIDDING)
An AF-BID request has been issued.

State 4 (BID CONFIRM RECEIVED)
An AF-BID (accepted) confirm has been received.

State 6 (BUSY)
One of the following situations has occured:

a) an AF-BEGIN-DIALOGUE indication has been received by a contention-loser, or has been received by a contention-winner while no AF-BEGIN-DIALOGUE req has been issued;

b) an AF-BEGIN-DIALOGUE confirm with a valid *correlator* parameter has been received; or

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

c) a C-ROLLBACK indication or confirm has been received.

State 7 (CLEANUP ROLLBACK INDICATION EXPECTED)
A superior has issued a C-BEGIN request and received a non confirmed dialogue termination AF indication.

State 8 (CLEANUP BEGIN INDICATION EXPECTED)
A subordinate provider has rejected a dialogue with coordination level COMMITMENT.

State 9 (CLEANUP ROLLBACK CONFIRM EXPECTED)
A superior has issued a C-ROLLBACK request and received a non confirmed dialogue termination AF indication.

A.6.2 SACF variables

A.6.2.1 Overview

Two categories of variables are defined for the SACF:

a) variables that pertain to an association. These variables are created at association establishment time, and are destroyed at association termination time. Association variables are prefixed by the letter "A". They are listed in table A.9; and

b) variables that model a decision local to the node, when there is a choice. Local decision variables are prefixed by the letter "L". They are listed in table A.10.

NOTE — An important subcategory are the A-variables shared with the MACF (see A.4.2.1 f).

Table A.9 - Association variables

Name	Meaning
Abm	bid mandatory
Abtr	begin-transaction-reject
Acbegq	C-BEGIN request received
Acopy	copy issued AF- service
Adc	dialogue correlator
Adru	dialogue rejected by user
Adt	SAF-DETACH-ASSOCIATION request received
Af	C-BEGIN fear
Alpi	last partner identifier
Anfd	not the first dialogue on association
Aq	queue
Arrh	received recovery-context-handle
Arvyrs	recovery response awaited
Atokr	token requested
Atokx	token expected (two-way-recovery only)
Atppm	attached to a TPPM
Atwr	two-way-recovery
Aw	contention-winner

Table A.10 - SACF local decision variables

Name	Meaning
Ldbid	decision to bid
Lddel	decision to delay bid rsp
Ldres	decision to reserve the association

A.6.2.2 Definitions of SACF variables

The following variables are defined for the SACF state table. SACF variables are Boolean variables, unless otherwise specified.

Abm (bid mandatory): when set to TRUE, Abm indicates that bidding is mandatory.

Abtr (begin-transaction-reject): when set to TRUE, Abtr indicates that an AF-ABORT (provider, begin-transaction-reject, rollbackRI) request has been received.

Acbegq (C-BEGIN request received): when set to TRUE, Acbegq indicates that a C-BEGIN request has been received and queued.

Acopy (copy issued AF- service): Acopy contains the issued AF- service, in the cases where either an AF-ABORT (rollbackRI) req or an AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) rsp has been issued.

Adc (dialogue correlator): Adc is a variable of type Integer. Adc is used to check whether an AF-BEGIN-DIALOGUE confirm is valid. The initiator sets Adc to a unique value and copies that value into the *correlator* parameter of the AF-BEGIN-DIALOGUE request. At the recipient, Adc takes the value of the *correlator* parameter provided by the AF-BEGIN-DIALOGUE indication. Upon sending an AF-BEGIN-DIALOGUE response, the *correlator* parameter carries the value kept by Adc.

Adru (dialogue rejected by user): when set to TRUE, Adru indicates that an AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) response has been received.

Adt (SAF-DETACH-ASSOCIATION request received): when set to TRUE, Adt indicates that a SAF-DETACH-ASSOCIATION request has been received.

Af (C-BEGIN fear): when set to TRUE, Af denotes that a stray C-BEGIN indication may be received.

Alpi (last partner identifier): Alpi is a variable of type Integer. For a contention-winner, Alpi indicates the valid value of the *correlator* parameter to be received during the next dialogue establishment. For a contention-loser, Alpi indicates the valid value of the *correlator* parameter to be sent during the next dialogue establishment.

Anfd (not the first dialogue on association): when set to TRUE, Anfd indicates that either an AF-BEGIN-DIALOGUE request has been issued by a contention-winner

ICSOFT.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

or that an AF-BEGIN-DIALOGUE indication has been received by a contention-loser.

Aq (queue): when set to TRUE, Aq indicates that a queue is established.

Arrh (received recovery-context-handle): Arrh is a variable of type Octet string. Arrh contains the recovery-context-handle received on the association.

Arvyrs (recovery response awaited): when set to TRUE, Arvyrs indicates that a C-RECOVER (ready) indication was received and a response is awaited.

Atokr (token requested): when set to TRUE, Atokr indicates that the contention-loser SACF has issued an AF-BID request with *token-requested* parameter set to TRUE, and has received an AF-BID (accepted) confirm, but has not yet received the *token*. Atokr is also used by the contention-winner to indicate that the token must be sent to the contention-loser when it arrives (after an AF-BEGIN-DIALOGUE indication without an AF-BID indication).

Atokx (token expected): When set to TRUE, Atokx indicates that a C-RECOVER indication or an AF-RECOVER indication has been received on a two-way-recovery channel and that the *token* is awaited.

Atppm (attached to a TPPM): when set to TRUE, Atppm indicates that the SACF is attached to a TPPM. When set to FALSE, Atppm indicates that the SACF is attached to a CPM.

Atwr (two-way-recovery): when set to TRUE, Atwr indicates that the channel is established in the two-way-recovery mode. When set to FALSE, this variable indicates that the channel is established in the one-way-recovery mode.

Aw (contention-winner): when set to TRUE, Aw indicates that the AEI is the contention-winner.

Ldbid (decision to bid): when set to TRUE, Ldbid indicates that the SACF will bid, although bidding is not mandatory.

Lddel (decision to delay bid rsp): when set to TRUE, Lddel indicates that the contention-winner that does not own the *token* will delay its response to an AF-BID (*token-requested* = TRUE) indication until receipt of a P-TOKEN-GIVE (sync-minor) indication.

Ldres (decision to reserve the association): when set to TRUE, Ldres indicates that the association is reserved by the contention-winner.

A.6.2.3 Initialization of SACF variables

SACF variables of type Boolean are initialized to FALSE, except for the following variables:

Abm: initialized to TRUE if the *Bid-Mandatory* parameter of the TP-INITIALIZE-RI APDU is set to TRUE.

Aw: initialized to TRUE if the the *Contention-winner-assignment* parameter is set to "initiator" upon sending the TP-INITIALIZE-RI APDU, or if the *Contention-winner-assignment* parameter is set to "recipient" upon receipt of the TP-INITIALIZE-RI PDU.

SACF variables of type Integer are initialized to zero.

SACF variables of type Octet string are initialized to EMPTY, except for the following variables:

Arrh: initialized to the value of the *Recovery-context-handle* parameter upon receipt of either the TP-INITIALIZE-RI, or TP-INITIALIZE-RC APDU. If the *Recovery-context-handle* parameter is not specified, Arrh is initialized to NULL.

A.6.3 SACF events

A.6.3.1 Internal events

Reject-bid
Abbreviation used for "Unsolicited BID reject", as defined in 10.6.1.

A.6.3.2 Synchronizing events

There is no synchronizing event defined for the SACF state table.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

A.6.4 SACF actions

A.6.4.1 Actions on services

Table A.11 lists actions that issue service primitives. These actions are named according to the following convention:

First character
A AF-
C C-
S SAF-

Next characters

AB ABORT
AL ASSOCIATION-LOST
BD BEGIN-DIALOGUE
BID BID
RB ROLLBACK
TOKG TOKEN-GIVE

Service Primitive type

rq request
i indication
rs response

Source parameter

P Provider
U User

Type parameter

A Accepted
R Rejected
RG Regular
KP Keep

Mapping parameter

d dataRI
rbc rollbackRC

Diagnostic parameter

AR association-reserved

The remaining characters qualify the action in a manner specific fo the service being issued.

NOTE — For example, ABIDrsA issues an AF-BID response with the Type parameter set to "accepted".

A.6.4.2 Actions on variables

For actions that manipulate SACF variables, the following conventions apply:

First character: V

The name of the variable being set begins at the second character.

The final characters are either:

CORR (set the *correlator* parameter);
F (set to FALSE);
NEW (set to a new unique value); or,
T (set to TRUE);

NOTE — An example is "VAbtrF", for "set Abtr to FALSE".

A.6.4.3 Actions with free-form names

[BIDREQ] (issue an AF-**BID** request)
Issues the correct type of AF-BID request.
- if Anfd, set *Last-Partner-Identifier* parameter to Alpi.
- if CFU, issue AF-BID (*token-requested* = TRUE) req.
- if ^CFU, issue AF-BID (*token-requested* = FALSE) req.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.11 - SACF actions on services

Action name	Parameter settings/service primitive issued
AABrqPa	AF-ABORT (provider, abortRI) req
ABDrsRPdAR	- set the *diagnostic* parameter to "association-reserved" AF-BEGIN-DIALOGUE (rejected(provider), dataRI) rsp
ABIDrsA	AF-BID (accepted) rsp
ABIDrsR	AF-BID (rejected) rsp
ATOKGrqKP	AF-TOKEN-GIVE (keep) req
ATOKGrqRG	AF-TOKEN-GIVE (regular) req
CRBrq	C-ROLLBACK req
CRBrs	C-ROLLBACK rsp
SALi	SAF-ASSOCIATION-LOST ind

[COPY] (**copy** issued service)
- copy the issued service to Acopy.

[ATTACHMACF] (**attach** to the **MACF**)
- if an AF-BEGIN-DIALOGUE (Dialogue fu selected) ind was received,
  - create a new TPPM MACF,
  - attach to the TPPM.
- if an AF-BEGIN-DIALOGUE (Recovery fu selected) ind was received,
  - attach to the CPM.

[DISCARDQ] (**discard** the **queue**)
- if Aq,
  - discard the queue.

[FLUSHALL] (**flush all** of the queue)
- flush the queue.

[FLUSHPAR] (**flush part** of the queue)
- flush the queue up to and excluding the C-BEGIN req.

[PASSTHRU] (**pass** the service primitive **through**)
- pass the service primitive through.

[PASSTOKEN]
(**pass** the **token** to the U-ASE)
- pass the received P-TOKEN-GIVE (sync-minor) ind to the U-ASE.

[QUEUE] (**queue** the service primitive)
- queue the service primitive received from the MACF.

[REPREQ] (**Repeat** AF- **request**)
- issue the primitive that is in Acopy with the mapping parameter changed into rollbackRC.

[RESETS]
- set Acbegq, Adt, Aq and Atokr to FALSE.

[RETTOKEN] (**return token**)
Returns the *token* to the contention winner.
- if Ptok and ^Aw,
  - issue AF-TOKEN-GIVE (regular) req.

[SETCORR]
(**set Correlator** parameter)
- set the *correlator* parameter to Adc.

[SETDIAG] (**set diagnostic**)
Sets the *diagnostic* parameter of the next AF- service primitive issued by the SACF.
- set the *diagnostic* parameter to "protocol-error".

[SETLPI] (**set** *Last-Partner-Identifier* parameter)
- if Anfd, set the *Last-Partner-Identifier* parameter to Alpi.

A.6.5 Notational conventions

In Table A.18, the following notational conventions are used:

CFU is a predicate which is TRUE if the functional units parameter of the service primitive in the respective event column contains the Commit functional unit or the Recovery functional unit.

DC is a predicate which is the value of the *correlator* parameter of the service primitive in the respective event column.

LPI is a predicate which is the value of the *Last-Partner-Identifier* parameter of the service primitive in the respective event column.

* is used instead of a state number if as a result of a state transition the association ceases to exist.

A.7 Predicates

In addition to those variables defined in A.4.2 and A.6.2 the state tables make use of predicates which reflect some state of the local system. These predicates do not have to be initialized nor modified by the PM. Table A.12 lists such predicates.

Table A.12 - Predicates

Name	Meaning
Pnew	new TPSUI
Ptok	token owned

Pnew (**new** TPSUI): when TRUE, Pnew indicates that the TPSUI is issuing its first TP-BEGIN-DIALOGUE request and that a new TPPM has been created and is to be initialized.

Ptok (**token** owned): when set to TRUE, Ptok indicates that the *token* is owned. This predicate is maintained outside of this International Standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 (*1 of 18*) — **Dialogue**

State / Predicates / Event	1 dialogue does not exist	1.1 C-BEGIN ind awaited Dsup	2 data transfer S.C. or P.C. w/ control	3 data transfer P.C. w/o control ^Dsh	4 AF-U-ERROR req issued S.C. or P.C. w/o control	5 AF-U-ERROR ind rcv'd P.C. w/ control ^Dsh	11 AF-END-DIALOGUE (conf=TRUE) req issued S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	12 AF-END-DIALOGUE (conf=TRUE) ind rcv'd S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	15 TP-PREPARE req issued C-READY ind awaited S.C. or P.C. w/ control DI, ^Dsup	20.1 TP-COMMIT req rcv'd C-READY ind awaited S.C. or P.C. w/ control ^Dcr DI, ^Dsup
TP-BEGIN-DIALOGUE (No transaction branch) req	^Ldrej, Pnew [INITMACF] [INITDIASB] [VNtpsuiT] [VAtppmT] [ABDrq] 2									
	^Ldrej, ^Pnew Ntpsui, ^Ncr [INITDIASB] [VAtppmT] [ABDrq] 2									
	Ldrej, Ldunk [TBDcRPu] 1									
	Ldrej, ^Ldunk [TBDcRP] 1									
TP-BEGIN-DIALOGUE (Transaction branch) req	^Ldrej, Pnew [INITMACF] [INITDIASB] [ADDBRSB] [VNtpsuiT] [VAtppmT] [ABDrq] [CBErq] 2									
	^Ldrej, ^Pnew Ntpsui, ^Ncr ^Nt [INITDIASB] [ADDBRSB] [VAtppmT] [ABDrq] [CBErq] 2									
	Ldrej, Ldunk [TBDcRPu] 1									
	Ldrej, ^Ldunk [TBDcRP] 1									
AF-BEGIN-DIALOGUE (Dialogue fu selected, Polarized Control fu selected, Commit fu selected, No transaction branch) ind	^Ldrej [INITDIASP] [VNtpsuiT] [VAtppmT] [TBDi] 3									
	Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqBF] 1									

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 2 of 18)* **— Dialogue**

State / Predicates / Event	1 dialogue does not exist	1.1 C-BEGIN ind awaited	2 data transfer	3 data transfer	4 AF-U-ERROR req issued	5 AF-U-ERROR ind rcv'd	11 AF-END-DIALOGUE (conf=TRUE) req issued	12 AF-END-DIALOGUE (conf=TRUE) ind rcv'd	15 TP-PREPARE req issued C-READY ind awaited	20.1 TP-COMMIT req rcv'd C-READY ind awaited
		Dsup	S.C. or P.C. w/ control	P.C. w/o control ^Dsh	S.C. or P.C. w/o control	P.C. w/ control ^Dsh	S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/ control DI, ^Dsup	S.C. or P.C. w/ control ^Dcr DI, ^Dsup
AF-BEGIN-DIALOGUE (Dialogue fu selected, Polarized Control fu selected, Commit fu not selected, No transaction branch) ind	^Ldrej [INITDIASP] [VNtpsuiT] [VAtppmT] [TBDi] 3 Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqF] 1									
AF-BEGIN-DIALOGUE (Dialogue fu selected, Shared Control fu selected, Commit fu selected, No transaction branch) ind	^Ldrej [INITDIASP] [VNtpsuiT] [VAtppmT] [TBDi] 2 Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqBF] 1									
AF-BEGIN-DIALOGUE (Dialogue fu selected, Shared Control fu selected, Commit fu not selected, No transaction branch) ind	^Ldrej [INITDIASP] [VNtpsuiT] [VAtppmT] [TBDi] 2 Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqF] 1									
AF-BEGIN-DIALOGUE (Dialogue fu selected, Polarized Control fu selected, Transaction branch) ind or AF-BEGIN-DIALOGUE (Dialogue fu selected, Shared Control fu selected, Transaction branch) ind	^Ldrej [INITDIASP] [VAtppmT] [VDbegdiSAVE] 1.1 Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqCB] 1									
TP-BEGIN-DIALOGUE (accepted) rsp			Dsup Ncr [DELIMIT] 2	Dsup Ncr [DELIMIT] 3		Dsup Ncr [DELIMIT] 5				

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 3 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
Event		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr ^DI	^Dcr or ^Ncr ^DI	DI, ^Dsup	^Dcr DI, ^Dsup
TP-BEGIN-DIALOGUE (rejected) rsp			^Du, ^DI, Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqF] 1	^Du, ^DI, Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqF] 1		^Du, ^DI, Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqF] 1		Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqF] 1		
			Du, ^DI, Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqBF] 1	Du, ^DI, Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqBF] 1		Du, ^DI, Dsup ^Nrn, ^Da [ABDrsRUd] [SDETrqBF] 1				
			DI, Dsup ^Nrn, ^Da [ABDrsRUr] [SDETrqRBC] [DELBR] 1	DI, Dsup ^Nrn, ^Da [ABDrsRUr] [SDETrqRBC] [DELBR] 1		DI, Dsup ^Nrn, ^Da [ABDrsRUr] [SDETrqRBC] [DELBR] 1				
AF-BEGIN-DIALOGUE (accepted, dataRI) cnf			^Dsup Dcr [TBDcX] [VDcrF] [VDaT] 2	^Dsup Dcr [TBDcX] [VDcrF] [VDaT] 3	^Dsup Dcr [TBDcX] [VDcrF] [VDaT] 4				^Dsup Dcr [TBDcX] [VDcrF] [VDaT] 15	
			^Dsup ^Dcr [VDaT] 2	^Dsup ^Dcr [VDaT] 3	^Dsup ^Dcr [VDaT] 4		^Dsup [VDaT] 11		^Dsup ^Dcr [VDaT] 15	[VDaT] 20.1
AF-BEGIN-DIALOGUE (rejected(provider), dataRI) cnf			^DI, ^Dsup [TBDcX] [SDETrqF] 1	^DI, ^Dsup [TBDcX] [SDETrqF] 1	^DI, ^Dsup [TBDcX] [SDETrqF] 1		^Dsup [TBDcX] [SDETrqF] 1			
			DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25				[TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	[TBDcXr] [SDETrqRB] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
AF-BEGIN-DIALOGUE (rejected(user), dataRI) cnf			^DI, ^Dsup [TBDcX] [SDETrqF] 1	^DI, ^Dsup [TBDcX] [SDETrqF] 1	^DI, ^Dsup [TBDcX] [SDETrqF] 1		^Dsup [TBDcX] [SDETrqF] 1			
			Du, DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	Du, DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	Du, DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25				Du [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	Du [TBDcXr] [SDETrqRB] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 4 of 18)* **— Dialogue**

State / Predicates / Event	1 dialogue does not exist	1.1 C-BEGIN ind awaited Dsup	2 data transfer S.C. or P.C. w/ control	3 data transfer P.C. w/o control ^Dsh	4 AF-U-ERROR req issued S.C. or P.C. w/o control	5 AF-U-ERROR ind rcv'd P.C. w/ control ^Dsh	11 AF-END-DIALOGUE (conf=TRUE) req issued S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	12 AF-END-DIALOGUE (conf=TRUE) ind rcv'd S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	15 TP-PREPARE req issued C-READY ind awaited S.C. or P.C. w/ control DI, ^Dsup	20.1 TP-COMMIT req rcv'd C-READY ind awaited S.C. or P.C. w/ control ^Dcr DI, ^Dsup
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) cnf			DI, ^Dsup [TBDcX] [CRBrs] [SDETrqF] [ABDET] [DELBRANCH] 25	DI, ^Dsup [TBDcX] [CRBrs] [SDETrqF] [ABDET] [DELBRANCH] 25	DI, ^Dsup [TBDcX] [CRBrs] [SDETrqF] [ABDET] [DELBRANCH] 25				[TBDcX] [CRBrs] [SDETrqF] [ABDET] [DELBRANCH] 25	[TBDcXr] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
SAF-ASSOCIATION-LOST ind			^DI [TBDcRP] 1 DI [TBDcRP] [ABDET] [DELBRANCH] 25	^DI [TBDcRP] 1 DI [TBDcRP] [ABDET] [DELBRANCH] 25	^DI [TBDcRP] 1 DI [TBDcRP] [ABDET] [DELBRANCH] 25		[TBDcRP] 1		[TBDcRP] [ABDET] [DELBRANCH] 25	[TBDcRPr] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
TP-END-DIALOGUE (confirmation = FALSE) req			^Du, ^DI, Dsup ^Ncr [DELIMIT] [AEDrq] [SDETrqF] 1 Du, ^DI, Dsup ^Ncr [DELIMIT] [AEDrq] [SDETrqBF] 1 ^DI, ^Dsup ^Dcr [AEDrq] [SDETrqF] 1		Dsh, ^Du, ^DI, Dsup [AEDrq] [SDETrqF] 1 Dsh, Du, ^DI, Dsup [AEDrq] [SDETrqBF] 1 Dsh, ^DI, ^Dsup ^Dcr [AEDrq] [SDETrqF] 1				
TP-END-DIALOGUE (confirmation = TRUE) req			^DI, Dsup ^Ncr [DELIMIT] [AEDrq] 11 ^DI, ^Dsup ^Dcr [AEDrq] 11		Dsh, ^DI, Dsup ^Ncr [AEDrq] 11 Dsh, ^DI, ^Dsup ^Dcr [AEDrq] 11					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 5 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/ control	S.C. or P.C. w/ control ^Dcr
Event		Dsup		^Dsh		^Dsh			DI, ^Dsup	DI, ^Dsup
AF-END-DIALOGUE (confirmation = FALSE) ind			Dsh, ^DI, Dsup ^Ncr [TEDi] [SDETrqF] 1	^DI, Dsup ^Ncr [TEDi] [SDETrqF] 1	^DI, Dsup ^Ncr [TEDi] [SDETrqF] 1		Dsh [TEDi] [SDETrqF] 1			
			Dsh, ^DI, ^Dsup ^Dcr [TEDi] [SDETrqF] 1	^DI, ^Dsup ^Dcr [TEDi] [SDETrqF] 1	^DI, ^Dsup ^Dcr [TEDi] [SDETrqF] 1					
			Dsh, Dx [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25		Dsh, Dx [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25				Dsh, Dx [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25	Dsh, Dx [TPABiBTEDr] [SDETrqRB] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2
AF-END-DIALOGUE (confirmation = TRUE) ind			Dsh, ^DI, Dsup ^Ncr [TEDi] 12	^DI, Dsup ^Ncr [TEDi] 12	Dsh, ^DI Denb=1 [DECDENB] 2		Dsh Denb=0 [TPABiED] [SDETrqF] 1		Dsh, Dx Denbb=0 [TPABiBTED] [SDETrqRB] [ABDET]	Dsh, Dx Denbb=0 [TPABiBTEDr] [SDETrqRB] [ABDET]
			Dsh, ^DI, ^Dsup ^Dcr [TEDi] 12	^DI, ^Dsup ^Dcr [TEDi] 12	Dsh, ^DI Denb>1 [DECDENB] 4		Dsh Denb>0 [DECDENB] 11		[DELBRANCH] 25	[INITRB] [OWEDONE] [COUNTRB] 23.2
			Dsh, Dx [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25		Dsh, Dx Denbb=0 [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25				Du, Dsh ^Dcr, ^Dbcr Denbb>0 [DECDENB] 15	Du, Dsh ^Dbcr Denbb>0 [DECDENB] 20.1
					Dsh, Dx Denbb>0 [DECDENB] 4					
					^Dsh, ^DI [TEDi] 2					
TP-END-DIALOGUE rsp								[DELIMIT] [AEDrs] [SDETrqF] 1		
AF-END-DIALOGUE cnf							[TEDc] [SDETrqF] 1			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 6 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/ control	S.C. or P.C. w/ control ^Dcr
Event		Dsup		^Dsh		^Dsh			DI, ^Dsup	DI, ^Dsup
TP-U-ERROR req			Dsh, Dsup ^Ncr [DELIMIT] [AUErq] [VDenbINC] 4		Dsh, Dsup [AUErq] [VDenbINC] 4			[DELIMIT] [AUErq] 2		
			Dsh, ^Dsup [AUErq] [VDenbINC] 4		Dsh, ^Dsup [AUErq] [VDenbINC] 4					
			^Dsh, Dsup ^Ncr [DELIMIT] [AUErq] 2	Dsup ^Ncr [DELIMIT] [AUErq] 4						
			^Dsh, ^Dsup [AUErq] 2	^Dsup [AUErq] 4						
AF-U-ERROR ind			Dsh, Dsup ^Da [TUEi] [VDepnbINC] 2		Dsh, Dsup [TUEi] [AUErs] 4		Dsh Denb=0 [TUEi] 2			
			Dsh, Dsup Da [TUEi] [AUErs] 2				Dsh Denb>0 [TUEi] 4			
			Dsh, ^Dsup ^Dcr [TUEi] [AUErs] 2		Dsh, ^Dsup ^Dcr [TUEi] [AUErs] 4					
			^Dsh, Dsup [TUEi] 5	Dsup [TUEi] 3	^Dsh, Dsup 4		^Dsh [TUEi]			
			^Dsh, ^Dsup ^Dcr [TUEi] 5	^Dsup ^Dcr [TUEi] 3	^Dsh, ^Dsup ^Dcr 4		3		^Dcr [TRBi] [INITRB] [OWEDONE] 15	[TRBi] [INITRB] [OWEDONE] 20.1
AF-U-ERROR cnf					Dsh Denb=1 [DECDENB] 2		Dsh Denb>0 [DECDENB] 11		Dsh Denb>0 [DECDENB] 15	Dsh Denb>0 [DECDENB] 20.1
					Dsh Denb>1 [DECDENB] 4					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 7 of 18)* **— Dialogue**

State / Predicates / Event	1 dialogue does not exist	1.1 C-BEGIN ind awaited Dsup	2 data transfer S.C. or P.C. w/ control	3 data transfer P.C. w/o control ^Dsh	4 AF-U-ERROR req issued S.C. or P.C. w/o control	5 AF-U-ERROR ind rcv'd P.C. w/ control ^Dsh	11 AF-END-DIALOGUE (conf=TRUE) req issued S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	12 AF-END-DIALOGUE (conf=TRUE) ind rcv'd S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	15 TP-PREPARE req issued C-READY ind awaited S.C. or P.C. w/ control DI, ^Dsup	20.1 TP-COMMIT req rcv'd C-READY ind awaited S.C. or P.C. w/ control ^Dcr DI, ^Dsup
TP-U-ABORT req			^Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqF] 1	^Du, Dsup [AABrqUd] [SDETrqF] 1	^Du, Dsup [DELIMIT] [AABrqUd] [SDETrqF] 1		
			Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqBF] 1	Du, Dsup [AABrqUd] [SDETrqBF] 1	Du, Dsup [DELIMIT] [AABrqUd] [SDETrqBF] 1		
			^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^Dsup [AABrqUd] [SDETrqF] 1	^Dsup [AABrqUd] [SDETrqF] 1		
			DI, Dsup ^Ncr [DELIMIT] [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 2	DI, Dsup ^Ncr [DELIMIT] [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 3	DI, Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 4	DI, Dsup ^Ncr [DELIMIT] [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 5				
			DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 2	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 3	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 4	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 5			[ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 15	
AF-ABORT (user, dataRI) ind			^DI, Dsup [TUABi] [SDETrqF] 1	^DI, Dsup [TUABi] [SDETrqF] 1	^DI, Dsup [TUABi] [SDETrqF] 1	^DI [TUABi] [SDETrqF] 1	[TUABi] [SDETrqF] 1	[TUABi] [SDETrqF] 1		
			^DI, ^Dsup ^Dcr [TUABi] [SDETrqF] 1	^DI, ^Dsup ^Dcr [TUABi] [SDETrqF] 1	^DI, ^Dsup ^Dcr [TUABi] [SDETrqF] 1					
			DI, ^Dsup ^Dcr, ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	DI, ^Dsup ^Dcr, ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	DI, ^Dsup ^Dcr, ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	DI, ^Dsup ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25			^Dcr, ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	^Dbcr [TUABiRI] [SDETrqRB] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 8 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
Event		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr	^Dcr or ^Ncr		^Dcr
							^DI	^DI	DI, ^Dsup	DI, ^Dsup
AF-ABORT (provider, abortRI) ind or A-ABORT ind or A-ABORT req or A-P-ABORT ind or A-RELEASE (result = affirmative) rsp or A-RELEASE (result = affirmative) cnf			^DI [SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1	[SETDIAGTP] [TPABi] 1	[SETDIAGTP] [TPABi] 1		
		1	DI, Dsup Ncr [SETDIAGTP] [TPABi] 1	DI, Dsup Ncr [SETDIAGTP] [TPABi] 1		DI, Dsup Ncr [SETDIAGTP] [TPABi] 1				
			DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8				
			DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2			[SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	[SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 9 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control ^Dsh	S.C. or P.C. w/o control	P.C. w/ control ^Dsh	S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/ control DI, ^Dsup	S.C. or P.C. w/ control ^Dcr DI, ^Dsup
Event		Dsup								
Protocol error or Internal error		[SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	[SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	[SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1		
			DI, Dsup Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	DI, Dsup Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1		DI, Dsup Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1				
			DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8				
			DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2			[SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	[SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
TP-GRANT-CONTROL req			^Dsh, Dsup ^Ncr [DELIMIT] [AGCrq] 3			Dsup ^Ncr [DELIMIT] [AGCrq] 3				
			^Dsh, ^Dsup [AGCrq] 3			^Dsup [AGCrq] 3				
AF-GRANT-CONTROL ind				Dsup [TGCi] 2	^Dsh, Dsup [TGCi] 2					
				^Dsup ^Dcr [TGCi] 2	^Dsh, ^Dsup ^Dcr [TGCi] 2					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 10 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/ control	S.C. or P.C. w/ control ^Dcr
Event		Dsup		^Dsh		^Dsh			DI, ^Dsup	DI, ^Dsup
TP-REQUEST-CONTROL req				Dsup ^Ncr [DELIMIT] [ARCrq] 3 ^Dsup [ARCrq] 3						
AF-REQUEST-CONTROL ind			^Dsh, Dsup [TRCi] 2 ^Dsh, ^Dsup ^Dcr [TRCi] 2	3	^Dsh 4		^Dsh 11		^Dsh ^Dcr 15	^Dsh 20.1
TP-HANDSHAKE req			Dh, Dsup ^Ncr [DELIMIT] [AHSrq] 6 Dh, ^Dsup [AHSrq] 6		Dh, Dsh, Dsup [AHSrq] 6 Dh, Dsh, ^Dsup [AHSrq] 6					
AF-HANDSHAKE ind			Dh, Dsh, Dsup [THSi] 7 Dh, Dsh, ^Dsup ^Dcr [THSi] 7	Dh, Dsup [THSi] 7 Dh, ^Dsup ^Dcr [THSi] 7	Dh, Dsh, Dsup Denb=1 [DECDENB] 2 Dh, Dsh, Dsup Denb>1 [DECDENB] 4 Dh, ^Dsh, Dsup [THSi] 2 Dh, Dsh, ^Dsup ^Dcr Denb=1 [DECDENB] 2 Dh, Dsh, ^Dsup ^Dcr Denb>1 [DECDENB] 4 Dh, ^Dsh, ^Dsup ^Dcr [THSi] 2		Dh, Dsh Denb=0 [THSi] 10 Dh, Dsh Denb>0 [DECDENB] 11		Dsh ^Dcr [TRBi] [INITRB] [OWEDONE] 15	Dsh [TRBi] [INITRB] [OWEDONE] 20.1

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 11 of 18)* **— Dialogue**

State / Predicates / Event	1 dialogue does not exist	1.1 C-BEGIN ind awaited	2 data transfer	3 data transfer	4 AF-U-ERROR req issued	5 AF-U-ERROR ind rcv'd	11 AF-END-DIALOGUE (conf=TRUE) req issued	12 AF-END-DIALOGUE (conf=TRUE) ind rcv'd	15 TP-PREPARE req issued C-READY ind awaited	20.1 TP-COMMIT req rcv'd C-READY ind awaited
			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr ^DI	^Dcr or ^Ncr ^DI	DI, ^Dsup	^Dcr DI, ^Dsup
TP-HANDSHAKE-AND-GRANT-CONTROL req			Dh, ^Dsh, Dsup ^Ncr [DELIMIT] [AHSGCrq] 13 Dh, ^Dsh, ^Dsup [AHSGCrq] 13							
AF-HANDSHAKE-AND-GRANT-CONTROL ind				Dh, Dsup [THSGCi] 14 Dh, ^Dsup ^Dcr [THSGCi] 14	Dh, ^Dsh, Dsup [THSGCi] 2 Dh, ^Dsh, ^Dsup ^Dcr [THSGCi] 2					
TP-BEGIN-TRANSACTION req			Du, ^DI, ^Dsup ^Nt, ^Da [ADDBRSB] [BEGTRANS] [CBErq] 2 Du, ^DI, ^Dsup ^Nt, Ptok, Da [ADDBRSB] [BEGTRANS] [CBErq] 2		Du, Dsh, ^DI, ^Dsup ^Nt, ^Da [ADDBRSB] [BEGTRANS] [CBErq] 4 Du, Dsh, ^DI, ^Dsup ^Nt, Ptok, Da [ADDBRSB] [BEGTRANS] [CBErq] 4					
C-BEGIN ind		Dsh [TBDiSAVE] [ADDBRSP] [VNtpsuiT] 2 ^Dsh [TBDiSAVE] [ADDBRSP] [VNtpsuiT] 3	Du, Dsh, ^DI, Dsup Nr [TPABiBTR] [AABrqPrTR] [SDETrqRBC] 1 Du, Dsh, ^DI, Dsup ^Nr [TBTi] [ADDBRSP] 2	Du, ^DI, Dsup Nr [TPABiBTR] [AABrqPrTR] [SDETrqRBC] 1 Du, ^DI, Dsup ^Nr [TBTi] [ADDBRSP] 3	Du, ^DI, Dsup Nr [TPABiBTR] [AABrqPrTR] [SDETrqRBC] 1 Du, ^DI, Dsup ^Nr [TBTi] [ADDBRSP] 4		Du, Dsh, Dsup [TPABiBTED] [CRBrq] [SDETrqRBC] 1			
C-BEGIN cnf			DI, ^Dsup [VDbcrT] [VDxF] 2	DI, ^Dsup [VDbcrT] [VDxF] 3	DI, ^Dsup [VDbcrT] [VDxF] 4	DI, ^Dsup [VDbcrT] [VDxF] 5			[VDbcrT] [VDxF] 15	[VDbcrT] [VDxF] 20.1
TP-DATA req			Dsup ^Ncr [DELIMIT] [UASErq] 2 ^Dsup [UASErq] 2		Dsh, Dsup [UASErq] 4 Dsh, ^Dsup [UASErq] 4					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 12 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates		S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	S.C. or P.C. w/ control	S.C. or P.C. w/ control ^Dcr	
Event		Dsup	^Dsh		^Dsh			DI, ^Dsup	DI, ^Dsup	
U-ASE ind			Dsh, Dsup [TDTi] 2	Dsup [TDTi] 3			Dsh Denb=0 [TDTi] 11			
			Dsh, ^Dsup ^Dcr [TDTi] 2	^Dsup ^Dcr [TDTi] 3	4		Dsh Denb>0 11		Dsh ^Dcr [TDTi] 15	Dsh [TRBi] [INITRB] [OWEDONE] 20.1
									^Dsh ^Dcr, Ddp [TDTi] 15	^Dsh Ddp [TRBi] [INITRB] [OWEDONE] 20.1
TP-DEFERRED-END-DIALOGUE req			DI, ^Dsup ^De ^Lddef [ADErq] [VDeT] 2		Dsh, DI, ^Dsup ^De ^Lddef [ADErq] [VDeT] 4					
			DI, ^Dsup ^De Lddef [VDdefT] [VDeT] 2		Dsh, DI, ^Dsup ^De Lddef [VDdefT] [VDeT] 4					
TP-DEFERRED-GRANT-CONTROL req			^Dsh, DI, ^Dsup ^Dg, ^De ^Lddef [ADGrq] [VDgT] 2							
			^Dsh, DI, ^Dsup ^Dg, ^De Lddef [VDdefT] [VDgT] 2							
AF-DEFER (end-dialogue) ind			Dsh, DI, Dsup ^De [TDEi] [VDeT] 2	DI, Dsup ^De [TDEi] [VDeT] 3	DI, Dsup ^De [TDEi] [VDeT] 4					
AF-DEFER (grant-control) ind				DI, Dsup ^Dg, ^De [TDGi] [VDgT] 3	^Dsh, DI, Dsup ^Dg, ^De [TDGi] [VDgT] 4					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 13 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
Event		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr ^DI	^Dcr or ^Ncr ^DI	DI, ^Dsup	^Dcr DI, ^Dsup
TP-PREPARE req			Dsh, DI, Nr ^Ddef [APRrq] [VDpsT] 15		Dsh, DI, Nr ^Ddef [APRrq] [VDpsT] 15					
			Dsh, DI, Nr Ddef [DEFREQ] [APRrq] [VDpsT] 15		Dsh, DI, Nr Ddef [DEFREQ] [APRrq] [VDpsT] 15					
			Dsh, DI, ^Dsup Ni, Np ^Ddef [APRrq] [VDpsT] 15		Dsh, DI, ^Dsup Ni, Np ^Ddef [APRrq] [VDpsT] 15					
			Dsh, DI, ^Dsup Ni, Np Ddef [DEFREQ] [APRrq] [VDpsT] 15		Dsh, DI, ^Dsup Ni, Np Ddef [DEFREQ] [APRrq] [VDpsT] 15					
TP-PREPARE (data-permitted = FALSE) req			^Dsh, DI, Nr ^Ddef [APRrq] [VDpsT] 15							
			^Dsh, DI, Nr Ddef [DEFREQ] [APRrq] [VDpsT] 15							
			^Dsh, DI, ^Dsup Ni, Np ^Ddef [APRrq] [VDpsT] 15							
			^Dsh, DI, ^Dsup Ni, Np Ddef [DEFREQ] [APRrq] [VDpsT] 15							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 14 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
Event		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr ^Dl	^Dcr or ^Ncr ^Dl	Dl, ^Dsup	^Dcr Dl, ^Dsup
TP-PREPARE (data-permitted = TRUE) req			^Dsh, Dl, Nr ^Ddef [APRrq] [VDdpT] [VDpsT] 15							
			^Dsh, Dl, Nr Ddef [DEFREQ] [APRrq] [VDdpT] [VDpsT] 15							
			^Dsh, Dl, ^Dsup Ni, Np ^Ddef [APRrq] [VDdpT] [VDpsT] 15							
			^Dsh, Dl, ^Dsup Ni, Np Ddef [DEFREQ] [APRrq] [VDdpT] [VDpsT] 15							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 15 of 18)* **— Dialogue**

State / Predicates / Event	1 dialogue does not exist	1.1 C-BEGIN ind awaited Dsup	2 data transfer S.C. or P.C. w/ control	3 data transfer P.C. w/o control ^Dsh	4 AF-U-ERROR req issued S.C. or P.C. w/o control	5 AF-U-ERROR ind rcv'd P.C. w/ control ^Dsh	11 AF-END-DIALOGUE (conf=TRUE) req issued S.C. or P.C. w/ control ^Dcr or ^Ncr ^DI	12 AF-END-DIALOGUE (conf=TRUE) ind rcv'd S.C. or P.C. w/o control ^Dcr or ^Ncr ^DI	15 TP-PREPARE req issued C-READY ind awaited S.C. or P.C. w/ control DI, ^Dsup	20.1 TP-COMMIT req rcv'd C-READY ind awaited S.C. or P.C. w/ control ^Dcr DI, ^Dsup
TP-COMMIT req			DI, Nr ^Dcr, ^Ddef [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1		Dsh, DI, Nr ^Dcr, ^Ddef [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1				^Dcr [COUNTRDY] [VNtT] 20.1	
			DI, Nr ^Dcr, Ddef [DEFREQ] [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1		Dsh, DI, Nr ^Dcr, Ddef [DEFREQ] [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1					
			DI, ^Dsup Ni, Np ^Dcr, ^Ddef [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1		Dsh, DI, ^Dsup Ni, Np ^Dcr, ^Ddef [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1					
			DI, ^Dsup Ni, Np ^Dcr, Ddef [DEFREQ] [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1		Dsh, DI, ^Dsup Ni, Np ^Dcr, Ddef [DEFREQ] [PREPREQ] [COUNTRDY] [VNtT] [VDpsT] 20.1					
AF-PREPARE ind			Dsh, DI, Dsup [TPRi] [VNpT] 18		Dsh, DI, Dsup [TRBi] [INITRB] [OWEDONE] 4					
AF-PREPARE (data-permitted = FALSE) ind				DI, Dsup [TPRi] [VNpT] 18	^Dsh, DI, Dsup [TRBi] [INITRB] [OWEDONE] 4					
AF-PREPARE (data-permitted = TRUE) ind				DI, Dsup [TPRi] [VNpT] [VDdpT] 18	^Dsh, DI, Dsup [TRBi] [INITRB] [OWEDONE] 4					
C-READY ind									^Dcr [TRYi] [COUNTRDY] 17	[COUNTRDY] 20.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 16 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
Event		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr ^DI	^Dcr or ^Ncr ^DI	DI, ^Dsup	^Dcr DI, ^Dsup
TP-ROLLBACK req			DI, Dsup ^Ncr, ^Nfrb [INITRB] [OWEDONE] [VDaT] 23.3	DI, Dsup ^Ncr, ^Nfrb [INITRB] [OWEDONE] [VDaT] 23.3	DI, Dsup ^Nfrb [INITRB] [OWEDONE] 23.3	DI, Dsup ^Ncr, ^Nfrb [INITRB] [OWEDONE] [VDaT] 23.3				
			DI, Dsup ^Ncr, Nfrb [VDaT] 23.3	DI, Dsup ^Ncr, Nfrb [VDaT] 23.3	DI, Dsup Nfrb 23.3	DI, Dsup ^Ncr, Nfrb [VDaT] 23.3				
			DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1			^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	
			Nfrb, DI, ^Dsup [RBREQ] 23.1	Nfrb, DI, ^Dsup [RBREQ] 23.1	Nfrb, DI, ^Dsup [RBREQ] 23.1	Nfrb, DI, ^Dsup [RBREQ] 23.1			Nfrb [RBREQ] 23.1	
C-ROLLBACK ind			DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4				
			^Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2			^Du ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2
			Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2			Du ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(continued 17 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control ^Dcr or ^Ncr	S.C. or P.C. w/o control ^Dcr or ^Ncr	S.C. or P.C. w/ control	S.C. or P.C. w/ control ^Dcr
Event		Dsup		^Dsh		^Dsh	^DI	^DI	DI, ^Dsup	DI, ^Dsup
AF-HEURISTIC-REPORT (rollbackRI) ind									^Du, ^Dcr [TRBi] [THRi] [LOGDAM] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du [TRBi] [THRi] [LOGDAM] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2
									Du, ^Dcr [TRBi] [THRi] [LOGDAM] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du [TRBi] [THRi] [LOGDAM] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2
AF-ABORT (provider, rollbackRI) ind			Du, DI, ^Dsup ^Dbcr [TPABiR] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du, DI, ^Dsup ^Dbcr [TPABiR] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du, DI, ^Dsup ^Dbcr [TPABiR] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du, DI, ^Dsup ^Dbcr [TPABiR] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2			Du ^Dbcr [TPABiR] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du ^Dbcr [TPABiR] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2
AF-ABORT (user, rollbackRI) ind			DI, Dsup ^Ncr [TUABiR] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4	DI, Dsup ^Ncr [TUABiR] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4	DI, Dsup [TUABiR] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4	DI, Dsup ^Ncr [TUABiR] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4				
			DI, Dsup Ncr [TUABi] [CRBrs] [SDETrqF] [DELBR] 1	DI, Dsup Ncr [TUABi] [CRBrs] [SDETrqF] [DELBR] 1		DI, Dsup Ncr [TUABi] [CRBrs] [SDETrqF] [DELBR] 1				
			DI, ^Dsup ^Dcr [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup ^Dcr [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup ^Dcr [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2			^Dcr [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	[TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.13 *(concluded 18 of 18)* **— Dialogue**

State	1	1.1	2	3	4	5	11	12	15	20.1
	dialogue does not exist	C-BEGIN ind awaited	data transfer	data transfer	AF-U-ERROR req issued	AF-U-ERROR ind rcv'd	AF-END-DIALOGUE (conf=TRUE) req issued	AF-END-DIALOGUE (conf=TRUE) ind rcv'd	TP-PREPARE req issued C-READY ind awaited	TP-COMMIT req rcv'd C-READY ind awaited
Predicates			S.C. or P.C. w/ control	P.C. w/o control	S.C. or P.C. w/o control	P.C. w/ control	S.C. or P.C. w/ control	S.C. or P.C. w/o control	S.C. or P.C. w/ control	S.C. or P.C. w/ control
Event		Dsup		^Dsh		^Dsh	^Dcr or ^Ncr ^Dl	^Dcr or ^Ncr ^Dl	Dl, ^Dsup	^Dcr Dl, ^Dsup
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) ind									^Dcr [TUABiR] [THRi] [LOGDAM] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	[TUABiR] [THRi] [LOGDAM] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
CAF-RECOVER (ready) ind									[CREnsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [TPABiR] [THRiH] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	[CREnsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [TPABiR] [THRiH] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
Rollback-by-TPPM			Dl, Dsup ^Ncr, ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3	Dl, Dsup ^Ncr, ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3	Dl, Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3	Dl, Dsup ^Ncr, ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3				
			Dl, Dsup ^Ncr, Nfrb 23.3	Dl, Dsup ^Ncr, Nfrb 23.3	Dl, Dsup Nfrb 23.3	Dl, Dsup ^Ncr, Nfrb 23.3				
			Dl, ^Dsup ^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1	Dl, ^Dsup ^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1	Dl, ^Dsup ^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1	Dl, ^Dsup ^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1			^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1	^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1
			Dl, ^Dsup Nfrb [RBREQ] 23.1	Dl, ^Dsup Nfrb [RBREQ] 23.1	Dl, ^Dsup Nfrb [RBREQ] 23.1	Dl, ^Dsup Nfrb [RBREQ] 23.1			Nfrb [RBREQ] 23.1	Nfrb [RBREQ] 23.1
Rollback-all			Dsup 23.3	Dsup 23.3	Dsup 23.3	Dsup 23.3				
			^Dsup [RBREQ] 23.1	^Dsup [RBREQ] 23.1	^Dsup [RBREQ] 23.1	^Dsup [RBREQ] 23.1			[RBREQ] 23.1	[RBREQ] 23.1

End of table A.13

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*1 of 10*) — Handshake

State / Predicates / Event	6 AF-HANDSHAKE req issued P.C. or S.C.	7 AF-HANDSHAKE ind rcv'd P.C. or S.C.	8 AF-HANDSHAKE ind received on state 6, or req issued on st. 7 S.C.	9 AF-END-DIALOGUE ind rcv'd on state 6 S.C. ^DI	10 AF-HANDSHAKE ind rcv'd on state 11 S.C. ^DI	13 AF-HANDSHAKE &-GRANT-CTL req issued P.C.	14 AF-HANDSHAKE &-GRANT-CTL ind rcv'd P.C.
TP-BEGIN-DIALOGUE (accepted) rsp		Dsup Ncr [DELIMIT] 7					Dsup Ncr [DELIMIT] 14
TP-BEGIN-DIALOGUE (rejected) rsp		^Du, ^DI, Dsup ^Nrm, ^Da [ABDrsRUd] [SDETrqF] 1					^Du, ^DI, Dsup ^Nrm, ^Da [ABDrsRUd] [SDETrqF] 1
		Du, ^DI, Dsup ^Nrm, ^Da [ABDrsRUd] [SDETrqBF] 1					Du, ^DI, Dsup ^Nrm, ^Da [ABDrsRUd] [SDETrqBF] 1
		DI, Dsup ^Nrm, ^Da [ABDrsRUr] [SDETrqRBC] [DELBR] 1					DI, Dsup ^Nrm, ^Da [ABDrsRUr] [SDETrqRBC] [DELBR] 1
AF-BEGIN-DIALOGUE (accepted, dataRI) cnf	^Dsup Dcr [TBDcX] [VDcrF] [VDaT] 6					^Dsup Dcr [TBDcX] [VDcrF] [VDaT] 13	
	^Dsup ^Dcr [VDaT] 6					^Dsup ^Dcr [VDaT] 13	
AF-BEGIN-DIALOGUE (rejected(provider), dataRI) cnf	^DI, ^Dsup [TBDcX] [SDETrqF] 1					^DI, ^Dsup [TBDcX] [SDETrqF] 1	
	DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25					DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	
AF-BEGIN-DIALOGUE (rejected(user), dataRI) cnf	^DI, ^Dsup [TBDcX] [SDETrqF] 1					^DI, ^Dsup [TBDcX] [SDETrqF] 1	
	Du, DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25					Du, DI, ^Dsup [TBDcX] [SDETrqRB] [ABDET] [DELBRANCH] 25	
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) cnf	DI, ^Dsup [TBDcX] [CRBrs] [SDETrqF] [ABDET] [DELBRANCH] 25					DI, ^Dsup [TBDcX] [CRBrs] [SDETrqF] [ABDET] [DELBRANCH] 25	

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 2 of 10*) — **Handshake**

State / Predicates / Event	6 AF-HANDSHAKE req issued P.C. or S.C.	7 AF-HANDSHAKE ind rcv'd P.C. or S.C.	8 AF-HANDSHAKE ind received on state 6, or req issued on st. 7 S.C.	9 AF-END-DIALOGUE ind rcv'd on state 6 S.C. ^DI	10 AF-HANDSHAKE ind rcv'd on state 11 S.C. ^DI	13 AF-HANDSHAKE &-GRANT-CTL req issued P.C.	14 AF-HANDSHAKE &-GRANT-CTL ind rcv'd P.C.
SAF-ASSOCIATION-LOST ind	^DI [TBDcRP] 1					^DI [TBDcRP] 1	
	DI [TBDcRP] [ABDET] [DELBRANCH] 25					DI [TBDcRP] [ABDET] [DELBRANCH] 25	
AF-END-DIALOGUE (confirmation = FALSE) ind	Dsh, ^DI, Dsup [TEDi] [SDETrqF] 1						
	Dsh, ^DI, ^Dsup ^Dcr [TEDi] [SDETrqF] 1						
	Dsh, Dx [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25						
AF-END-DIALOGUE (confirmation = TRUE) ind	Dsh, ^DI, Dsup Denb=0 [TEDi] 9						
	Dsh, ^DI, Dsup Denb>0 [DECDENB] 6						
	Dsh, ^DI, ^Dsup ^Dcr Denb=0 [TEDi] 9						
	Dsh, ^DI, ^Dsup ^Dcr Denb>0 [DECDENB] 6						
	Dsh, Dx Denbb=0 [TPABiBTED] [SDETrqRB] [ABDET] [DELBRANCH] 25						
	Dsh, Dx Denbb>0 [DECDENB] 6						

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 3 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
TP-U-ERROR req	Dsh, Dsup [AUErq] [VDenbINC] 6	Dsup ^Ncr [DELIMIT] [AUErq] 2	[AUErq]	[AUErq]	[AUErq]		Dsup ^Ncr [DELIMIT] [AUErq] 2
	Dsh, ^Dsup [AUErq] [VDenbINC] 6	^Dsup [AUErq] 2	6	6	11		^Dsup [AUErq] 2
AF-U-ERROR ind	Dsh, Dsup Denb=0 [TUEi] 2	Dsh ^Da [TUEi] [VDepnbINC] 7					
	Dsh, Dsup Denb>0 [TUEi] 4						
	Dsh, ^Dsup ^Dcr Denb=0 [TUEi] 2	Dsh Da [TUEi] [AUErs] 7	[TUEi] 7	[TUEi] 12	[TUEi] 7		
	Dsh, ^Dsup ^Dcr Denb>0 [TUEi] 4						
	^Dsh, Dsup [TUEi] 3					Dsup [TUEi] 3	
	^Dsh, ^Dsup ^Dcr [TUEi] 3					^Dsup ^Dcr [TUEi] 3	
AF-U-ERROR cnf	Dsh, Denb>0 [DECDENB] 6						

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 4 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 8	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
TP-U-ABORT req	^Du, ^DI, Dsup [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup [AABrqUd] [SDETrqF] 1	^Du, Dsup [AABrqUd] [SDETrqF] 1	^Du, Dsup [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup [AABrqUd] [SDETrqF] 1	^Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqF] 1
	Du, ^DI, Dsup [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup [AABrqUd] [SDETrqBF] 1	Du, Dsup [AABrqUd] [SDETrqBF] 1	Du, Dsup [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup [AABrqUd] [SDETrqBF] 1	Du, ^DI, Dsup ^Ncr [DELIMIT] [AABrqUd] [SDETrqBF] 1
	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^Dsup [AABrqUd] [SDETrqF] 1	^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1	^DI, ^Dsup [AABrqUd] [SDETrqF] 1
	DI, Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 6	DI, Dsup ^Ncr [DELIMIT] [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 7	DI, Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 8			DI, Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 13	DI, Dsup ^Ncr [DELIMIT] [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 14
	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 6	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 7	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 8			DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 13	DI, ^Dsup [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 14
AF-ABORT (user, dataRi) ind	^DI, Dsup [TUABi] [SDETrqF] 1	^DI [TUABi] [SDETrqF] 1	^DI [TUABi] [SDETrqF] 1	[TUABi] [SDETrqF] 1	[TUABi] [SDETrqF] 1	^DI, Dsup [TUABi] [SDETrqF] 1	^DI [TUABi] [SDETrqF] 1
	^DI, ^Dsup ^Dcr [TUABi] [SDETrqF] 1					^DI, ^Dsup ^Dcr [TUABi] [SDETrqF] 1	
	DI, ^Dsup ^Dcr, ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	Du, DI, ^Dsup ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	Du, DI, ^Dsup ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25			DI, ^Dsup ^Dcr, ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25	Du, DI, ^Dsup ^Dbcr [TUABi] [SDETrqRB] [ABDET] [DELBRANCH] 25

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 5 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
AF-ABORT (provider, abortRI) ind or A-ABORT ind or A-ABORT req or A-P-ABORT ind or A-RELEASE (result = affirmative) rsp or A-RELEASE (result = affirmative) cnf	^DI [SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1	[SETDIAGTP] [TPABi] 1	[SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1	^DI [SETDIAGTP] [TPABi] 1
		DI, Dsup Ncr [SETDIAGTP] [TPABi] 1					DI, Dsup Ncr [SETDIAGTP] [TPABi] 1
	DI, Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8			DI, Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8
	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2			DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 6 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
Protocol error or Internal error	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	[SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	[SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1	^DI [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1
		DI, Dsup Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1					DI, Dsup Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] 1
	DI, Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8			DI, Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	DI, Dsup ^Ncr [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8
	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2			DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [SETDIAGTP] [TPABiR] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
TP-REQUEST-CONTROL req		^Dsh, Dsup ^Ncr [DELIMIT] [ARCrq] 7					
		^Dsh, ^Dsup [ARCrq] 7					
AF-REQUEST-CONTROL ind	^Dsh [TRCi] 6					13	
TP-HANDSHAKE req		Dsh, Dsup ^Ncr [DELIMIT] [AHSrq] 8					
		Dsh, ^Dsup [AHSrq] 8					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 7 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
AF-HANDSHAKE ind	Dsh, Dsup Denb=0 [THSi] 8 Dsh, Dsup Denb>0 [DECDENB] 6 Dsh, ^Dsup ^Dcr Denb=0 [THSi] 8 Dsh, ^Dsup ^Dcr Denb>0 [DECDENB] 6						
TP-HANDSHAKE rsp		Dsh, Dsup ^Ncr [DELIMIT] [AHSrs] 2 Dsh, ^Dsup [AHSrs] 2 ^Dsh, Dsup ^Ncr [DELIMIT] [AHSrs] 3 ^Dsh, ^Dsup [AHSrs] 3	[AHSrs] 6		[AHSrs] 11		
AF-HANDSHAKE cnf	Dsh Denb>0 [THSc] 4 Dsh Denb=0 [THSc] 2 ^Dsh [THSc] 2		[THSc] 7	[THSc] 12			
TP-HANDSHAKE-AND-GRANT-CONTROL rsp							Dsup ^Ncr [DELIMIT] [AHSGCrs] 2 ^Dsup [AHSGCrs] 2
AF-HANDSHAKE-AND-GRANT-CONTROL cnf						[THSGCc] 3	

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 8 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
C-BEGIN ind	Du, Dsh, ^DI, Dsup Nr [TPABiBTR] [AABrqPrTR] [SDETrqRBC] 1 Du, Dsh, ^DI, Dsup ^Nr [ADDBRSP] [TBTi] 6						
C-BEGIN cnf	DI, ^Dsup [VDbcrT] [VDxF] 6	DI, ^Dsup [VDbcrT] [VDxF] 7	DI, ^Dsup [VDbcrT] [VDxF] 8			DI, ^Dsup [VDbcrT] [VDxF] 13	
TP-DATA req		Dsh, Dsup ^Ncr [DELIMIT] [UASErq] 7 Dsh, ^Dsup [UASErq] 7					
U-ASE ind	Dsh, Dsup Denb=0 [TDTi] 6 Dsh ^Dsup Denb=0 [TDTi] 6 Dsh Denb>0 6						
AF-DEFER (end-dialogue) ind	Dsh, DI, Dsup ^De [TDEi] [VDeT] 6						
AF-PREPARE ind	Dsh, DI, Dsup [TRBi] [INITRB] [OWEDONE] 6						

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*continued 9 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
TP-ROLLBACK req	DI, Dsup ^Nfrb [INITRB] [OWEDONE] 23.3	DI, Dsup ^Ncr, ^Nfrb [INITRB] [OWEDONE] [VDaT] 23.3	DI, Dsup ^Nfrb [INITRB] [OWEDONE] 23.3			DI, Dsup ^Nfrb [INITRB] [OWEDONE] 23.3	DI, Dsup ^Ncr, ^Nfrb [INITRB] [OWEDONE] [VDaT] 23.3
	DI, Dsup Nfrb 23.3	DI, Dsup ^Ncr, Nfrb [VDaT] 23.3	DI, Dsup Nfrb 23.3			DI, Dsup Nfrb 23.3	DI, Dsup ^Ncr, Nfrb [VDaT] 23.3
	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1			DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	DI, ^Dsup ^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1
	DI, ^Dsup Nfrb [RBREQ] 23.1	DI, ^Dsup Nfrb [RBREQ] 23.1	DI, ^Dsup Nfrb [RBREQ] 23.1			DI, ^Dsup Nfrb [RBREQ] 23.1	DI, ^Dsup Nfrb [RBREQ] 23.1
C-ROLLBACK ind	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4			DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4	DI, Dsup [TRBi] [INITRB] [OWEDONE] 23.4
	^Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2			^Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] 23.2
	Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2			Du, DI, ^Dsup ^Dcr [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2	Du, DI, ^Dsup [TRBi] [CRBrs] [INITRB] [OWEDONE] [COUNTRB] [CPSAP] 23.2
AF-ABORT (provider, rollbackRI) ind	Du, DI, ^Dsup ^Dbcr [TPABiRI] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du, DI, ^Dsup ^Dbcr [TPABiRI] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du, DI, ^Dsup ^Dbcr [TPABiRI] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2			Du, DI, ^Dsup ^Dbcr [TPABiRI] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2	Du, DI, ^Dsup ^Dbcr [TPABiRI] [CRBrs] [SDETrqF] [ABDET] [INITRB] [OWEDONE] [COUNTRB] 23.2

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.14 (*concluded 10 of 10*) — **Handshake**

State	6	7	8	9	10	13	14
	AF-HANDSHAKE req issued	AF-HANDSHAKE ind rcv'd	AF-HANDSHAKE ind received on state 6, or req issued on st. 7	AF-END-DIALOGUE ind rcv'd on state 6	AF-HANDSHAKE ind rcv'd on state 11	AF-HANDSHAKE &-GRANT-CTL req issued	AF-HANDSHAKE &-GRANT-CTL ind rcv'd
Predicates	P.C. or S.C.	P.C. or S.C.	S.C.	S.C. ^DI	S.C. ^DI	P.C.	P.C.
Event							
AF-ABORT (user, rollbackRI) ind	DI, Dsup [TUABiR] [INITRB] [OWEDONE] [ABPTNR] [NOTCHAIN] 23.4	DI, Dsup ^Ncr [TUABiR] [INITRB] [OWEDONE] [ABPTNR] [NOTCHAIN] 23.4	DI, Dsup [TUABiR] [INITRB] [OWEDONE] [ABPTNR] [NOTCHAIN] 23.4			DI, Dsup [TUABiR] [INITRB] [OWEDONE] [ABPTNR] [NOTCHAIN] 23.4	DI, Dsup ^Ncr [TUABiR] [INITRB] [OWEDONE] [ABPTNR] [NOTCHAIN] 23.4
		DI, Dsup Ncr [TUABi] [CRBrs] [SDETrqF] [DELBR] 1					DI, Dsup Ncr [TUABi] [CRBrs] [SDETrqF] [DELBR] 1
	DI, ^Dsup ^Dcr [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2			DI, ^Dsup ^Dcr [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	DI, ^Dsup [TUABiR] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2
Rollback-by-TPPM	DI, Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3	DI, Dsup ^Ncr, ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3	DI, Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3			DI, Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3	DI, Dsup ^Ncr, ^Nfrb [TRBi] [INITRB] [OWEDONE] 23.3
	DI, Dsup Nfrb 23.3	DI, Dsup ^Ncr, Nfrb 23.3	DI, Dsup Nfrb 23.3			DI, Dsup Nfrb 23.3	DI, Dsup ^Ncr, Nfrb 23.3
	DI, ^Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] [RBREQ] 23.1	DI, ^Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] [RBREQ] 23.1	DI, ^Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] [RBREQ] 23.1			DI, ^Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] [RBREQ] 23.1	DI, ^Dsup ^Nfrb [TRBi] [INITRB] [OWEDONE] [RBREQ] 23.1
	DI, ^Dsup Nfrb [RBREQ] 23.1	DI, ^Dsup Nfrb [RBREQ] 23.1	DI, ^Dsup Nfrb [RBREQ] 23.1			DI, ^Dsup Nfrb [RBREQ] 23.1	DI, ^Dsup Nfrb [RBREQ] 23.1
Rollback-all	Dsup 23.3	Dsup 23.3	Dsup 23.3			Dsup 23.3	Dsup 23.3
	^Dsup [RBREQ] 23.1	^Dsup [RBREQ] 23.1	^Dsup [RBREQ] 23.1			^Dsup [RBREQ] 23.1	^Dsup [RBREQ] 23.1

End of table A.14

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*1 of 17*) — Commitment

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
Predicates	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Event	DI ^Dsup	DI Dsup	DI	DI	DI ^Dsup	DI ^Dsup, Dch	DI ^Dsup	DI ^Dsup, Dch	DI Dsup	DI Dsup, Nch	DI ^Dsup	DI ^Dchat
TP-BEGIN-DIALOGUE (accepted) rsp		Ncr [DELIMIT] 18										
TP-BEGIN-DIALOGUE (rejected) rsp		^Nrn, ^Da [ABDrsRUr] [SDETrqRBC] [DELBR] 1										
TP-U-ABORT req	[ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 17	^Ncr [DELIMIT] [ABTPSUI] [NOTCHAIN] [INITRB] [OWEDONE] 18		^Danyb Ni, Nfa [ABTPSUI] 20.3	^Danyb Nfa, Dch [ABTPSUI] [RBNEXTSB] 21.1 ^Danyb Nfa, ^Dch [ABTPSUI] 21.1	^Danyb Nfa [ABTPSUI] 21.2	^Danyb Nfa, Dch [ABTPSUI] [RBNEXTSB] 21.3 ^Danyb Nfa, ^Dch [AABrqUd] [ABDET] [SDETrqF] 21.3	^Danyb Nfa [ABTPSUI] 21.4	^Danyb Nfa, Nch [ABTPSUI] [RBNEXTSB] 21.5 ^Danyb Nfa, ^Nch [ABTPSUI] 21.5	^Danyb Nfa [ABTPSUI] 21.6		

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 2 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
AF-ABORT (provider, abortRi) ind or A-P-ABORT ind or A-ABORT ind or A-ABORT req or A-RELEASE (result = affirmative) rsp or A-RELEASE (result = affirmative) cnf		[SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] 23.8	Dsup, ^Danyb [SETDIAGTP] [TPABi] [ABDET] [NOTCHAIN] [VNfaT] [CAFPLrqSP] 99	^Danyb, ^Dch [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [CAFPLrqSB] 99				^Danyb, ^Nch [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] 21.5	^Danyb [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] 21.5		
				Dsup, Danyb ^Db [ABDET] [NOTCHAIN] [CAFPLrqSP] 99					Danyb ^Db [ABDET] 21.5	Danyb [ABDET] 21.5		
				Dsup, Dchat [VDchatF] [NOTCHAIN] [CAFPLrqSP] 99					Dchat [VDchatF] 21.5			
	[SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2		^Dsup [SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Dsup, ^Danyb [SETDIAGTP] [TPABi] [ABDET] [NOTCHAIN] [VNfaT] [CAFPLrqSP] 20.3	^Danyb, Dch [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [RBNEXTSB] [CAFPLrqSB] 99	[SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [CAFPLrqSB] 99			^Danyb, Nch [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [RBNEXTSB] 21.5			
				^Dsup, Danyb ^Db [ABDET] [NOTCHAIN] 20.3	Danyb ^Db [ABDET] [CAFPLrqSB] 99							
				^Dsup, Dchat [VDchatF] [NOTCHAIN] 20.3	Dchat [VDchatF] [CAFPLrqSB] 99							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 3 of 17*) **— Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI ^Dsup	DI Dsup	DI	DI	DI ^Dsup	DI ^Dsup, Dch	DI ^Dsup	DI ^Dsup, Dch	DI Dsup	DI Dsup, Nch	DI ^Dsup	DI ^Dchat
Event												
Protocol error or Internal error		[SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [INITRB] [OWEDONE] 23.8	Dsup [SETDIAGTP] [TPABiR] [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [INITRB] [OWEDONE] 23.8	Dsup, ^Danyb [SETDIAGTP] [TPABi] [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [VNfaT] [CAFPLrqSP] 99	^Danyb, ^Dch [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] [CAFPLrqSB] 99		^Danyb, ^Dch [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] 21.3	[SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] 21.3	^Danyb, ^Nch [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] 21.5	^Danyb [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] 21.5		
				Dsup, Danyb ^Db [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [CAFPLrqSP] 99					Danyb ^Db [ABDET] [SETDIAG] [AABrqPa] 21.5	Danyb [ABDET] [SETDIAG] [AABrqPa] 21.5		
				Dsup Dchat [NOTCHAIN] [SETDIAG] [AABrqPa] [VDchatF] [CAFPLrqSP] 99					Dchat [SETDIAG] [AABrqPa] [VDchatF] 21.5			
	[SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [INITRB] [OWEDONE] [COUNTRB] 23.2		^Dsup [SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Dsup, ^Danyb [SETDIAGTP] [TPABi] [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] [VNfaT] 20.3	^Danyb, Dch [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] [RBNEXTSB] [CAFPLrqSB] 99	[SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] [CAFPLrqSB] 99	^Danyb, Dch [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] [RBNEXTSB] 21.3		^Danyb, Nch [SETDIAGTP] [TPABi] [ABDET] [SETDIAG] [AABrqPa] [OWEDONE] [RBNEXTSB] 21.5			
				^Dsup, Danyb ^Db [ABDET] [NOTCHAIN] [SETDIAG] [AABrqPa] 20.3	Danyb ^Db [ABDET] [SETDIAG] [AABrqPa] [CAFPLrqSB] 99		Danyb, ^Dch ^Db [ABDET] [SETDIAG] [AABrqPa] 21.3					

(Continued on next page)

ECMA TRM.ORM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 4 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
Protocol error or Internal error (Concluded 2 of 2)				^Dsup Db, ^Dchat [NOTCHAIN] 20.3			Danyb, Dch ^Db [ABDET] [SETDIAG] [AABrqPa] [RBNEXTSB] 21.3					
				^Dsup Db, Dchat [NOTCHAIN] [SETDIAG] [AABrqPa] [VDchatF] 20.3	Dchat [SETDIAG] [AABrqPa] [VDchatF] [CAFPLrqSB] 99		Db 21.3		Db, ^Dchat 21.5		25	99
TP-DATA req		Dsh ^Ncr [DELIMIT] [UASErq] 18										
		^Dsh ^Ncr, Ddp [DELIMIT] [UASErq] 18										
TP-COMMIT req	[COUNTRDY] [VNtT] 20.2	^Ncr [COUNTRDY] [VNtT] 20.2									[VNcntDEC] [COUNTRDY] [VNtT] 25	
C-COMMIT ind				Dsup ^Nch [RECCOM] [OWEDONECO] 21.5								
				Dsup De [NOTCHAIN] [RECCOM] [OWEDONECO] 21.5								
C-COMMIT+C-BEGIN ind				Dsup Nch, ^De [RECCOM] [OWEDONECO] 21.5								

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 5 of 17*) — Commitment

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		A C T I V E		R E A D Y	D E C I D E D (C O M M I T)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estbishmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
AF-ABORT (user, commitRI) ind				Dsup, ^Dtb [NOTCHAIN] [RECCOM] [TUABi] [ABPTNR] [OWEDONECO] 21.5 Dsup, Dtb [NOTCHAIN] [RECCOM] [ABPTNR] [OWEDONECO] 21.5								
TP-DONE (heuristic-report) req					Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.1	Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.2	Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.3	Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.4	Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.5	Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.6		Dd, ^Dfdone [LOGDAM] [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 99
TP-DONE req					Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.1	Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.2	Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.3	Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.4	Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.5	Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 21.6		Dd [COUNTCOM] [VDfdoneT] [VNfaF] [VDdF] 99
C-COMMIT cnf					^Danyb, ^De [COUNTCOM] [CPSAP] 21.3 ^Danyb, De [SDETrqF] [ABDET] [COUNTCOM] 21.3 ^Dbpart, Dtb [AABrqUd] [SDETrqF] [ABDET] [COUNTCOM] 21.3 Dbpart [SDETrqF] [ABDET] [COUNTCOM] 21.3	^Danyb [CRBrq] [COUNTCOM] [CPSAP] 21.4 ^Dbpart, Dtb [AABrqUr] [SDETrqRBC] [ABDET] [COUNTCOM] 21.3						

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 6 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		A C T I V E		R E A D Y	D E C I D E D (C O M M I T)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
AF-HEURISTIC-REPORT (commitRC) ind					^Danyb, ^De [THRi] [LOGDAM] [COUNTCOM] [CPSAP] 21.3	^Danyb [THRi] [LOGDAM] [CRBrq] [COUNTCOM] [CPSAP] 21.4						
					^Danyb, De [THRi] [LOGDAM] [SDETrqF] [ABDET] [COUNTCOM] 21.3							
					^Dbpart, Dtb [THRi] [LOGDAM] [AABrqUd] [SDETrqF] [ABDET] [COUNTCOM] 21.3	^Dbpart, Dtb [THRi] [LOGDAM] [AABrqUr] [SDETrqRBC] [ABDET] [COUNTCOM] 21.3						
					Dbpart [THRi] [LOGDAM] [SDETrqF] [ABDET] [COUNTCOM] 21.3							
AF-ABORT (user, commitRC) ind					^Danyb, Du [TUABi] [SDETrqF] [ABDET] [OWEDONE] [COUNTCOM] 21.3							
					Danyb, Du ^Dbpart [SDETrqF] [ABDET] [COUNTCOM] 21.3							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 7 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
AF-ABORT-AND-HEURISTIC-REPORT (commitRC) ind					^Danyb, Du [TUABI] [THRI] [LOGDAM] [SDETrqF] [ABDET] [OWEDONE] [COUNTCOM] 21.3							
					Danyb, Du ^Dbpart [THRI] [LOGDAM] [SDETrqF] [ABDET] [COUNTCOM] 21.3							
TP-ROLLBACK req	^Nfrb [RBREQ] [INITRB] [OWEDONE] 23.1	^Nfrb ^Ncr [INITRB] [OWEDONE] [VDaT] 23.3									^Nfrb [INITRB] [OWEDONE] [COUNTRB] 23.2	
	Nfrb [RBREQ] 23.1	Nfrb ^Ncr [VDaT] 23.3									Nfrb [COUNTRB] 23.2	
C-ROLLBACK ind		[TRBI] [LOGDAMRB] [INITRB] [OWEDONE] 23.4	Dsup [TRBI] [LOGDAMRB] [INITRB] [OWEDONE] 23.4	Dsup [TRBI] [LOGDAMRB] [INITRB] [OWEDONE] 23.4								
AF-ABORT (user, rollbackRI) ind				^Danyb, Dsup [TUABIR] [LOGDAMRB] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4								
		[TUABIR] [LOGDAMRB] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4	Dsup [TUABIR] [LOGDAMRB] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4	Danyb, Dsup [TRBI] [LOGDAMRB] [ABPTNR] [NOTCHAIN] [INITRB] [OWEDONE] 23.4								

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 8 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
CAF-RECOVER (ready) ind	[CRErsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2		^Dsup [CRErsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABiR] [THRiH] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [INITRB] [OWEDONE] [COUNTRB] 23.2	^Dsup, ^Danyb ^Ldretry [SETTOKX] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [VNfaT] 20.3	^Danyb ^Dch [RECVRCOMR] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [OWEDONE] 21.1							^Dsup [RECVRCOMR] [VDchatT] [VDrvypF] 21.1
				^Dsup, ^Danyb Ldretry [CRErsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [VNfaT] 20.3	^Danyb Dch [RECVRCOMR] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [RBNEXTSB] [OWEDONE] 21.1	^Danyb [RECVRCOMR] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [OWEDONE] 21.1					
				^Dsup, Danyb ^Db ^Ldretry [SETTOKX] [VDchatT] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] 20.3	Danyb ^Db [RECVRCOMR] [VDchatT] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] 21.1							
				^Dsup, Danyb ^Db Ldretry [CRErsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] 20.3								

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 9 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
CAF-RECOVER (ready) ind (Concluded 2 of 2)				^Dsup, Danyb Db, ^Dchat ^Ldretry [VDchatT] [SETTOKX] 20.3								
				^Dsup, Danyb Db, ^Dchat Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] 20.3								
				^Dsup, Danyb Db, Dchat Ldretry ^Ldretryo [CREnsRT] [SETTOKX] [CAFDETrqF] 20.3	Danyb Db, Dchat [CREnsRT] [SETTOKX] [CAFDETrqF] 21.1							
				^Dsup, Danyb Db, Dchat ^Ldretry Ldretryo [SETTOKX] [OLDCHANNEL] [CREnsRT] [CAFDETrqF] 20.3								
				^Dsup, Danyb Db, Dchat Ldretry Ldretryo [CREnsRT] [SETTOKX] [CAFDETrqF] [OLDCHANNEL] [CREnsRT] [CAFDETrqF] [VDchatF] 20.3								

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 10 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
CAF-RECOVER (commit) ind				Dsup, ^Danyb Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [RECCOM] [OWEDONE] 21.5					Nch, ^Danyb Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [RBNEXTSB] [OWEDONE] 21.5	^Danyb Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [OWEDONE] 21.5		
				Dsup, ^Danyb ^Ldretry [SETTOKX] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [RECCOM] [OWEDONE] 21.5					Nch, ^Danyb ^Ldretry [SETTOKX] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [RBNEXTSB] [OWEDONE] 21.5	^Danyb ^Ldretry [SETTOKX] [VDchatT] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [OWEDONE] 21.5		
									^Nch, ^Danyb Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [OWEDONE] 21.5			
									^Nch, ^Danyb ^Ldretry [VDchatT] [SETTOKX] [DIALOGUE] [SETDIAGTP] [TPABI] [SETDIAG] [AABrqPa] [ABDET] [OWEDONE] 21.5			

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 11 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
CAF-RECOVER (commit) ind (Continued 2 of 3)				Danyb, ^Db Dsup, ^Ldretry [VDchatT] [SETTOKX] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [RECCOM] [OWEDONECO] 21.5					Danyb, ^Db ^Ldretry [VDchatT] [SETTOKX] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] 21.5			Dsup ^Ldretry [VDchatT] [SETTOKX] [VDrvypF] [RECCOM] [OWEDONECO] 21.5
				Danyb, ^Db Dsup, Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [RECCOM] [OWEDONECO] 21.5					Danyb, ^Db Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAG] [AABrqPa] [ABDET] 21.5			Dsup Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [VDrvypF] [RECCOM] [OWEDONECO] 21.5
				Danyb Dsup, Dchat Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] [OLDCHANNEL] [VDchatF] [CAFDETrqCU] [NOTCHAIN] [RECCOM] [OWEDONECO] 21.5					Danyb, Db ^Dchat Ldretry [CREnsRT] [SETTOKX] [CAFDETrqF] 21.5			
				Danyb Dsup, Dchat ^Ldretry [SETTOKX] [OLDCHANNEL] [CAFDETrqCU] [NOTCHAIN] [RECCOM] [OWEDONECO] 21.5					Danyb, Db ^Dchat ^Ldretry [VDchatT] [SETTOKX] 21.5			
									Danyb, Db Dchat ^Ldretry Ldretryo [SETTOKX] [OLDCHANNEL] [CREnsRT] [CAFDETrqF] 21.5			

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 12 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
CAF-RECOVER (commit) ind (Concluded 3 of 3)									Danyb, Db Dchat Ldretry ^Ldretryo [CREnsRT] [SETTOKX] [CAFDETrqF] 21.5 Danyb, Db Dchat Ldretry Ldretryo [CREnsRT] [SETTOKX] [CAFDETrqF] [OLDCHANNEL] [CREnsRT] [CAFDETrqF] [VDchatF] 21.5			
C-RECOVER (commit) ind				Dsup, Dchat Ldretry [CREnsRT] [CAFDETrqF] [VDchatF] [RECCOM] [OWEDONECO] 21.5 Dsup, Dchat ^Ldretry [SETTOKX] [RECCOM] [OWEDONECO] 21.5								
C-RECOVER (done) cnf					^Atwr [CAFDETrqF] [VDchatF] [COUNTCOM] 21.3 Atwr, ^Atolox [CAFDETrqF] [VDchatF] [COUNTCOM] 21.3							
AF-HEURISTIC-REPORT (recoverDoneRC) ind					^Atwr [THRi] [LOGDAM] [CAFDETrqF] [VDchatF] [COUNTCOM] 21.3 Atwr, ^Atolox [THRi] [LOGDAM] [CAFDETrqF] [VDchatF] [COUNTCOM] 21.3							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 13 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI ^Dsup	DI Dsup	DI	DI	DI ^Dsup	DI ^Dsup, Dch	DI ^Dsup	DI ^Dsup, Dch	DI Dsup	DI Dsup, Nch	DI ^Dsup	DI ^Dchat
Event												
C-RECOVER (unknown) cnf				Dsup [TRBi] [LOGDAMRB] [CAFDETrqF] [VDchatF] [INITRB] [OWEDONE] 23.8								
C-RECOVER (retry-later) cnf				Dsup ^Atwr [CAFDETrqF] [VDchatF] [VDrvypT] 99	^Atwr [CAFDETrqF] [VDchatF] [VDrvypT] 99							
				Dsup Atwr, ^Atokx [CAFDETrqF] [VDchatF] [VDrvypT] 99	Atwr, ^Atokx [CAFDETrqF] [VDchatF] [VDrvypT] 99							
AF-TOKEN-GIVE (two-way-recovery) ind				Atwr, Atokx ^Dsup, Dchat [VAtokxF] 20.3	Atwr, Atokx Dchat [VAtokxF] [ATOKGrqTWR] 21.1				Atwr, Atokx Dchat [VAtokxF] 21.5			
AF-TOKEN-PLEASE ind				Dchat 20.3	Dchat 21.1				Dchat 21.5			
CAF-GIVE ind					[CAFDETrqNU] 21.1		[CAFDETrqNU] 21.3		[CAFDETrqNU] 21.5			Dsup [VDchatT] [RECVRRDY] 20.3
												^Dsup [VDchatT] [RECVRCOMI] 21.1
CAF-FAIL ind					21.1		21.3		21.5			Dsup [CAFPLrqSP] 99
												^Dsup [CAFPLrqSB] 99
Heuristic-decision	Ni 17	[LOGHD] 18	Dsup [LOGHD] 20.2	Dsup [LOGHD] 20.3								Dsup [LOGHD] 99
			^Dsup, Ni 20.2	^Dsup, Ni 20.3							25	
Heuristic-damage-comp	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 17	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 18	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 20.2	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 20.3	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 21.1	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 21.2	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 21.3	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 21.4	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 21.5	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 21.6	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 25	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 99
Delay-recovery				^Dsup, Dchat [CREnsRT] [CAFDETrqF] [VDchatF] 20.3					Dchat [CREnsRT] [CAFDETrqF] [VDchatF] 21.5			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 14 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction ° branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
Retry-recovery												Drvyp, Dsup [CAFPLrqSP] [VDrvypF] 99 Drvyp, ^Dsup [CAFPLrqSB] [VDrvypF] 99
Rollback-by-TPPM	^Nfrb [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1	^Nfrb ^Ncr [TRBi] [INITRB] [OWEDONE] 23.3	^Nfrb, Dsup ^Ncr [TRBi] [INITRB] [OWEDONE] 23.3 ^Nfrb,^Dsup [TRBi] [RBREQ] [INITRB] [OWEDONE] 23.1								^Nfrb [TRBi] [INITRB] [OWEDONE] [COUNTRB] 23.2	
	Nfrb [RBREQ] 23.1	Nfrb ^Ncr 23.3	Nfrb, Dsup ^Ncr 23.3 Nfrb, ^Dsup [RBREQ] 23.1								Nfrb [COUNTRB] 23.2	
Restart-TPPM				Dsup [TPABI] [CAFPLrqSP] 99 ^Dsup 20.3	[TPABI] [CAFPLrqSB] 99				21.5			
Rollback-all	[RBREQ] 23.1	23.3	^Dsup [RBREQ] 23.1 Dsup 23.3	^Dsup, ^Db [RBREQ] 23.1 ^Dsup, Db ^Dchat [LOGDAMH] [COUNTRB] 23.2 ^Dsup, Dchat [LOGDAMH] [CREnsU] [CAFDETrqF] [VDchatF] [COUNTRB] 23.2 Dsup 23.3							[COUNTRB] 23.2	

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 15 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		A C T I V E		R E A D Y	D E C I D E D (C O M M I T)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
Enter-ready			Dsup, Ptok [DELIMIT] [PTOKGrq] [CRYrq] 20.3 Dsup, ^Ptok [DELIMIT] [CRYrq] 20.3 ^Dsup 20.3								25	
Set-done-true			[VDdT] 20.2	[VDdT] 20.3	[VDdT] 21.1	[VDdT] 21.2	[VDdT] 21.3	[VDdT] 21.4	[VDdT] 21.5	[VDdT] 21.6	[VDdT] 25	[VDdT] 99
Continue-commit			^De Ptok [COMREQ] 21.1 De Ptok [NOTCHAIN] [COMREQ] 21.1	^Dsup, ^Db ^De Ptok [COMREQ] 21.1 ^Dsup, ^Db De Ptok [NOTCHAIN] [COMREQ] 21.1 ^Dsup, Dchat [RECVRCOMR] 21.1 ^Dsup, ^Dchat Db [CAFPLrqSP] 99							[COUNTCOM] 21.3	
Rollback-next-trans					^Db, Dch 21.2 Db, Dch 21.1 ^Dch 21.1		^Danyb, Dch [CRBrq] 21.4 Dtb, Dch [AABrqUr] [SDETrqRBC] 21.3 Db, Dch 21.3 ^Dch 21.3		^Db, Nch 21.6 Db, Nch 21.5 ^Nch 21.5			99
Complete-commit							Danyb ^Nrpend [CMPCOMSB] [DELBR] 1 Danyb Nrpend [CMPCOMSB] [COUNTRB] 23.2	[CMPCOMSB] 23.1	Danyb, ^Db ^Nrpend [COMRSP] [DELBR] [NEXTAAID] [SDETrqF] 1 Danyb, ^Db Nrpend [COMRSP] [NEXTAAID] [SDETrqF] [ABDET] 23.8	[COMRSP] [CMPCOMSP] [NEXTAAID] 23.3		

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*continued 16 of 17*) — Commitment

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI	DI
Event	^Dsup	Dsup			^Dsup	^Dsup, Dch	^Dsup	^Dsup, Dch	Dsup	Dsup, Nch	^Dsup	^Dchat
Complete-commit (Continued 2 of 3)									Danyb, Db ^Dchat ^Nrpend [DELBR] [NEXTAAID] 1			
									Danyb, Db ^Dchat Nrpend [NEXTAAID] 23.8			
									Danyb, Db Dchat ^Nrpend [RECVRDONE] [DELBR] [NEXTAAID] [CAFDETrqF] [VDchatF] 1			
									Danyb, Db Dchat Nrpend [RECVRDONE] [NEXTAAID] [CAFDETrqF] [VDchatF] 23.8			
							^Danyb, De ^Nrpend [CMPCOMSB] [DELBR] 1		^Danyb, De ^Nrpend [COMRSP] [DELBR] [NEXTAAID] [SDETrqF] 1			
							^Danyb, De Nrpend [CMPCOMSB] [COUNTRB] 23.2		^Danyb, De Nrpend [COMRSP] [NEXTAAID] [SDETrqF] 23.8			
							^Danyb, ^De Dch, Dsh [CMPCOMSB] [RESETD] 2		^Danyb, ^De Nch, Dsh [COMRSP] [CMPCOMSP] [NEXTAAID] [RESETD] 2			
							^Danyb, ^De Dch, ^Dsh, Dg [CMPCOMSB] [RESETD] 3		^Danyb, ^De Nch, ^Dsh, Dg [COMRSP] [CMPCOMSP] [NEXTAAID] [RESETD] 2			
(Continued on next page)												

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.15 (*concluded 17 of 17*) — **Commitment**

State	17	18	20.2	20.3	21.1	21.2	21.3	21.4	21.5	21.6	25	99
Node state		ACTIVE		READY	DECIDED (COMMIT)							
	C-READY ind received TP-COMMIT req awaited	C-PREPARE ind received	Last ready awaited	commit indication awaited	commit confirm awaited	commit confirm awaited need to rbck	commit confirm received	commit confirm received rbck initiat'd	Last commit confirm awaited	Last commit confirm awaited need to rbck	zombie still born transaction branch	channel estblshmnt awaited
Predicates	DI ^Dsup	DI Dsup	DI	DI	DI ^Dsup	DI ^Dsup, Dch	DI ^Dsup	DI ^Dsup, Dch	DI Dsup	DI Dsup, Nch	DI ^Dsup	DI ^Dchat
Event												
Complete-commit (Concluded 3 of 3)							^Danyb, ^De Dch, ^Dsh, ^Dg [CMPCOMSB] [RESETD] 2		^Danyb, ^De Nch, ^Dsh, ^Dg [COMRSP] [CMPCOMSP] [NEXTAAID] [RESETD] 3			
							^Danyb, ^De ^Dch, Dsh [CMPCOMSB] [DELBR] [RESETD] 2		^Danyb, ^De ^Nch, Dsh [COMRSP] [CMPCOMSP] [DELBR] [NEXTAAID] [RESETD] 2			
							^Danyb, ^De ^Dch, ^Dsh, Dg [CMPCOMSB] [DELBR] [RESETD] 3		^Danyb, ^De ^Nch, ^Dsh, Dg [COMRSP] [CMPCOMSP] [DELBR] [NEXTAAID] [RESETD] 2			
							^Danyb, ^De ^Dch, ^Dsh, ^Dg [CMPCOMSB] [DELBR] [RESETD] 2		^Danyb, ^De ^Nch, ^Dsh, ^Dg [COMRSP] [CMPCOMSP] [DELBR] [NEXTAAID] [RESETD] 3			

End of table A.15

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*1 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	Dl ^Dsup	Dl ^Dsup	Dl Dsup	Dl, ^Db Dsup	Dl Dsup	Dl, ^Danyb Dsup, Nch	Dl, Dsup Nfa, Dd	Dl Dsup, ^Nch
Event								
TP-BEGIN-DIALOGUE (accepted) rsp				Nif Ncr [VNcrF] [VDahT] 23.4				
TP-BEGIN-DIALOGUE (rejected) rsp				^Nm, ^Da [ABDrsRUrbc] [DELBR] [SDETrqF] 1				
AF-BEGIN-DIALOGUE (accepted, dataRI) cnf	^Danyb Dcr [TBDcX] [VDcrF] [VDaT] 23.1							
	Danyb Dcr [VDcrF] [VDaT] 23.1							
	^Dcr [VDaT] 23.1							
AF-BEGIN-DIALOGUE (accepted, rollbackRC) cnf	^Du, ^Dtb Dcr [TBDcX] [VDcrF] [VDaT] [COUNTRB] 23.2							
	^Du, ^Dtb ^Dcr [VDaT] [COUNTRB] 23.2							
	Du, ^Dtb Dcr [TBDcX] [VDcrF] [VDaT] [CPSAP] [COUNTRB] 23.2							
	Du, ^Dtb ^Dcr [VDaT] [CPSAP] [COUNTRB] 23.2							
	Dtb [AABrqUd] [SDETrqF] [ABDET] [COUNTRB] 23.2							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 2 of 15*) — **Rollback**

State / Predicates / Event	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
AF-BEGIN-DIALOGUE (rejected(provider), dataRI) cnf	^Danyb [TBDcX] [SDETrqRBC] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb [SDETrqRBC] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							
AF-BEGIN-DIALOGUE (rejected(user), dataRI) cnf	^Danyb, Du [TBDcX] [SDETrqRBC] [ABDET] [OWEDONE] [COUNTRB] 23.2							
	Danyb, Du [SDETrqRBC] [ABDET] [COUNTRB] 23.2							
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) cnf	^Danyb [TBDcX] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							
AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) cnf	^Danyb [TBDcX] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 3 of 15*) — **Rollback**

State / Predicates / Event	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
SAF-ASSOCIATION-LOST ind	^Danyb [TBDcRP] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2 Danyb [ABDET] [NOTCHAIN] [COUNTRB] 23.2							
AF-END-DIALOGUE (confirmation = FALSE) ind	Dx, ^Danyb [TPABiBTED] [SDETrqRBC] [ABDET] [OWEDONE] [COUNTRB] 23.2 Dx, Danyb [SDETrqRBC] [ABDET] [COUNTRB] 23.2							
AF-END-DIALOGUE (confirmation = TRUE) ind	Dx, ^Danyb Denbb=0 [TPABiBTED] [SDETrqRBC] [ABDET] [OWEDONE] [COUNTRB] 23.2 Dx, Danyb Denbb=0 [SDETrqRBC] [ABDET] [COUNTRB] 23.2 Dx, Danyb Denbb>0 [DECDENB] 23.1							
AF-U-ERROR ind	23.1		23.3		23.5			
AF-U-ERROR cnf	Dsh Denb>0 [DECDENB] 23.1		Dsh Denb>0 [DECDENB] 23.3		Dsh Denb>0 [DECDENB] 23.5			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 4 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI	DI	DI	DI, ^Db	DI	DI, ^Danyb	DI, Dsup	DI
Event	^Dsup	^Dsup	Dsup	Dsup	Dsup	Dsup, Nch	Nfa, Dd	Dsup, ^Nch
TP-U-ABORT req	Nfa, ^Danyb [ABTPSUI] [NOTCHAIN] 23.1	Nfa, ^Danyb [ABDET] [NOTCHAIN] [AABrqUd] [OPSAP] [SDETrqF] 23.2	Nfa, ^Danyb [ABTPSUI] [NOTCHAIN] 23.3	^Ncr, Nfa, ^Danyb [ABTPSUI] [NOTCHAIN] 23.4	Nfa, ^Danyb [ABTPSUI] [NOTCHAIN] 23.5	Nfa [NOTCHAIN] [AABrqUd] [SDETrqBF] [ABDET] 23.7	^Danyb, ^Nch [AABrqUd] [OPSAP] [SDETrqF] [ABDET] 23.7 ^Danyb, Nch [AABrqUr] [OPSAP] [SDETrqRBC] [ABDET] [NOTCHAIN] 23.7	
AF-ABORT (user, dataRI) ind	Dx, ^Danyb [TUABi] [SDETrqRBC] [ABDET] [OWEDONE] [COUNTRB] 23.2 Dx, Danyb [SDETrqRBC] [ABDET] [COUNTRB] 23.2	Dch [TUABi] [SDETrqF] [ABDET] [OWEDONE] [NOTCHAIN] 23.2				[TUABi] [SDETrqF] [ABDET] [OWEDONE] [NOTCHAIN] 23.7		

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 5 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates / **Event**	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
AF-ABORT (provider, abortRI) ind or A-P-ABORT ind or A-ABORT ind or A-ABORT req or A-RELEASE (result = affirmative) rsp or A-RELEASE (result = affirmative) cnf	^Danyb, ^Dps [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [COUNTRB] [NOTCHAIN] 23.2	^Danyb [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [NOTCHAIN] 23.2	^Danyb [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [NOTCHAIN] 23.8	^Danyb, ^Ncr [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [NOTCHAIN] 23.8	^Danyb [SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [NOTCHAIN] 23.7	[SETDIAGTP] [TPABi] [ABDET] [OWEDONE] [NOTCHAIN] 23.7		
	Danyb, ^Dps [ABDET] [NOTCHAIN] [COUNTRB] 23.2			^Danyb Ncr [SETDIAGTP] [TPABi] [ABDET] 1	Danyb, ^Dd [ABDET] [NOTCHAIN] [NXTTRAN] 23.5			
			Danyb [ABDET] [NOTCHAIN] 23.8	Danyb [ABDET] [NOTCHAIN] 23.8	Danyb, Dd [ABDET] [NOTCHAIN] 23.7			
	^Danyb, Dps [SETDIAGTP] [TPABi] [THRi] [LOGDAMH] [ABDET] [OWEDONE] [COUNTRB] [NOTCHAIN] 23.2							
	Danyb, Dps [THRi] [LOGDAMH] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 6 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
Event								
Protocol error or internal error	^Danyb, ^Dps [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2	^Danyb [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] 23.2	^Danyb [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] 23.8	^Danyb, ^Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] 23.8	^Danyb [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] 23.7	[SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] 23.7	^Danyb [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] 23.7	
	Danyb, ^Dps [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [COUNTRB] 23.2			^Danyb, Ncr [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] 1	Danyb, ^Db, ^Dd [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [NXTTRAN] 23.5			
		Db 23.2	Danyb, ^Db [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] 23.8	Danyb [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] 23.8	Danyb, ^Db, Dd [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] 23.7		Db 23.7	Db 23.8
	^Danyb, Dps [SETDIAGTP] [TPABi] [THRi] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb, Dps [THRi] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							
AF-GRANT-CONTROL ind or AF-REQUEST-CONTROL ind	^Dsh 23.1		^Dsh 23.3		^Dsh 23.5			
AF-HANDSHAKE ind or AF-HANDSHAKE cnf	Dh 23.1		Dh 23.3		Dh 23.5			
AF-HANDSHAKE-AND-GRANT-CONTROL ind or AF-HANDSHAKE-AND-GRANT-CONTROL cnf	Dh, ^Dsh 23.1		Dh, ^Dsh 23.3		Dh, ^Dsh 23.5			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 7 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI	DI	DI	DI, ^Db	DI	DI, ^Danyb	DI, Dsup	DI
Event	^Dsup	^Dsup	Dsup	Dsup	Dsup	Dsup, Nch	Nfa, Dd	Dsup, ^Nch
C-BEGIN ind						^Dd [NXTTRAN] [CBEAFTRB] 23.6 Dd [CPSAP] [CBEAFTRB] 23.7		
C-BEGIN cnf	[VDbcrT] [VDxF] 23.1							
U-ASE ind	23.1		23.3		23.5			
AF-DEFER (end-dialogue) ind			^De 23.3		^De 23.5			
AF-DEFER (grant-control) ind			^De, ^Dg 23.3		^De, ^Dg 23.5			
AF-PREPARE ind or AF-PREPARE (data-permitted = FALSE) ind or AF-PREPARE (data-permitted = TRUE) ind			23.3		23.5			
TP-DONE (heuristic-report) req	Dd, ^Dfdone ^Nr, Np [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.1 Dd, ^Dfdone Nr [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.1	Dd, ^Dfdone ^Nr, Np [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.2 Dd, ^Dfdone Nr [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.2	Dd, ^Dfdone Np [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.3	Dd, ^Dfdone ^Ncr, Np [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.4				Dd, ^Dfdone Np [VDdF] [VNfaF] [VDfdoneT] [LOGHD] [COUNTRB] 23.8
TP-DONE req	Dd [VDdF] [VNfaF] [VDfdoneT] [COUNTRB] 23.1	^Drbrep, Dd [VDdF] [VNfaF] [VDfdoneT] [COUNTRB] 23.2 Drbrep, Dd [VDdF] [VNfaF] 23.2	Dd [VDdF] [VNfaF] [VDfdoneT] [VDaT] [VNrnT] [COUNTRB] 23.3	Dd ^Ncr [VDdF] [VNfaF] [VDfdoneT] [VDaT] [VNrnT] [COUNTRB] 23.4	Ni, Dd [VDdF] [VNfaF] 23.5	Dd [VDdF] [VNfaF] 23.6	Dd [VDdF] [VNfaF] [NXTTRAN] 23.7	Dd, Danyb [VDdF] [VNfaF] [VDfdoneT] [COUNTRB] 23.8

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 8 of 15*) — **Rollback**

State / Predicates / Event	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
	DI	DI	DI	DI, ^Db	DI	DI, ^Danyb	DI, Dsup	DI
	^Dsup	^Dsup	Dsup	Dsup	Dsup	Dsup, Nch	Nfa, Dd	Dsup, ^Nch
C-ROLLBACK ind	^Du, ^Danyb [CRBrs] [COUNTRB] 23.2				^Danyb, Nch [RBRSPNOAB] 23.6			
	Du, ^Danyb [CRBrs] [COUNTRB] [CPSAP] 23.2				^Danyb, ^Nch, ^Dd [RBRSPNOAB] [NXTTRAN] 23.5			
	Danyb [AABrqUrbc] [SDETrqF] [ABDET] [COUNTRB] 23.2				^Danyb, ^Nch, Dd [RBRSPNOAB] [CPSAP] 23.7			
					Danyb, ^Dd [RBRSPAB] [ABDET] [SDETrqF] [NXTTRAN] 23.5			
			23.4		Danyb, Dd [RBRSPAB] [SDETrqF] [ABDET] 23.7			
AF-HEURISTIC-REPORT (rollbackRI) ind	^Du, ^Danyb [THRi] [LOGDAM] [CRBrs] [COUNTRB] 23.2							
	Du, ^Danyb [THRi] [LOGDAM] [CRBrs] [CPSAP] [COUNTRB] 23.2							
	Danyb [THRi] [LOGDAM] [AABrqUrbc] [SDETrqF] [ABDET] [COUNTRB] 23.2							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 9 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates **Event**	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
AF-ABORT (provider, diagnostic = begin-transaction-reject, rollbackRI) ind	^Danyb [TPABi] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							
AF-ABORT (user, rollbackRI) ind	^Danyb [TUABi] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2		^Danyb [TUABi] [ABPTNR] [NOTCHAIN] [OWEDONE] 23.4		^Danyb [TUABi] [RBRSPNOAB] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] 23.7			
	Danyb [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2		Danyb [ABPTNR] [NOTCHAIN] 23.4		Danyb, Dd [RBRSPNOAB] [SDETrqF] [ABDET] [NOTCHAIN] 23.7			
					Danyb, ^Dd [RBRSPNOAB] [SDETrqF] [ABDET] [NOTCHAIN] [NXTTRAN] 23.5			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 10 of 15*) — **Rollback**

State / Predicates / Event	23.1 rollback req issued rollback confirm awaited DI ^Dsup	23.2 rollback ind/cnf received rollback compl. awaited DI ^Dsup	23.3 rollback not recvd from sup report to sup awaited DI Dsup	23.4 rollback ind recvd from sup report to sup awaited DI, ^Db Dsup	23.5 rollback req issued rollback confirm from sup await'd DI Dsup	23.6 report to sup done C-BEGIN awaited DI, ^Danyb Dsup, Nch	23.7 report to sup done TP-DONE req awaited DI, Dsup Nfa, Dd	23.8 rollback not complete DI Dsup, ^Nch
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRi) ind	^Danyb [THRi] [TUABi] [LOGDAM] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb [THRi] [LOGDAM] [CRBrs] [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							
C-ROLLBACK cnf	^Du, ^Danyb [COUNTRB] 23.2				^Du, ^Danyb 23.6			
	Du, ^Danyb [COUNTRB] [CPSAP] 23.2				Du, ^Danyb, ^Dd [NXTTRAN] 23.5			
	Dtb [AABrqUd] [SDETrqF] [ABDET] [COUNTRB] 23.2				Du, ^Danyb, Dd [CPSAP] 23.7			
	Dbpart [SDETrqF] [ABDET] [COUNTRB] 23.2				Dtb, ^Dd [AABrqUd] [SDETrqBF] [ABDET] [NXTTRAN] 23.5			
					Dtb, Dd [AABrqUd] [SDETrqBF] [ABDET] 23.7			
					Dbpart, ^Dd [SDETrqF] [ABDET] [NXTTRAN] 23.5			
					Dbpart, Dd [SDETrqF] [ABDET] 23.7			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 11 of 15*) — **Rollback**

State / Predicates / Event	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
AF-HEURISTIC-REPORT (rollbackRC) ind	^Danyb, Dch [THRi] [LOGDAM] [COUNTRB] 23.2							
	^Danyb, ^Dch [THRi] [LOGDAM] [COUNTRB] [CPSAP] 23.2							
	Dtb [THRi] [LOGDAM] [AABrqUd] [SDETrqF] [ABDET] [COUNTRB] 23.2							
	Dbpart [THRi] [LOGDAM] [SDETrqF] [ABDET] [COUNTRB] 23.2							
AF-ABORT (provider, diagnostic = begin-transaction-reject, rollbackRC) ind	Du, ^Dbcr ^Danyb [TPABi] [SDETrqF] [ABDET] [OWEDONE] [COUNTRB] 23.2							
	Du, ^Dbcr Danyb [SDETrqF] [ABDET] [COUNTRB] 23.2							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 12 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI	DI	DI	DI, ^Db	DI	DI, ^Danyb	DI, Dsup	DI
Event	^Dsup	^Dsup	Dsup	Dsup	Dsup	Dsup, Nch	Nfa, Dd	Dsup, ^Nch
AF-ABORT (user, rollbackRC) ind	^Danyb [TUABi] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2				^Danyb [TUABi] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] 23.7			
	Danyb [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2				Danyb, Dd [SDETrqF] [ABDET] [NOTCHAIN] 23.7			
					Danyb, ^Dd [NOTCHAIN] [SDETrqF] [ABDET] [NXTTRAN] 23.5			
AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) ind	^Danyb [THRi] [TUABi] [LOGDAM] [SDETrqF] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2							
	Danyb [THRi] [LOGDAM] [SDETrqF] [ABDET] [NOTCHAIN] [COUNTRB] 23.2							

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 13 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI	DI	DI	DI, ^Db	DI	DI, ^Danyb	DI, Dsup	DI
Event	^Dsup	^Dsup	Dsup	Dsup	Dsup	Dsup, Nch	Nfa, Dd	Dsup, ^Nch
CAF-RECOVER (ready) ind	^Danyb [CRErsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABi] [THRi] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] [COUNTRB] 23.2	^Danyb [CRErsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [SETDIAGTP] [TPABi] [SETDIAG] [AABrqPa] [ABDET] [NOTCHAIN] [OWEDONE] 23.2						
	Danyb [CRErsU] [SETTOKX] [CAFDETrqF] [DIALOGUE] [THRi] [LOGDAMH] [SETDIAG] [AABrqPa] [ABDET] [COUNTRB] 23.2	Db [CRErsU] [SETTOKX] [CAFDETrqF] 23.2						
Heuristic-decision-comp	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.1	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.2	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.3	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.4	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.5	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.6	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.7	memsp (SldD, Naaid, Nbrid) [LOGREMOVE] 23.8
Rollback-all	23.1	23.2	23.3	23.4				23.8
Set-done-true	[VDdT] 23.1	[VDdT] 23.2	[VDdT] 23.3	[VDdT] 23.4	[VDdT] 23.5	[VDdT] 23.6	[VDdT] 23.7	[VDdT] 23.8

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*continued 14 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI ^Dsup	DI ^Dsup	DI Dsup	DI, ^Db Dsup	DI Dsup	DI, ^Danyb Dsup, Nch	DI, Dsup Nfa, Dd	DI Dsup, ^Nch
Event								
Report-rollback				^Danyb, Dah ^Nch [ABDrsArbc] [VDahF] [VDaT] [VDrbrepT] [NXTTRAN] 23.4				
				^Danyb, Dah Nch [ABDrsArbc] [VDahF] [VDaT] [VDrbrepT] 23.6				
				^Danyb, ^Dah ^Nch [RBRSPNOAB] [VDaT] [VDrbrepT] [NXTTRAN] 23.4				
				^Danyb, ^Dah Nch [RBRSPNOAB] [VDaT] [VDrbrepT] 23.6				
				Dtb [RBRSPAB] [SDETrqF] [ABDET] [VDaT] [VDrbrepT] [NXTTRAN] 23.4				
			[RBREQ] [VDaT] [VDrbrepT] 23.5	Dbpart [RBRSPNOAB] [SDETrqF] [ABDET] [VDaT] [VDrbrepT] [NXTTRAN] 23.4				[NXTTRAN] 23.8
		[VDrbrepT] 23.2						

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.16 (*concluded 15 of 15*) — **Rollback**

State	23.1 rollback req issued rollback confirm awaited	23.2 rollback ind/cnf received rollback compl. awaited	23.3 rollback not recvd from sup report to sup awaited	23.4 rollback ind recvd from sup report to sup awaited	23.5 rollback req issued rollback confirm from sup await'd	23.6 report to sup done C-BEGIN awaited	23.7 report to sup done TP-DONE req awaited	23.8 rollback not complete
Predicates	DI	DI	DI	DI, ^Db	DI	DI, ^Danyb	DI, Dsup	DI
Event	^Dsup	^Dsup	Dsup	Dsup	Dsup	Dsup, Nch	Nfa, Dd	Dsup, ^Nch
Complete-rollback		Danyb [DELBR] 1		Danyb [DELBR] 1	Danyb [DELBR] 1		Danyb [DELBR] 1	Danyb [DELBR] 1
		^Danyb, Dc, ^Du [RESETD] [CMPRBSB] [CBErq] 2					^Danyb, Dc, Nch [OPSAP] [RESETD] 2	
		^Danyb, ^Dc, ^Du [RESETD] [CMPRBSB] [CBErq] 3					^Danyb, ^Dc, Nch [OPSAP] [RESETD] 3	
		^Danyb, Dc, Du [OPSAP] [RESETD] [DELBR] 2		^Danyb, Dc [OPSAP] [RESETD] [DELBR] 2	^Danyb, Dc [OPSAP] [RESETD] [DELBR] 2	Dc [OPSAP] [RESETD] 2	^Danyb, Dc, ^Nch [OPSAP] [RESETD] [DELBR] 2	
		^Danyb, ^Dc, Du [OPSAP] [RESETD] [DELBR] 3		^Danyb, ^Dc [OPSAP] [RESETD] [DELBR] 3	^Danyb, ^Dc [OPSAP] [RESETD] [DELBR] 3	^Dc [OPSAP] [RESETD] 3	^Danyb, ^Dc, ^Nch [OPSAP] [RESETD] [DELBR] 3	

End of table A.16

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*1 of 7*) **— Channel**

State / Predicates / Event	1 channel does not exist	2 free channel, available	3 free channel, not available	4 channel owned by TPPM	5 token awaited CAF-PLEASE req outstanding	6 AF-BEGIN-DIALOGUE cnf awaited	7 clean-up
					Atwr		
AF-BEGIN-DIALOGUE (Recovery fu selected, one-way-recovery) ind	^Ldrej [ABDrsAd] [VAtwrF] 3 Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqF] 1						
AF-BEGIN-DIALOGUE (Recovery fu selected, two-way-recovery) ind	^Ldrej [ABDrsAd] [VAtwrT] 3 Ldrej [SETDIAGBD] [ABDrsRPd] [SDETrqF] 1						
AF-BEGIN-DIALOGUE (accepted, dataRI) cnf						memsp (SnD, Caaid, Cbrid) [VAtppmT] [VCinitT] [CAFGIVi] 4 ^memsp (SnD, Caaid, Cbrid) [VCinitT] 2	
AF-BEGIN-DIALOGUE (rejected(provider), dataRI) cnf						memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SDETrqF] 1 ^memsp (SnD, Caaid, Cbrid) [SDETrqF] 1	
SAF-ASSOCIATION-LOST ind			1		1	memsp (SnD, Caaid, Cbrid) [CAFFAILi] 1 ^memsp (SnD, Caaid, Cbrid) 1	
AF-END-DIALOGUE ind			1		1		
AF-ABORT (provider, abortRI) ind or A-ABORT ind or A-P-ABORT ind or A-RELEASE (result = affirmative) rsp or A-RELEASE (result = affirmative) cnf		1	1		memsp (SnD, Caaid, Cbrid) [CAFFAILi] 1 ^memsp (SnD, Caaid, Cbrid) 1	memsp (SnD, Caaid, Cbrid) [CAFFAILi] 1 ^memsp (SnD, Caaid, Cbrid) 1	1
Protocol error or Internal error		[SETDIAG] [AABrqPa] 1	[SETDIAG] [AABrqPa] 1		memsp (SnD, Caaid, Cbrid) [SETDIAG] [AABrqPa] [CAFFAILi] 1 ^memsp (SnD, Caaid, Cbrid) [SETDIAG] [AABrqPa] 1	memsp (SnD, Caaid, Cbrid) [SETDIAG] [AABrqPa] [CAFFAILi] 1 ^memsp (SnD, Caaid, Cbrid) [SETDIAG] [AABrqPa] 1	[SETDIAG] [AABrqPa] 1

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*continued 2 of 7*) — **Channel**

State	1	2	3	4	5	6	7
	channel does not exist	free channel, available	free channel, not available	channel owned by TPPM	token awaited CAF-PLEASE req outstanding	AF-BEGIN-DIALOGUE cnf awaited	clean-up
Predicates					Atwr		
Event							
C-RECOVER (ready) ind					^Atokx ^memsb (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [CREirsU] 3		
			^Atokx ^memsb (SnD, AAI, BI) [SETTOKX] [CREirsU] 3		^Atokx ^memsb (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [CREirsU] 3		
					^Atokx memsb (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [VAtppmT] [SETTOKX] [CAFREiRi] 4		
			^Atokx memsb (SnD, AAI, BI) [SETTOKX] [VAtppmT] [CAFREiRi] 4		^Atokx memsb (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [VAtppmT] [CAFREiRi] 4		

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*continued 3 of 7*) — **Channel**

State / Predicates / Event	1 channel does not exist	2 free channel, available	3 free channel, not available	4 channel owned by TPPM	5 token awaited CAF-PLEASE req outstanding Atwr	6 AF-BEGIN-DIALOGUE cnf awaited	7 clean-up
AF-RECOVER (ready) ind			^Atokx Ldretry [SETTOKX] [CREirsRTC] 3		^Atokx Ldretry memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [CREirsRTC] 3		
			^Atokx Ldretry ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [CREirsRTC] 3				
			^Ldretry, ^Atokx ^memsb (SnD, AAI, BI) [SETTOKX] [CREirsU] 3		^Ldretry, ^Atokx ^memsb (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [CREirsU] 3		
					^Ldretry, ^Atokx ^memsb (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [CREirsU] 3		
			^Ldretry, ^Atokx memsb (SnD, AAI, BI) [SETTOKX] [VAtppmT] [CAFREiFi] 4		^Ldretry, ^Atokx memsb (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [VAtppmT] [SETTOKX] [CAFREiFi] 4		
					^Ldretry, ^Atokx memsb (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [VAtppmT] [CAFREiFi] 4		

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*continued 4 of 7*) — **Channel**

State	1	2	3	4	5	6	7
	channel does not exist	free channel, available	free channel, not available	channel owned by TPPM	token awaited CAF-PLEASE req outstanding	AF-BEGIN-DIALOGUE cnf awaited	clean-up
Predicates					Atwr		
Event							
C-RECOVER (commit) ind			^Atokx ^memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) [SETTOKX] [CREsDC] 3		^Atokx ^memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [CREsDC] 3		Atwr [SETTOKX] [CREsRTC] 3
					^Atokx ^memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [CREsDC] 3		^Atwr [CREsRTC] 2
					^Atokx memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [SETTOKX] [CAFFAILi] [AHRrqHrdC] 3		
			^Atokx memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) [SETTOKX] [AHRrqHrdC] 3		^Atokx memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [AHRrqHrdC] 3		
					^Atokx memsp (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [VAtppmT] [CAFREiC] 4		
			^Atokx memsp (SnD, AAI, BI) [SETTOKX] [VAtppmT] [CAFREiC] 4		^Atokx memsp (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [VAtppmT] [CAFREiC] 4		

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*continued 5 of 7*) — **Channel**

State	1	2	3	4	5	6	7
	channel does not exist	free channel, available	free channel, not available	channel owned by TPPM	token awaited CAF-PLEASE req outstanding	AF-BEGIN-DIALOGUE cnf awaited	clean-up
Predicates					Atwr		
Event							
AF-RECOVER (commit) ind			^Atokx Ldretry [SETTOKX] [CREirsRTC] 3		^Atokx Ldretry memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [CREirsRTC] 3		
					^Atokx Ldretry ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [CREirsRTC] 3		
					^Atokx ^Ldretry ^memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [CREirsDC] 3		
			^Atokx ^Ldretry ^memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) [SETTOKX] [CREirsDC] 3		^Atokx ^Ldretry ^memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [CREirsDC] 3		
					^Atokx ^Ldretry memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [AHRrqHrdC] 3		
			^Atokx ^Ldretry memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) [SETTOKX] [AHRrqHrdC] 3		^Atokx ^Ldretry memsp (SldD, AAI, BI) ^memsp (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [AHRrqHrdC] 3		
					^Atokx ^Ldretry memsp (SnD, AAI, BI) memsp (SnD, Caaid, Cbrid) [CAFFAILi] [SETTOKX] [VAtppmT] [CAFREiC] 4		

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*continued 6 of 7*) — **Channel**

State / Predicates / Event	1 channel does not exist	2 free channel, available	3 free channel, not available	4 channel owned by TPPM	5 token awaited CAF-PLEASE req outstanding	6 AF-BEGIN-DIALOGUE cnf awaited	7 clean-up
					Atwr		
AF-RECOVER (commit) ind (Concluded 2 of 2)			^Atokx ^Ldretry memsp (SnD, AAI, BI) [SETTOKX] [VAtppmT] [CAFREiC] 4		^Atokx ^Ldretry memsp (SnD, AAI, BI) ^memsp (SnD, Caaid, Cbrid) [SETTOKX] [VAtppmT] [CAFREiC] 4		
C-RECOVER (retry-later) cnf or C-RECOVER (unknown) cnf							Atwr 3 ^Atwr 2
AF-TOKEN-GIVE (two-way-recovery) ind			Atwr, Atokx [VAtokxF] 2 Atwr, ^Atokx 2		memsp (SnD, Caaid, Cbrid) [VAtppmT] [CAFGIVi] 4 ^memsp (SnD, Caaid, Cbrid) 2	Aq [FLUSHALL 6 ^Aq [PASSTHRU] 6	
AF-TOKEN-PLEASE ind		Atwr, Ptok [ATOKGrqTWR] 3					
CAF-PLEASE req	^Ldtwr [VAtwrF] [SETAAID] [ABDrqRO] 6 Ldtwr [VAtwrT] [SETAAID] [ABDrqRT] 6	[VAtppmT] [SETAAID] [CAFGIVi] 4	Atwr, ^Atokx [SETAAID] [ATOKPrq] 5				
CAF-DETACH (type = free) req				Atwr, Ptok [VAtppmF] 2 Atwr, ^Ptok [VAtppmF] 3 ^Atwr, Cinit [VAtppmF] 2 ^Atwr, ^Cinit [VAtppmF] 3			
CAF-DETACH (type = not-used) req				Atwr [VAtppmF] 2 ^Atwr, Cinit [VAtppmF] 2 ^Atwr, ^Cinit [VAtppmF] 3			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.17 (*concluded 7 of 7*) — **Channel**

State / Predicates / Event	1 channel does not exist	2 free channel, available	3 free channel, not available	4 channel owned by TPPM	5 token awaited CAF-PLEASE req outstanding	6 AF-BEGIN-DIALOGUE cnf awaited	7 clean-up
					Atwr		
CAF-DETACH (type = clean-up) req				[VAtppmF] 7			
Terminate-channel		Ptok [AEDrqF] [SDETrqF] 1					

End of table A.17

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*1 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner) Aw	1.2 AF-BID ind accepted (winner) Aw	2 STRAY	3 BIDDING ^Aw, Aq	4 BID CONFIRM RECEIVED ^Aw	6 BUSY ^Adt	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
SAF-DETACH-ASSOCIATION (free) req				^Aq [RETTOKEN] 1	[VAdtT] 3	^Aq [RETTOKEN] 1	[RETTOKEN] 1			
SAF-DETACH-ASSOCIATION (begin-fear) req				Aq [VAfT] [VAdtT] 2 ^Aq [VAfT] [RETTOKEN] 1			[VAfT] [RETTOKEN] 1			
SAF-DETACH-ASSOCIATION (rollback-indication-expected) req							Aq [DISCARDQ] [RETTOKEN] [RESETS] 1 ^Aq [VAdtT] 7			
SAF-DETACH-ASSOCIATION (rollback-confirm-expected) req							Aq [DISCARDQ] [RETTOKEN] [RESETS] 1 ^Aq [VAdtT] 9			
SAF-DETACH-ASSOCIATION (begin-indication-expected) req							[VAdtT] 8			
AF-BEGIN-DIALOGUE req	^Aw Abm [BIDREQ] [VAdcNEW] [SETCORR] [VAqT] [VAdtF] [QUEUE] 3 ^Aw Af [BIDREQ] [VAdcNEW] [SETCORR] [VAqT] [VAdtF] [QUEUE] 3 ^Aw Ldbid [BIDREQ] [VAdcNEW] [SETCORR] [VAqT] [VAdtF] [QUEUE] 3									

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 2 of 11*) — SACF

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner)	1.2 AF-BID ind accepted (winner)	2 STRAY	3 BIDDING	4 BID CONFIRM RECEIVED	6 BUSY	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
		Aw	Aw		^Aw, Aq	^Aw	^Adt			
AF-BEGIN-DIALOGUE req (Concluded 2 of 2)	^Aw ^Abm,^Af ^Ldbid [VAdcNEW] [SETCORR] [SETLPI] [PASSTHRU] [VAdtF] 2									
	Aw Af [VAdcNEW] [SETCORR] [VAlpiCORR] [VAnfdT] [VAqT] [VAdtF] [QUEUE] 2									
	Aw ^Af [VAdcNEW] [SETCORR] [VAlpiCORR] [VAnfdT] [VAdtF] [PASSTHRU] 2									
AF-BEGIN-DIALOGUE ind	Aw ^Abm, ^Ldres Ptok Anfd, LPI=Alpi CRU [VAdcCORR] [SETCORR] [ATOKGrqKP] [VAdtF] [ATTACHMACF] [PASSTHRU] 6		[VAdcCORR] [ATTACHMACF] [PASSTHRU] 6							
	Aw ^Abm, ^Ldres ^Ptok Anfd, LPI=Alpi CRU [VAdcCORR] [VAtokrT] [ATTACHMACF] [PASSTHRU] [VAdtF] 6									
(Continued on next page)										

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 3 of 11*) — **SACF**

State	1	1.1	1.2	2	3	4	6	7	8	9
	FREE	AF-BID ind pending (winner)	AF-BID ind accepted (winner)	STRAY	BIDDING	BID CONFIRM RECEIVED	BUSY	CLEANUP ROLLBACK INDICATION EXPECTED	CLEANUP BEGIN INDICATION EXPECTED	CLEANUP ROLLBACK CONFIRM EXPECTED
Predicates		Aw	Aw		^Aw, Aq	^Aw	^Adt			
Event										
AF-BEGIN-DIALOGUE ind (Continued 2 of 3)	Aw ^Abm, ^Ldres Anfd, LPI=Alpi ^CFU [VAdcCORR] [ATTACHMACF] [PASSTHRU] [VAdtF] 6									
	Aw ^Abm, Ldres Anfd, LPI=Alpi [VAdtF] [ABDtsRPdAR] 1									
	Aw ^Abm Anfd, LPI^=Alpi [VAdtF] 1			Aw ^Abm 2						
	Aw ^Abm, ^Ldres Ptok ^Anfd CFU [VAdcCORR] [SETCORR] [ATOKGrqKP] [VAdtF] [ATTACHMACF] [PASSTHRU] 6									
	Aw ^Abm, ^Ldres ^Ptok ^Anfd CFU [VAdcCORR] [VAtokrT] [ATTACHMACF] [PASSTHRU] [VAdtF] 6									
	Aw ^Abm, ^Ldres ^Anfd ^CFU [VAdcCORR] [ATTACHMACF] [PASSTHRU] [VAdtF] 6									
(Continued on next page)										

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 4 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner)	1.2 AF-BID ind accepted (winner)	2 STRAY	3 BIDDING	4 BID CONFIRM RECEIVED	6 BUSY	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
		Aw	Aw		^Aw, Aq	^Aw	^Adt			
AF-BEGIN-DIALOGUE ind (Concluded 3 of 3)	Aw ^Abm ^Anfd Ldres [ABDrsRPdAR] [VAdtF] 1									
	^Aw [VAnfdT] [VAlpiCORR] [VAdcCORR] [ATTACHMACF] [PASSTHRU] [VAdtF] 6			^Aw, ^Adt [VAnfdT] [DISCARDQ] [SALi] [RESETS] [VAlpiCORR] [VAdcCORR] [ATTACHMACF] [PASSTHRU] 6	^Adt [VAnfdT] [DISCARDQ] [SALi] [RESETS] [VAlpiCORR] [VAdcCORR] [ATTACHMACF] [PASSTHRU] 6					
				^Aw, Adt [VAnfdT] [DISCARDQ] [RESETS] [VAlpiCORR] [VAdcCORR] [ATTACHMACF] [PASSTHRU] 6	Adt [VAnfdT] [DISCARDQ] [RESETS] [VAlpiCORR] [VAdcCORR] [ATTACHMACF] [PASSTHRU] 6					
AF-BEGIN-DIALOGUE (accepted) rsp or AF-BEGIN-DIALOGUE (rejected(provider)) rsp or AF-BEGIN-DIALOGUE (rejected(user), dataRI) rsp or AF-BEGIN-DIALOGUE (rejected(user), rollbackRC) rsp							[SETCORR] [PASSTHRU] 6			
AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) rsp							[VAdrut] [SETCORR] [COPY] [PASSTHRU] 6			
AF-BEGIN-DIALOGUE (diagnostic = association-reserved, dataRI) cnf	1			DC^=Adc 2	3					
				^Aw DC=Adc [PASSTHRU] 6						
AF-BEGIN-DIALOGUE (diagnostic ^= association-reserved, dataRI) cnf	1			DC^=Adc 2	3	DC=Adc [PASSTHRU] 6				
				DC=Adc [PASSTHRU] 6						
AF-BEGIN-DIALOGUE (diagnostic ^= association-reserved, rollbackRI) cnf or AF-BEGIN-DIALOGUE (diagnostic ^= association-reserved, rollbackRC) cnf				DC=Adc [PASSTHRU] 6		DC=Adc [PASSTHRU] 6				

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 5 of 11*) — **SACF**

State	1	1.1	1.2	2	3	4	6	7	8	9
	FREE	AF-BID ind pending (winner)	AF-BID ind accepted (winner)	STRAY	BIDDING	BID CONFIRM RECEIVED	BUSY	CLEANUP ROLLBACK INDICATION EXPECTED	CLEANUP BEGIN INDICATION EXPECTED	CLEANUP ROLLBACK CONFIRM EXPECTED
Predicates										
Event		Aw	Aw		^Aw, Aq	^Aw	^Adt			
AF-BID (token-requested = FALSE) ind	Aw Anfd, LPl=Alpi Ldres [ABIDrsR] [VAdtF] 1									
	Aw Anfd, LPl=Alpi ^Ldres [ABIDrsA] [VAdtF] 1.2									
	Aw Anfd, LPl=Alpi [VAdtF] 1			Aw 2				Aw 7		Aw 9
	Aw ^Anfd Ldres [ABIDrsR] [VAdtF] 1									
	Aw ^Anfd ^Ldres [ABIDrsA] [VAdtF] 1.2									
AF-BID (token-requested = TRUE) ind	Aw Anfd, LPl=Alpi Ldres [ABIDrsR] [VAdtF] 1									
	Aw Anfd, LPl=Alpi ^Ldres, Ptok [ABIDrsA] [ATOKGrqRG] [VAdtF] 1.2									
	Aw Anfd, LPl=Alpi ^Ldres, ^Ptok ^Lddel [ABIDrsA] [VAtokrT] [VAdtF] 1.2									

(Continued on next page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 6 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner)	1.2 AF-BID ind accepted (winner)	2 STRAY	3 BIDDING	4 BID CONFIRM RECEIVED	6 BUSY	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
		Aw	Aw		^Aw, Aq	^Aw	^Adt			
AF-BID (token-requested = TRUE) ind (Concluded 2 of 2)	Aw Anfd, LPI=Alpi ^Ldres, ^Ptok Lddel [VAtokrT] [VAdtF] 1.1									
	Aw Anfd, LPI^=Alpi [VAdtF] 1			Aw 2				Aw 7		Aw 9
	Aw ^Anfd Ldres [ABIDrsR] [VAdtF] 1									
	Aw ^Anfd ^Ldres, Ptok [ABIDrsA] [ATOKGrqRG] [VAdtF] 1.2									
	Aw ^Anfd ^Ldres, ^Ptok ^Lddel [ABIDrsA] [VAtokrT] [VAdtF] 1.2									
	Aw ^Anfd ^Ldres, ^Ptok Lddel [VAtokrT] [VAdtF] 1.1									
AF-BID (accepted) cnf					Acbegq [FLUSHPAR] [VAfF] 4					
					^Acbegq, ^Adt [FLUSHALL] [VAfF] [VAqF] 4					
					^Acbegq, Adt [FLUSHALL] [VAfF] [VAqF] [VAdtF] 1					

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 7 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner) Aw	1.2 AF-BID ind accepted (winner) Aw	2 STRAY	3 BIDDING ^Aw, Aq	4 BID CONFIRM RECEIVED ^Aw	6 BUSY ^Adt	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
AF-BID (rejected) cnf					Adt [DISCARDQ] [VAfF] [RESETS] 1 ^Adt [DISCARDQ] [VAfF] [RESETS] [SALi] 1					
AF-END-DIALOGUE req				^Aq [PASSTHRU] 2 Aq [QUEUE] 2	[QUEUE] 3	^Aq [PASSTHRU] 4 Aq [QUEUE] 4	[PASSTHRU] 6			
AF-U-ERROR req or AF-ABORT (dataRI) req or AF-ABORT (diagnostic ^= begin-transaction-reject, rollbackRI) req or AF-GRANT-CONTROL req or AF-REQUEST-CONTROL req or AF-HANDSHAKE req or AF-HANDSHAKE-AND-GRANT-CONTROL req or AF-DEFER req or AF-PREPARE req or U-ASE req or C-ROLLBACK req				^Aq [PASSTHRU] 2 Aq [QUEUE] 2	[QUEUE] 3	^Aq [PASSTHRU] 4 Aq [QUEUE] 4	^Aq [PASSTHRU] 6 Aq [QUEUE] 6			
AF-ABORT (diagnostic = begin-transaction-reject, rollbackRI) req				^Aq [PASSTHRU] 2 Aq [QUEUE] 2	[QUEUE] 3	^Aq [PASSTHRU] 4 Aq [QUEUE] 4	^Aq [VAbtrT] [COPY] [PASSTHRU] 6			
AF-ABORT (rollbackRC) req						[PASSTHRU] 4	[PASSTHRU] 6			
AF-ABORT (abortRI) req				[PASSTHRU] *	[PASSTHRU] *	[PASSTHRU] *	[PASSTHRU] *			
AF-END-DIALOGUE ind or AF-END-DIALOGUE cnf or AF-U-ERROR cnf or AF-GRANT-CONTROL ind or AF-REQUEST-CONTROL ind or AF-HANDSHAKE ind or AF-HANDSHAKE cnf or AF-HANDSHAKE-AND-GRANT-CONTROL ind or AF-HANDSHAKE-AND-GRANT-CONTROL cnf or U-ASE ind	1			2	3		[PASSTHRU] 6			
AF-U-ERROR ind	1			2	3		[PASSTHRU] 6	7		9

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 8 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner)	1.2 AF-BID ind accepted (winner)	2 STRAY	3 BIDDING	4 BID CONFIRM RECEIVED	6 BUSY	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
		Aw	Aw		^Aw, Aq	^Aw	^Adt			
AF-END-DIALOGUE rsp or AF-U-ERROR rsp or AF-HANDSHAKE rsp or AF-HANDSHAKE-AND-GRANT-CONTROL rsp or C-BEGIN rsp or C-READY req or C-COMMIT req or C-COMMIT+C-BEGIN req or C-COMMIT rsp or AF-ABORT (user, commitRI) req or AF-ABORT (user, commitRC) req or AF-HEURISTIC-REPORT req or AF-ABORT-AND-HEURISTIC-REPORT req or C-ROLLBACK rsp or P-TOKEN-GIVE (sync-minor) req							[PASSTHRU] 6			
C-RECOVER rsp							[PASSTHRU] [VArvyrsF] 6			
AF-ABORT (provider, abortRI) ind	*.	*.	*.	^Adt [DISCARDQ] [PASSTHRU] * Adt [DISCARDQ] *	^Adt [DISCARDQ] [PASSTHRU] * Adt [DISCARDQ] *	^Adt [DISCARDQ] [PASSTHRU] * Adt [DISCARDQ] *	[DISCARDQ] [PASSTHRU] *	*.	*.	*.
AF-ABORT (user, dataRI) ind	1			2	3		[PASSTHRU] 6	7	1	9
A-ABORT ind or A-P-ABORT ind or A-RELEASE (result = affirmative) cnf	*.	*.	*.	^Adt [PASSTHRU] * Adt *	^Adt [PASSTHRU] * Adt *	^Adt [PASSTHRU] * Adt *	[PASSTHRU] *	*.	*.	*.
A-ABORT req or A-RELEASE (result = affirmative) rsp	[PASSTHRU] *	PASSTHRU] *	[PASSTHRU] *	[DISCARDQ] [PASSTHRU] *	[DISCARDQ] [PASSTHRU] *	[DISCARDQ] [PASSTHRU] *	[DISCARDQ] [PASSTHRU] *	[PASSTHRU] *	[PASSTHRU] *	[PASSTHRU] *
Protocol error	[SETDIAG] [AABrqPa] *	[SETDIAG] [AABrqPa] *	[SETDIAG] [AABrqPa] *	2	Adt [SETDIAG] [AABrqPa] * ^Adt 3	4	6	[SETDIAG] [AABrqPa] *	[SETDIAG] [AABrqPa] *	[SETDIAG] [AABrqPa] *
AF-DEFER ind or C-BEGIN cnf or C-READY ind or C-COMMIT ind or C-COMMIT+C-BEGIN ind or AF-ABORT (user, commitRI) ind or AF-ABORT (user, commitRC) ind or C-COMMIT cnf or AF-HEURISTIC-REPORT (commitRC) ind or AF-ABORT-AND-HEURISTIC-REPORT (commitRC) ind or AF-HEURISTIC-REPORT (recoverDoneRC) ind or C-RECOVER (commit) ind or C-RECOVER cnf or AF-RECOVER (commit) ind							[PASSTHRU] 6			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 9 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner)	1.2 AF-BID ind accepted (winner)	2 STRAY	3 BIDDING	4 BID CONFIRM RECEIVED	6 BUSY	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
		Aw	Aw		^Aw, Aq	^Aw	^Adt			
C-RECOVER (ready) ind or AF-RECOVER (ready) ind							[PASSTHRU] [VArvyrsT] 6			
AF-PREPARE ind							[PASSTHRU] 6			9
C-BEGIN req				Ptok [PASSTHRU] 2		Ptok [PASSTHRU] 4	Ptok [PASSTHRU] 6			
				^Ptok [VAcbegqT] [VAqT] [QUEUE] 2	[VAcbegqT] [QUEUE] 3	^Ptok [VAcbegqT] [VAqT] [QUEUE] 4				
C-BEGIN ind	A1 [VAfF] [CRBrq] 9			Af, ^Adt [SALi] [DISCARDQ] [RESETS] [VAfF] [CRBrq] 9	Af, ^Adt [SALi] [DISCARDQ] [RESETS] [VAfF] [CRBrq] 9					
				Af, Adt [DISCARDQ] [RESETS] [VAfF] [CRBrq] 9	Af, Adt [DISCARDQ] [RESETS] [VAfF] [CRBrq] 9		[PASSTHRU] 6		[CRBrq] 9	
C-ROLLBACK ind				[PASSTHRU] 6		[PASSTHRU] 6	[PASSTHRU] 6	[CRBrs] [RETTOKEN] 1		Abtr [REPREQ] [RETTOKEN] [VAbtrF] 1
										Adru [REPREQ] [RETTOKEN] [VAdruF] 1
										^Abtr, ^Adru [CRBrs] [RETTOKEN] 1
AF-HEURISTIC-REPORT (rollbackRI) ind or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) ind				[PASSTHRU] 6		[PASSTHRU] 6	[PASSTHRU] 6			
AF-ABORT (rollbackRI) ind				[PASSTHRU] 6		[PASSTHRU] 6	[PASSTHRU] 6			[CRBrs] [RETTOKEN] 1
C-ROLLBACK cnf or AF-HEURISTIC-REPORT (rollbackRC) ind or AF-ABORT (rollbackRC) ind or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRC) ind				[PASSTHRU] 6		[PASSTHRU] 6	[PASSTHRU] 6			[RETTOKEN] [VAbtrF] [VAdruF] 1
C-RECOVER (ready) req or AF-RECOVER req							Ptok [PASSTHRU] 6			
							^Ptok [VAqT] [QUEUE] 6			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*continued 10 of 11*) — **SACF**

State / Predicates / Event	1 FREE	1.1 AF-BID ind pending (winner)	1.2 AF-BID ind accepted (winner)	2 STRAY	3 BIDDING	4 BID CONFIRM RECEIVED	6 BUSY	7 CLEANUP ROLLBACK INDICATION EXPECTED	8 CLEANUP BEGIN INDICATION EXPECTED	9 CLEANUP ROLLBACK CONFIRM EXPECTED
		Aw	Aw		^Aw, Aq	^Aw	^Adt			
C-RECOVER (commit) req							Arvyrs [PASSTHRU] [VArvyrsF] 6			
							^Arvyrs Ptok [PASSTHRU] 6			
							^Arvyrs ^Ptok [QUEUE] 6			
AF-TOKEN-GIVE (regular) ind	Aw [VAfF] 1	[ABIDrsA] [ATOKGrqRG] [VAtokrF] 1.2	Atokr [ATOKGrqRG] [VAtokrF] 1.2	Aw, ^Adt Aq [FLUSHALL] [VAcbegqF] [VAqF] [VAfF] 2		Aq [FLUSHALL] [VAcbegqF] [VAqF] [VAfF] 4	Aw Atokr [SETCORR] ATOKGrqKP] [VAtokrF] 6			Aw [VAfF] 9
				Aw, Adt Aq [FLUSHALL] [VAcbegqF] [VAqF] [VAfF] 1						
			^Atokr [VAfF] 1.2	Aw ^Aq [VAfF] 2		^Aq				
	^Aw [ATOKGrqRG] 1			^Aw [ATOKGrqRG] 2	[ATOKGrqRG] 3	4				^Aw 9
AF-TOKEN-GIVE (keep) ind	^Aw [ATOKGrqRG] 1			^Aw DC^=Adc [ATOKGrqRG] 2	DC^=Adc [ATOKGrqRG] 3					
				^Aw DC=Adc [FLUSHALL] [PASSTOKEN] [VAfF] [VAqF] [VAcbegqF] 2			^Aw DC=Adc [FLUSHALL] [PASSTOKEN] [VAfF] [VAqF] [VAcbegqF] 6			

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Table A.18 (*concluded 11 of 11*) — **SACF**

State	1	1.1	1.2	2	3	4	6	7	8	9
	FREE	AF-BID ind pending (winner)	AF-BID ind accepted (winner)	STRAY	BIDDING	BID CONFIRM RECEIVED	BUSY	CLEANUP ROLLBACK INDICATION EXPECTED	CLEANUP BEGIN INDICATION EXPECTED	CLEANUP ROLLBACK CONFIRM EXPECTED
Predicates / **Event**		Aw	Aw		^Aw, Aq	^Aw	^Adt			
AF-TOKEN-GIVE (two-way-recovery) req							Ptok [PASSTHRU] 6			
							^Ptok [VAqT] [QUEUE] 6			
AF-TOKEN-GIVE (two-way-recovery) ind							[PASSTHRU] 6			
P-TOKEN-GIVE (sync-minor) ind	Aw [VAtF] 1			Aw, ^Adt Aq [FLUSHALL] [VAcbegqF] [VAqF] [VAtF] 2						Aw [VAtF] 9
				Aw, Adt Aq [FLUSHALL] [VAcbegqF] [VAqF] [VAtF] 1						
				Aw ^Aq [VAtF] 2						
	^Aw [ATOKGrqRG] 1			^Aw [ATOKGrqRG] 2	[ATOKGrqRG] 3		[PASSTOKEN] 6			^Aw 9
AF-TOKEN-PLEASE req							[PASSTHRU] 6			
AF-TOKEN-PLEASE ind	1			2	3		[PASSTHRU] 6			
Reject-bid		[ABIDrsR] 1								

End of table A.18

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex B
(normative)

Requirements for writing U-ASEs and application contexts

U-ASEs that are to be used within the OSI TP environment (i.e., included in an OSI TP application-context) shall observe the following guidelines while operating within the bounds of a dialogue:

a) They shall document in the application context definition, any direct use of ASEs or Presentation including the embedding of U-ASE APDUs in ASE APDUs or Presentation PDUs;

b) They shall document in the application context definition any direct use of ACSE that conflicts with the TP use of ACSE;

NOTE 1 It is recommended that the U-ASE does not make direct use of ACSE; however, the U-ASE could conceivably use the A-ABORT request or A-RELEASE request. The affirmative confirmation of the A-RELEASE request has the same effect on the TPPM as if an A-ABORT request has been received; depending on when the A-ABORT service is invoked or the A-RELEASE confirm is received, the existing transaction (if any) will be rolled back or recovered as appropriate (see 8.5.9).

c) They shall not use those non-sharable Presentation Services which are used by the TP-ASE, CCR, or ACSE;

NOTE 2 It is permissible for the U-ASE to use the Session minor synchronize service and resynchronization service, as long as resynchronization is not done to a point before the beginning of the transaction. The use of the resynchronization service by the U-ASE should not disrupt the TP services. The U-ASE may make use of the Session major or minor synchronize service when not part of a provider-supported transaction. In this case, a U-ASE is responsible for the movement of the tokens in order that an appropriate U-ASE can have the tokens at dialogue establishment time.

d) They shall not use the CCR service directly;

e) They shall ensure that they work properly within the restrictions of this protocol, without loss of semantics.

NOTE 3 For example, a U-ASE may not initiate commitment prior to receiving all expected U-ASE information.

In particular, when CCR is part of the application-context, the protocols of the U-ASEs that utilize the synchronize-minor token shall be such that the synchronize-minor token is owned by the issuer of the TP-BEGIN-TRANSACTION request;

f) The TP Service and Protocol make use of a dummy service, TP-DATA, to indicate where one or more U-ASE services may be included in the allowed sequence of TP services. For each such occurrence of TP-DATA, the U-ASE specification shall indicate the specific U-ASE service(s), and their sequencing rules that may be substituted. Thus the U-ASE specification shall expand (and complete if it is the only U-ASE) the TPSP rules specified in this standard.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex C
(informative)

Scenarios

C.1 Introduction

This annex contains scenarios which may assist in the understanding of OSI TP. Examples have been chosen to illustrate the most common functions and combinations of sequences of primitives. These are however just a few of the many possible sequences; these examples do not by themselves place any constraints on the use of OSI TP.

The representation of OSI TP Protocol flows and their relationship to OSI TP Service are shown in figure C.1.

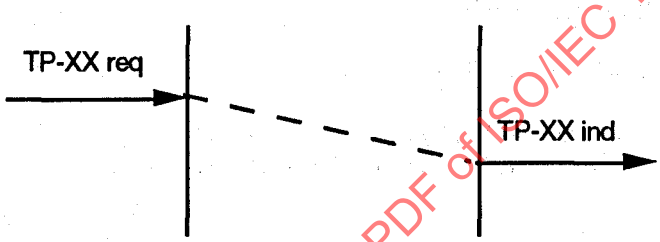


Figure C.1 — Representation of OSI TP Protocol flows

In the figures in this annex, a solid vertical line implies that there is an active dialogue and interaction can occur at any time. A dotted vertical line implies that there are still OSI TP Service interactions to complete the transaction, although the dialogue is terminated. In addition, note that some scenarios do not show complete sequences.

The following list gives the figure number and the title of scenarios supplied in this annex:

C.1.1 Scenarios with a single dialogue (successful cases)

C.2 Application supported transactions with handshakes in Polarized Control
C.3 Application supported transactions with handshakes in Shared Control
C.4 Provider-supported chained transactions
C.5 Provider-supported unchained transactions
C.6 TP-PREPARE with Data-Permitted="false" in Polarized Control
C.7 TP-PREPARE with Data-Permitted="true" in Polarized Control
C.8 TP-PREPARE in Shared Control
C.9 Handshake service, immediate response
C.10 Handshake service, delayed response
C.11 Deferred end dialogue, normal case

C.1.2 Scenarios with a single dialogue (unsuccessful cases)

(1) Negative Response to TP-HANDSHAKE

C.12 Negative response to a TP-HANDSHAKE in Polarized Control
C.13 Collision between a TP-HANDSHAKE and a TP-U-ERROR in Polarized Control
C.14 Negative response to a TP-HANDSHAKE in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.15 Collision between a TP-HANDSHAKE and a TP-U-ERROR in Shared Control

(2) TP-ROLLBACK scenarios

C.16 TP-ROLLBACK with Chained Transactions
C.17 TP-ROLLBACK with Unchained Transactions

(3) Rejection of a TP-BEGIN-TRANSACTION

C.18 Rejection of a TP-BEGIN-TRANSACTION

(4) TP-U-ERROR collision with TP-DATA

C.19 TP-U-ERROR collision with TP-DATA in Polarized Control
C.20 TP-U-ERROR collision with TP-DATA in Shared Control

(5) TP-ROLLBACK with TP-U-ABORT

C.21 TP-U-ABORT response to rollback

(6) TP-DEFERRED-END-DIALOGUE with TP-ROLLBACK

C.22 Deferred end dialogue cancelled by rollback in active phase
C.23 Deferred end dialogue cancelled by rollback in termination phase

(7) Dialogue establishment scenarios

C.24 Rejection of the dialogue establishment, coordination level "none"
C.25 Rejection of the dialogue establishment, coordination level "commitment"
C.26 Rejection of the dialogue establishment, coordination level "commitment" (TP-COMMIT request issued)
C.27 Aborting the dialogue establishment by the requestor
C.28 Dialogue establishment followed by rollback from the requestor, dialogue rejected by recipient
C.29 Dialogue establishment followed by rollback from the requestor, dialogue accepted by recipient
C.30 Failure after a dialogue Establishment
C.31 Rejection of a dialogue establishment request causing rollback
C.32 Rejection of a dialogue establishment request after TP-PREPRE
C.33 Aborting the dialogue establishment by requestor before acceptance
C.34 Aborting the dialogue establishment by requestor after acceptance
C.35 Aborting the dialogue establishment by recipient after acceptance
C.36 Rollback on a rejected dialogue
C.37 Rollback on an accepted dialogue
C.38 Failure during dialogue establishment before acceptance
C.39 Failure during dialogue establishment after acceptance
C.40 Dialogue establishment followed by rollback and subsequent communication failure
C.41 Confirmed dialogue establishment followed by rollback and subsequent user abort, recipient accepts the dialogue
C.42 Confirmed dialogue establishment followed by rollback and subsequent user abort, recipient rejects the dialogue

C.1.3 Scenarios with a single dialodue (failure case)

C.43 TP-P-ABORT during active phase
C.44 TP-P-ABORT during the first phase of commitment
C.45 TP-P-ABORT during the second phase of commitment after decision has reached subordinate
C.46 TP-P-ABORT during the second phase of commitment after the end of the transaction for the subordinate
C.47 TP-P-ABORT during the second phase of commitment before decision has reached subordinate

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.1.4 Collision scenarios on a single dialogue

C.48 Collision of TP-U-ERROR and TP-COMMIT
C.49 Collision of TP-U-ERRORs in Polarized Control
C.50 Collision of TP-U-ERRORs issued without having control
C.51 Suppression effect of TP-U-ERROR in Shared Control
C.52 Collision of TP-U-ERRORs in Shared Control
C.53 TP-U-ERROR during an outstanding handshake request
C.54 TP-U-ERROR during an outstanding handshake indication
C.55 Collision of TP-U-ERROR with TP-END-DIALOGUE
C.56 Collision of TP-U-ERRORs with TP-HANDSHAKE indication in Shared Control
C.57 Queuing TP-U-ERROR-RC before TP-BEGIN-DIALOGUE-RC is sent in Shared Control
C.58 Two TP-U-ERROR requests after a TP-HANDSHAKE collision in Shared Control
C.59 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-END-DIALOGUE (Confirmation="false")
C.60 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-END-DIALOGUE (Confirmation="true")
C.61 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-U-ERROR in Polarized Control
C.62 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-U-ERROR in Shared Control
C.63 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-REQUEST-CONTROL
C.64 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-BEGIN-TRANSACTION
C.65 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-HANDSHAKE
C.66 Collision of a TP-COMMIT request and a TP-BEGIN-DIALOGUE confirm (rejected(provider))
C.67 Collision of a TP-COMMIT request and a TP-BEGIN-DIALOGUE confirm (rejected(user))
C.68 Collision of TP-COMMIT and TP-DATA

C.1.5 Tree with multiple dialogues (successful cases)

C.69 Commitment in a multi-dialogue tree (Chained Transactions)
C.70 Commitment in a multi-dialogue tree with usage of TP-PREPARE (Unchained Transactions)

C.1.6 Tree with multiple dialogues (unsuccessful cases)

C.71 Rollback from root node in Chained Transactions
C.72 Rollback from an intermediate node in Unchained Transactions
C.73 Rollback from an intermediate node in Chained Transactions
C.74 Two failures during first phase of commitment
C.75 Rollback-related actions (three dialogues aborted)
C.76 Failure in active phase; two isolated trees
C.77 Failure in active phase; superior tree aborted
C.78 Failure after TP-COMMIT indication issued to superior; rollback of next transaction in superior tree
C.79 Failure after TP-COMMIT indication issued to superior; isolated node refuses to be root; rollback of next transaction
C.80 Failure after all TPSUIs have received TP-COMMIT indication; rollback of next transaction in both trees
C.81 Failure after all TPSUIs have received TP-COMMIT indication; isolated node refuses to be root; rollback of next transaction in both trees
C.82 Failure after TP-COMMIT indication issued to superior; superior aborts all its dialogues; rollback of next transaction in superior tree
C.83 Failure after TP-COMMIT-COMPLETE indication issued to subordinate; rollback of next transaction in both trees

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.1.7 Heuristic decisions and reporting

C.84 Dialogue aborted during commitment; subordinate takes (wrong) decision to rollback
C.85 Dialogue aborted during commitment; subordinate takes (correct) decision to commit
C.86 Dialogue aborted during commitment; subordinate takes (wrong) decision to rollback but is able to compensate
C.87 Dialogue aborted during commitment; subordinate takes contradicting heuristic decisions producing an internal heuristic mix
C.88 Dialogue aborted during commitment; subordinate takes (wrong) decision to commit
C.89 Dialogue aborted during commitment; subordinate takes (correct) decision to rollback but a heuristic-damage is reported
C.90 Dialogue aborted before subordinate is aware of transaction termination; no heuristic decision taken but a heuristic-damage is reported
C.91 Heuristic decision and reporting in a multi-dialogue tree; all nodes above the failure receive the heuristic report
C.92 Heuristic decision and reporting in a multi-dialogue tree; root node is able to compensate and does not report heuristic mix to its TPSUI
C.93 Heuristic decision and reporting in a multi-dialogue tree; Heuristic report during rollback procedure

C.1.8 Scenarios for SACF

C.94 BID used in the commit
C.95 BID not used in the commit

C.1.9 Scenarios for CPM

C.96 Channel establishment requested by contention-winner
C.97 Channel establishment requested by contention-loser(simple case)
C.98 Channel establishment requested by contention-loser(complex case)
C.99 Two way recovery (simple case)
C.100 Recovery collision due to requests by both a superior and a subordinate

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

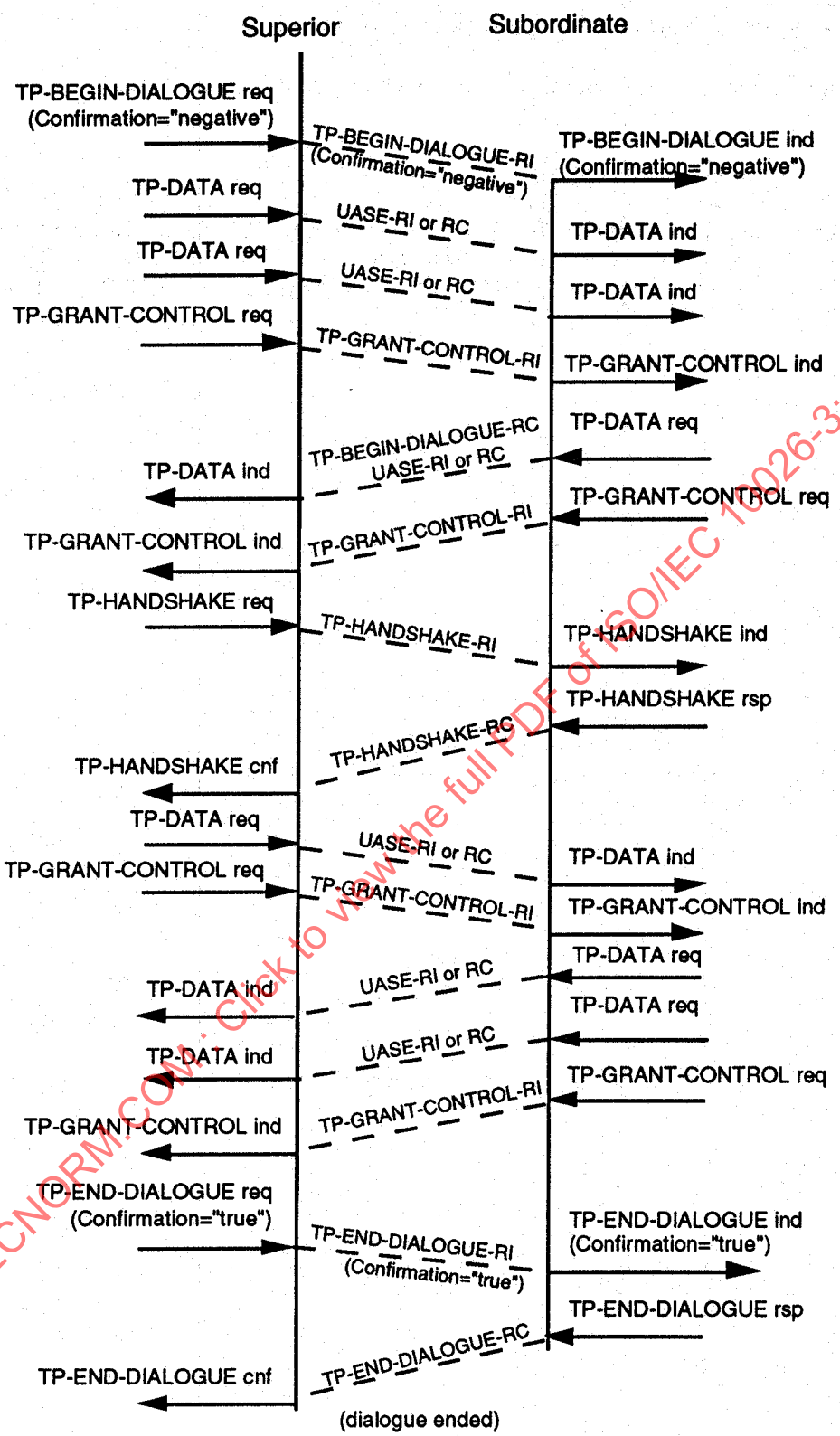
C.2 Scenarios with a single dialogue (successful cases)

The following scenarios demonstrate the service primitives in one dialogue between two TPSUIs.

C.2.1 Application supported transactions.

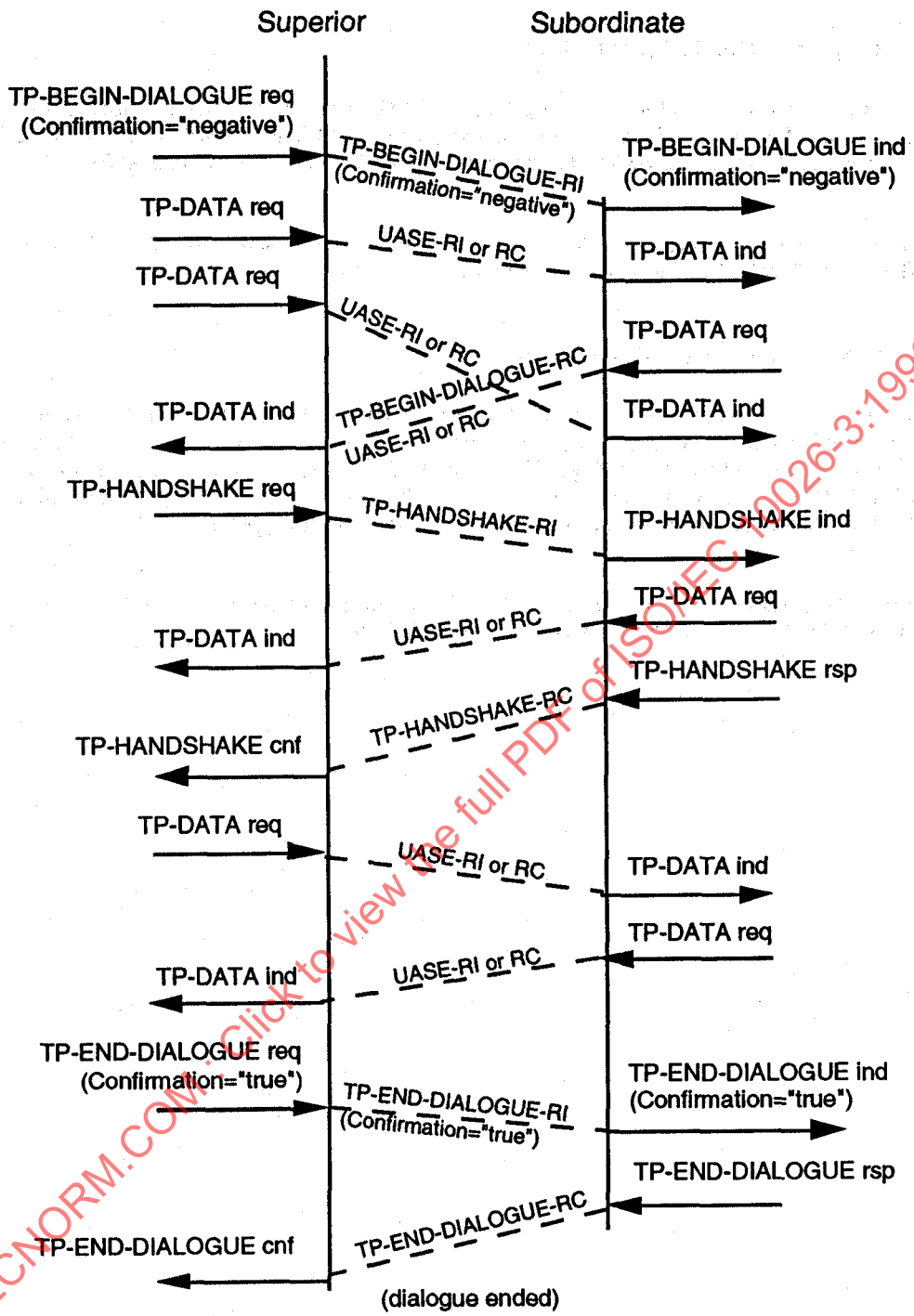
Two scenarios are shown. Figure C.2 illustrates Polarized Control and figure C.3 illustrates Shared Control. Apart from that the scenarios are equivalent. In each, a dialogue is begun and data is passed between the two TPSUIs. At the midpoint, the Handshake service is used to synchronize the two partners. After further transfer of data, the partners synchronize again and end the dialogue (by using the confirmed TP-END-DIALOGUE service).

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure C.2 — Application supported transactions with handshakes in Polarized Control



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure C.3 — Application supported transactions with handshakes in Shared Control

C.2.2 Provider-supported chained transactions

The scenario in figure C.4 shows a dialogue being established between two TPSUIs to support a chained sequence of transactions. After the second transaction is completed, the dialogue is ended. The TP-DEFERRED-END-DIALOGUE service is used to request termination of the dialogue after the successful conclusion of the second transaction.

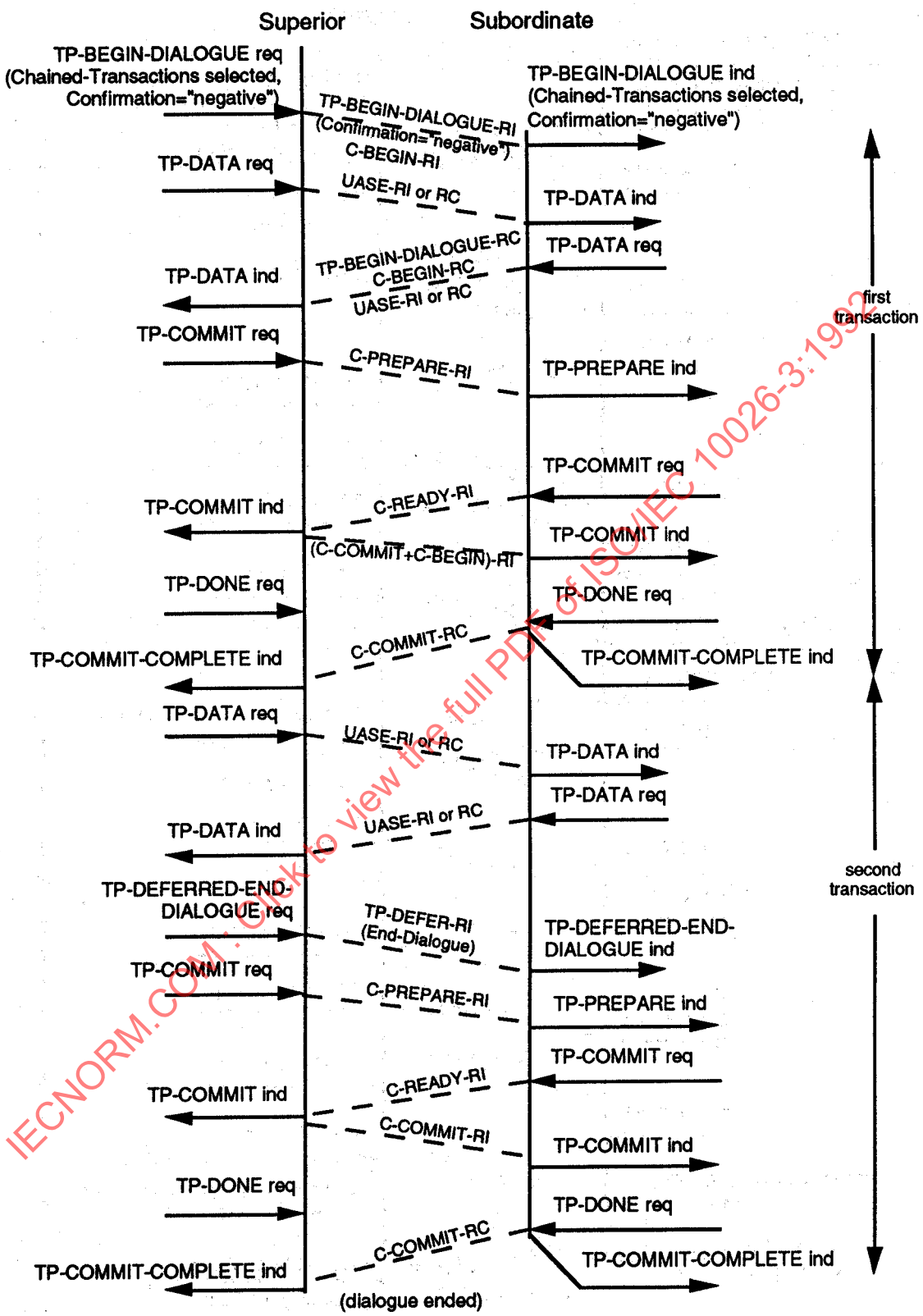
NOTE — As drawn, the scenario is accurate for Shared Control; for Polarized Control, each TP-DATA request would need to be followed by a TP-GRANT-CONTROL request to pass control of the dialogue.

C.2.3 Provider-supported unchained transactions

The scenario in figure C.5 shows a dialogue being established between two TPSUIs to support unchained transactions. Two transactions are performed. After the second transaction is completed, the dialogue is ended. The first transaction is begun simultaneously with the dialogue (parameter Begin-Transaction of TP-BEGIN-DIALOGUE set to "true"). After the first transaction has finished, data can be exchanged between the two TPSUIs outside the scope of a transaction although this is not shown. After the second transaction has ended, the dialogue is terminated by the superior TPSUI.

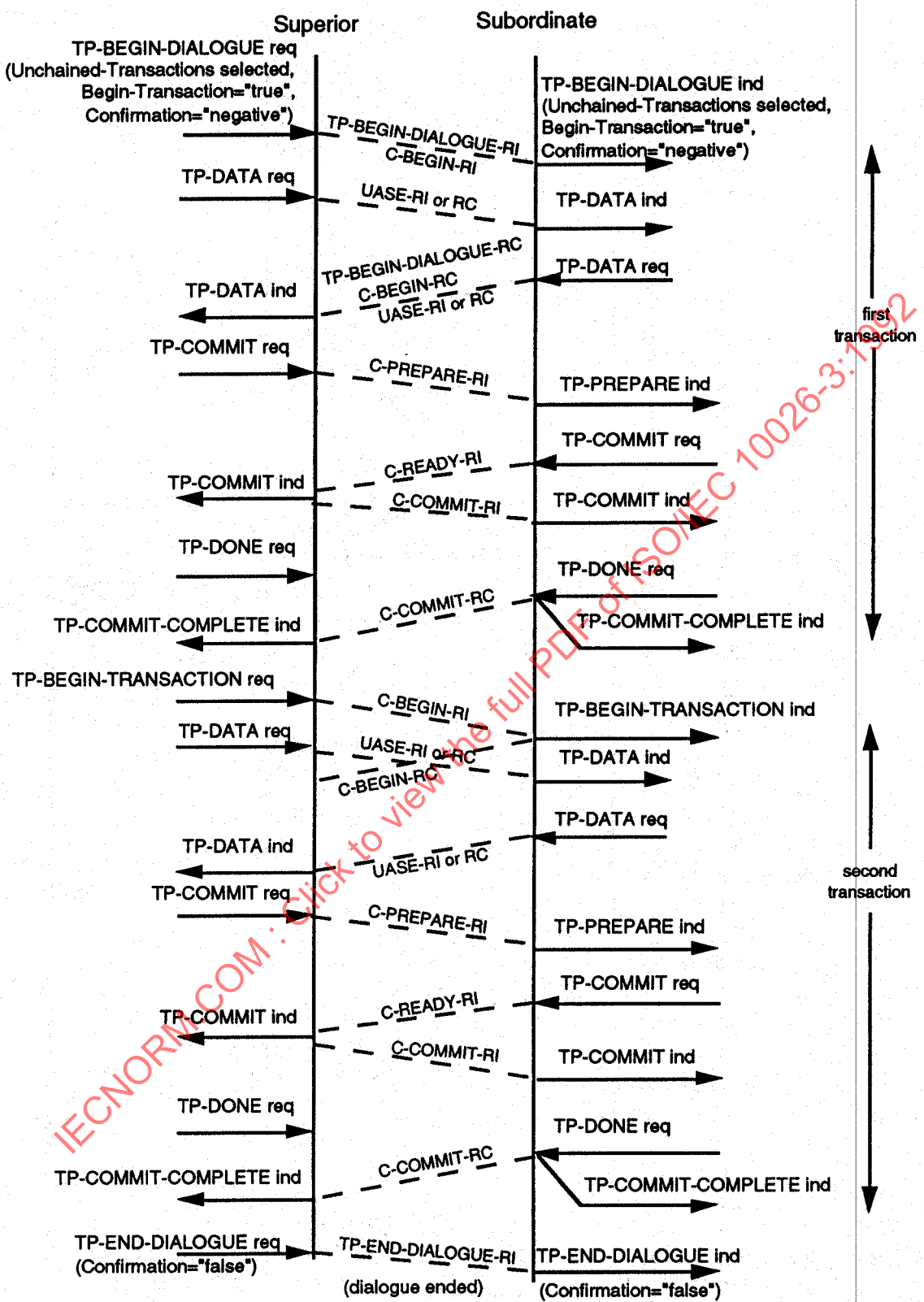
NOTE — As drawn, the scenario is accurate for Shared Control; for Polarized Control, each TP-DATA request would need to be followed by a TP-GRANT-CONTROL request to pass control of the dialogue.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure C.4 — Provider-supported chained transactions



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure C.5 — Provider-supported unchained transactions

C.2.4 TP-PREPARE scenarios

Three scenarios are documented for TP-PREPARE:

- TP-PREPARE with Data-Permitted="false" in Polarized Control;
- TP-PREPARE with Data-Permitted="true" in Polarized Control;
- TP-PREPARE in Shared Control.

C.2.4.1 TP-PREPARE with Data-Permitted="false" in Polarized Control

The scenario in figure C.6 describes a sequence of primitives in the case when data is exchanged between two TPSUIs and a TP-PREPARE request is issued with Data-Permitted="false" in Polarized Control.

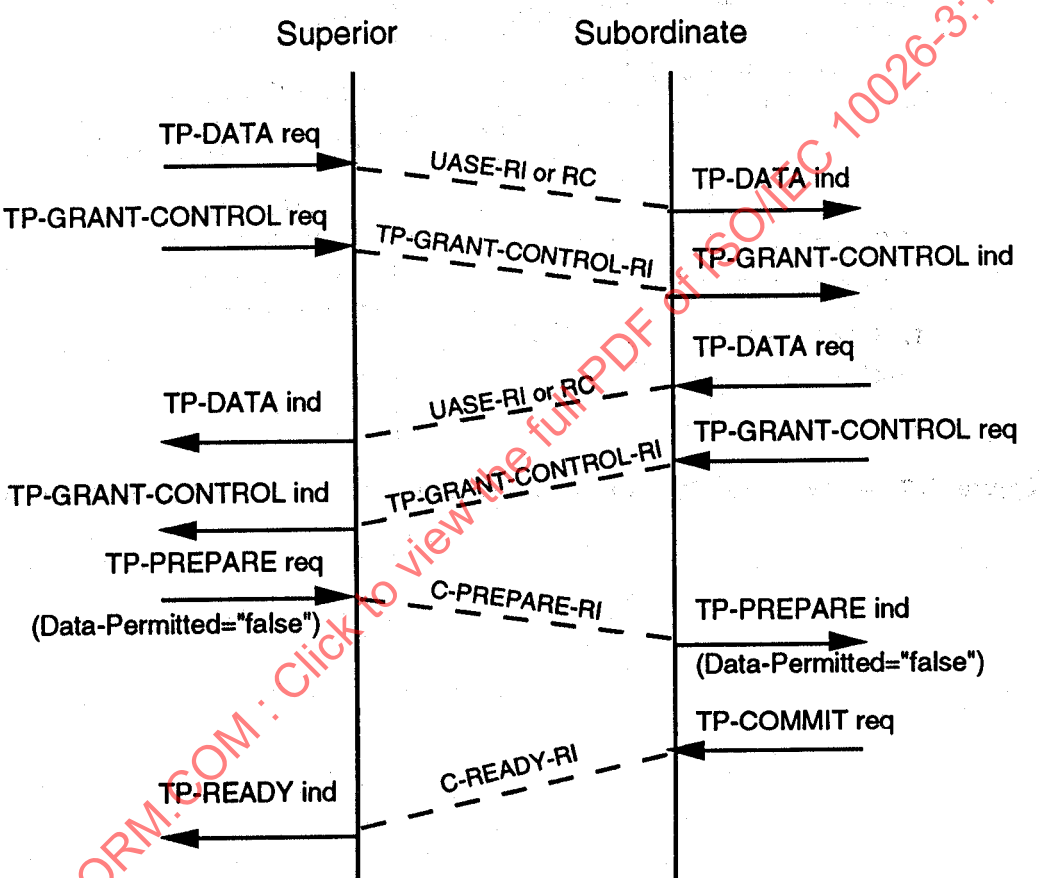


Figure C.6 — TP-PREPARE with Data-Permitted="false" in Polarized Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.2.4.2 TP-PREPARE with Data-Permitted="true" in Polarized Control

The scenario in figure C.7 describes a similar sequence of primitives in the case when data is sent to the remote TPSUI and then a TP-PREPARE request is issued with Data-Permitted="true" to allow the remote TPSUI to send data before issuing the TP-COMMIT request.

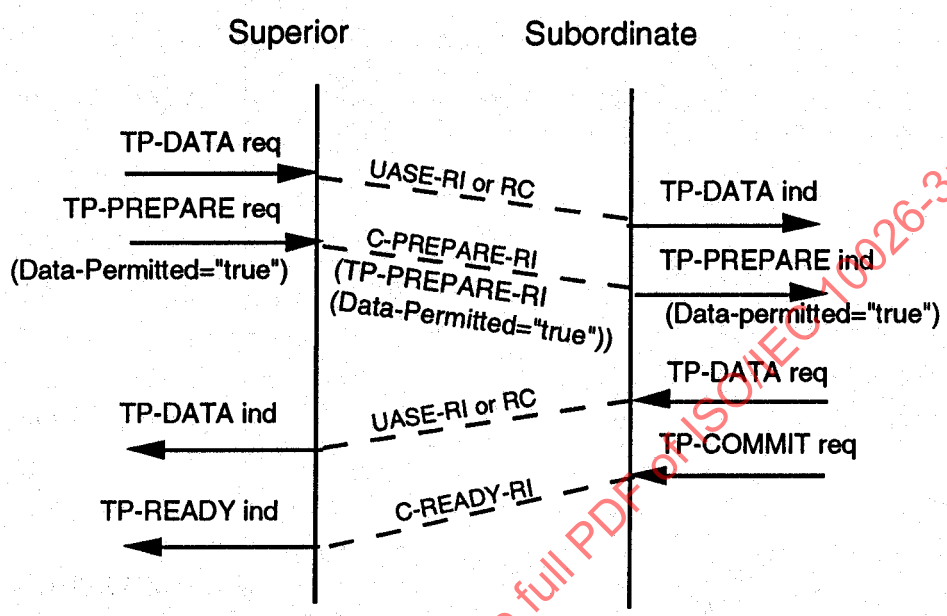


Figure C.7 — TP-PREPARE with Data-Permitted="true" in Polarized Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.2.4.3 TP-PREPARE in Shared Control

The scenario in figure C.8 describes a sequence of primitives in the case when data is exchanged between two TPSUIs and a TP-PREPARE request is issued in Shared Control.

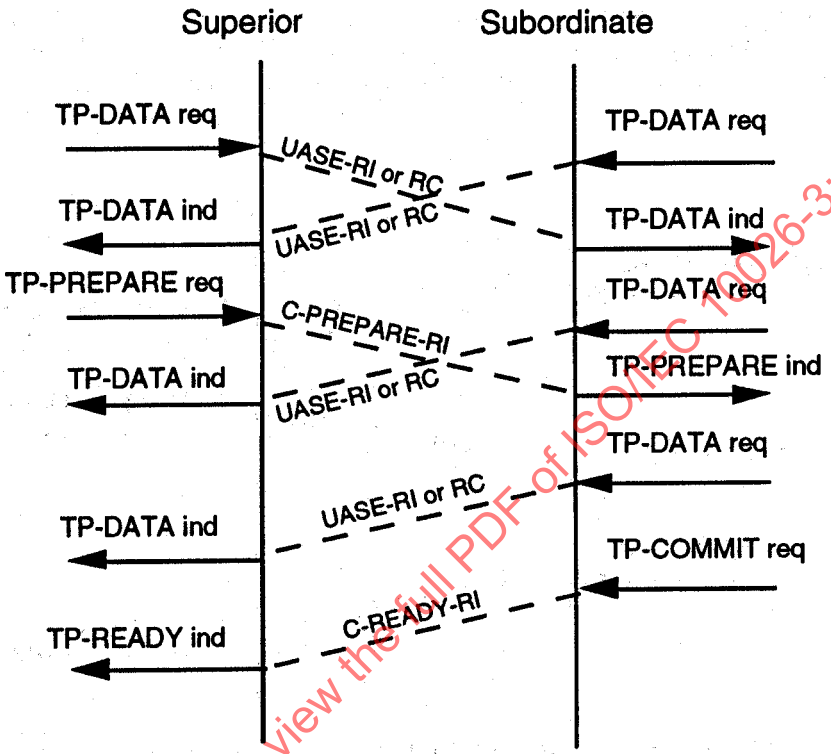


Figure C.8 — TP-PREPARE in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.2.5 Handshake services: illustration of Confirmation-Urgency parameter

Scenarios in figures C.9 and C.10 demonstrate the effect of the Confirmation-Urgency parameter of the TP-HANDSHAKE request and TP-HANDSHAKE-AND-GRANT-CONTROL request services. The Confirmation-Urgency parameter is passed to the TPPM at side B but it is not made visible to the TPSUI.

- Figure C.9: when 1) a TP-HANDSHAKE is used in Polarized Control; 2) TP-HANDSHAKE-AND-GRANT-CONTROL with Confirmation-Urgency set to "urgent"; or 3) TP-HANDSHAKE is used in Shared Control and Confirmation-Urgency set to "urgent". The confirmation shall be issued immediately.

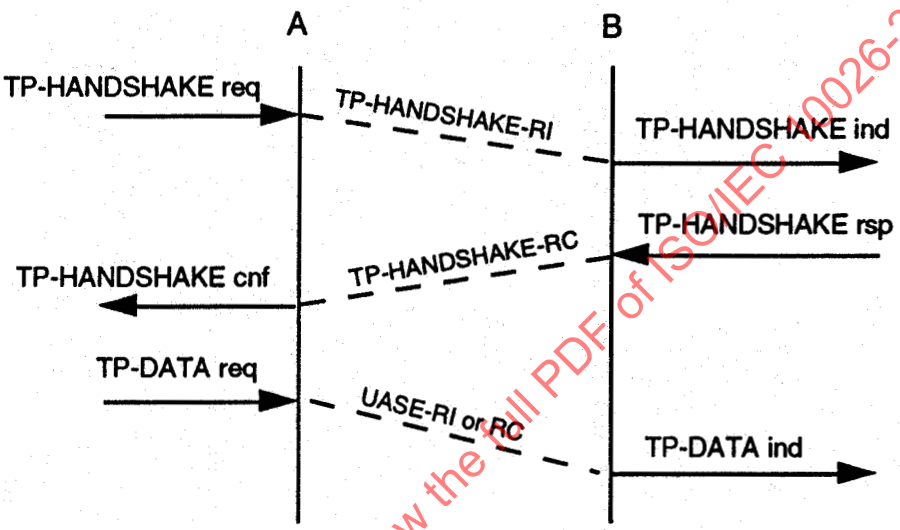


Figure C.9 — TP-HANDSHAKE, immediate response

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

- Figure C.10: when TP-HANDSHAKE is used in Shared Control or when TP-HANDSHAKE-AND-GRANT-CONTROL is used, it is a local TPPM decision whether to concatenate the handshake response with other services. This could result in a substantial delay as shown by case (B). The Confirmation-Urgency parameter allows the TPSUI to specify that the handshake confirm should be issued without delay as illustrated in case (A).

NOTE — Figure C.10 applies equally to TP-HANDSHAKE-AND-GRANT-CONTROL in Polarized Control.

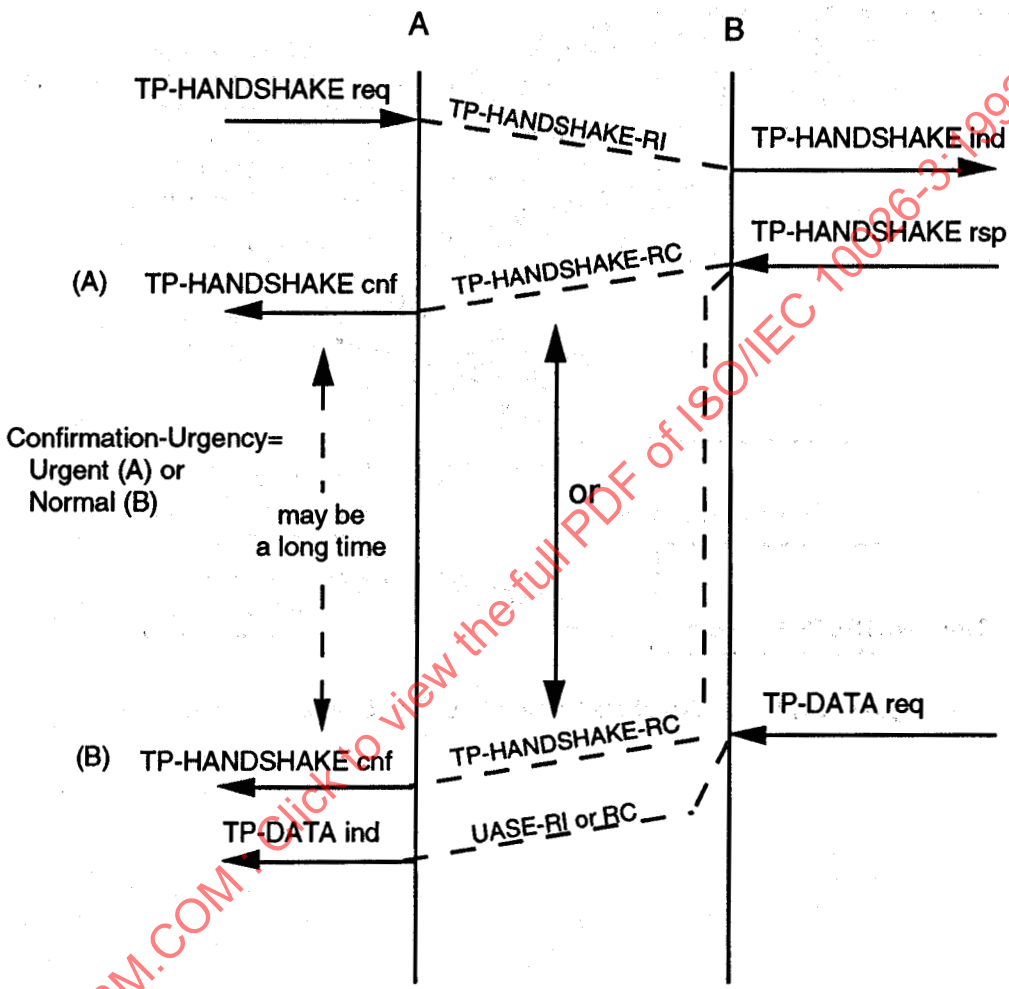


Figure C.10 — TP-HANDSHAKE, delayed response

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.2.6 Deferred End Dialogue service

The scenario in figure C.11 describes a sequence of primitives in the normal case of a TP-DEFERRED-END-DIALOGUE.

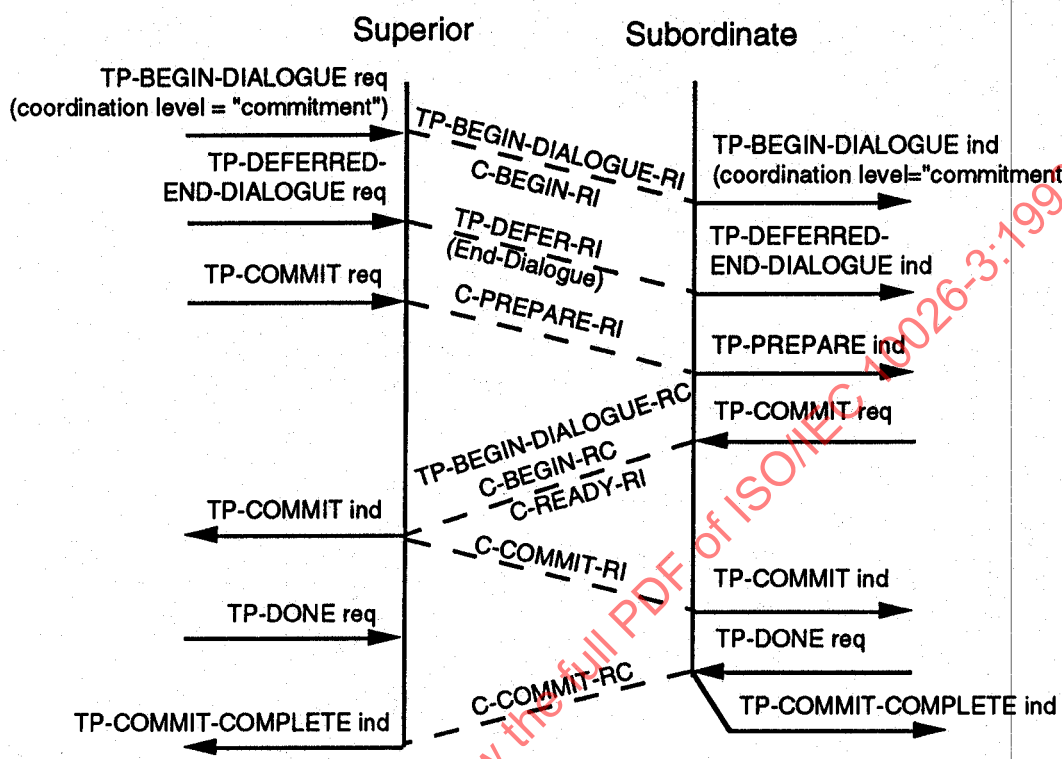


Figure C.11 — TP-DEFERRED-END-DIALOGUE, normal case

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3 Scenarios with a single dialogue (unsuccessful cases)

C.3.1 Negative response to a TP-HANDSHAKE

C.3.1.1 Negative response to a TP-HANDSHAKE in Polarized Control

The scenario in figure C.12 describes a sequence of primitives in the case when a TP-DATA request is followed by a TP-HANDSHAKE request and the TP-HANDSHAKE is negatively responded to in Polarized Control.

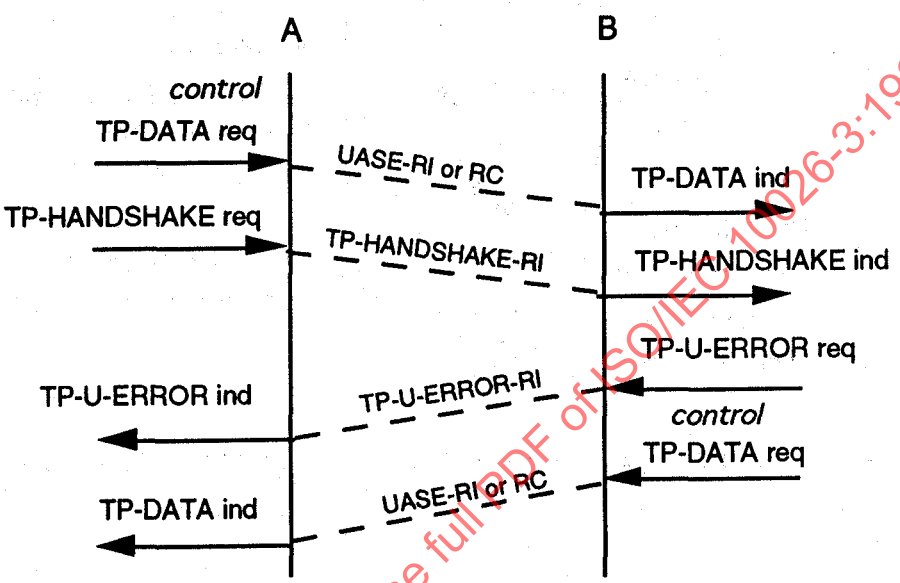


Figure C.12 — Negative response to a handshake request in Polarized Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

A collision between a TP-HANDSHAKE request and a TP-U-ERROR request is treated as an early negative response to TP-HANDSHAKE as shown in the scenario of figure C.13.

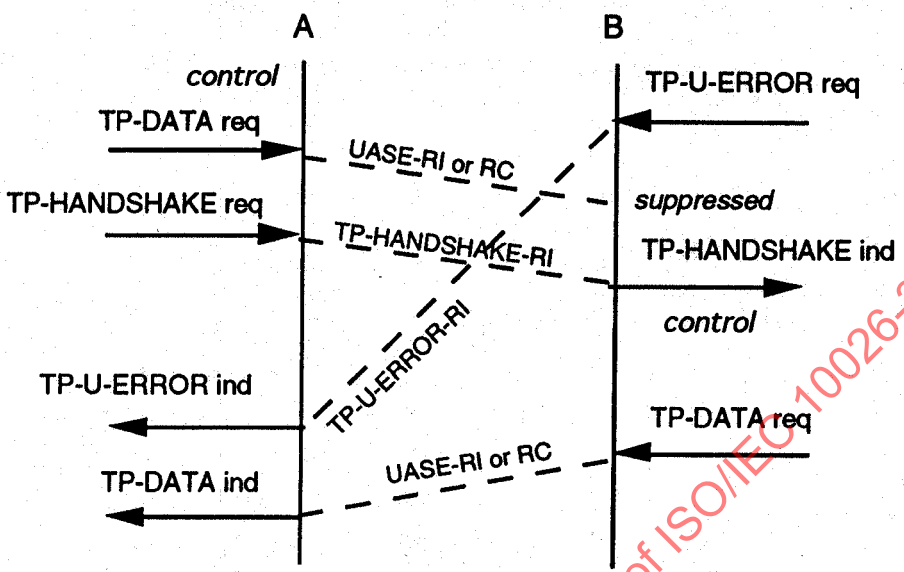


Figure C.13 — Anticipated negative response to a TP-HANDSHAKE in Polarized Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.1.2 Negative response to a TP-HANDSHAKE in Shared Control

The scenario in figure C.14 describes a sequence of primitives in the case when a TP-DATA exchange between two TPSUIs is followed by a TP-HANDSHAKE request that is negatively responded to in Shared Control. The simple negative response to TP-HANDSHAKE in Shared Control is shown below.

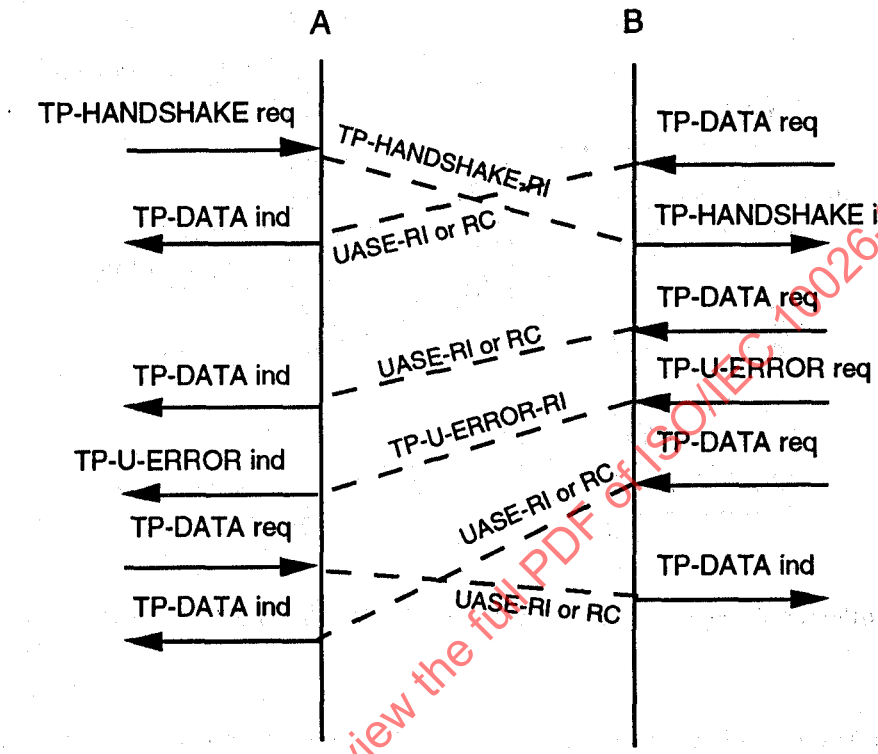


Figure C.14 — Negative response to a TP-HANDSHAKE in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

A collision between a TP-HANDSHAKE request and a TP-U-ERROR request is treated as an early negative response to TP-HANDSHAKE in Shared Control as shown in the scenario in figure C.15. The TP-HANDSHAKE indication is suppressed because B would not know if the corresponding request had been issued before the TP-U-ERROR indication (and the TP-U-ERROR request would have been an anticipated negative response to the handshake) or after the TP-U-ERROR (and the TP-HANDSHAKE is not related to TP-U-ERROR).

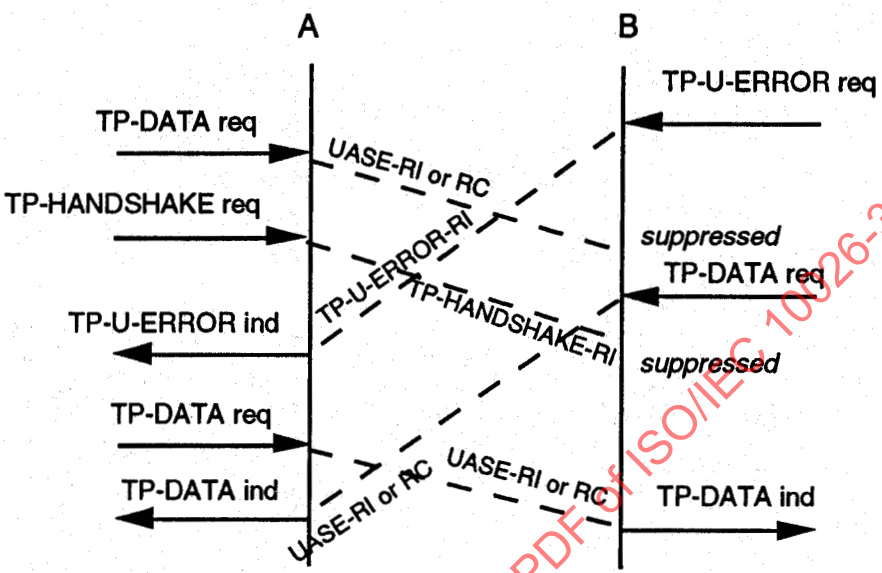


Figure C.15 — Anticipated negative response to a TP-HANDSHAKE request in shared Control

NOTE — In this scenario, TP-HANDSHAKE may be replaced by TP-END-DIALOGUE with the Confirmation parameter set to "true".

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.2 TP-ROLLBACK scenarios

C.3.2.1 TP-ROLLBACK with Chained Transactions

The scenario in figure C.16 describes a sequence of primitives in the case when a transaction is rolled back by one of the TPSUIs issuing a TP-ROLLBACK request during the active phase. The TPSUIs each immediately begin a new transaction.

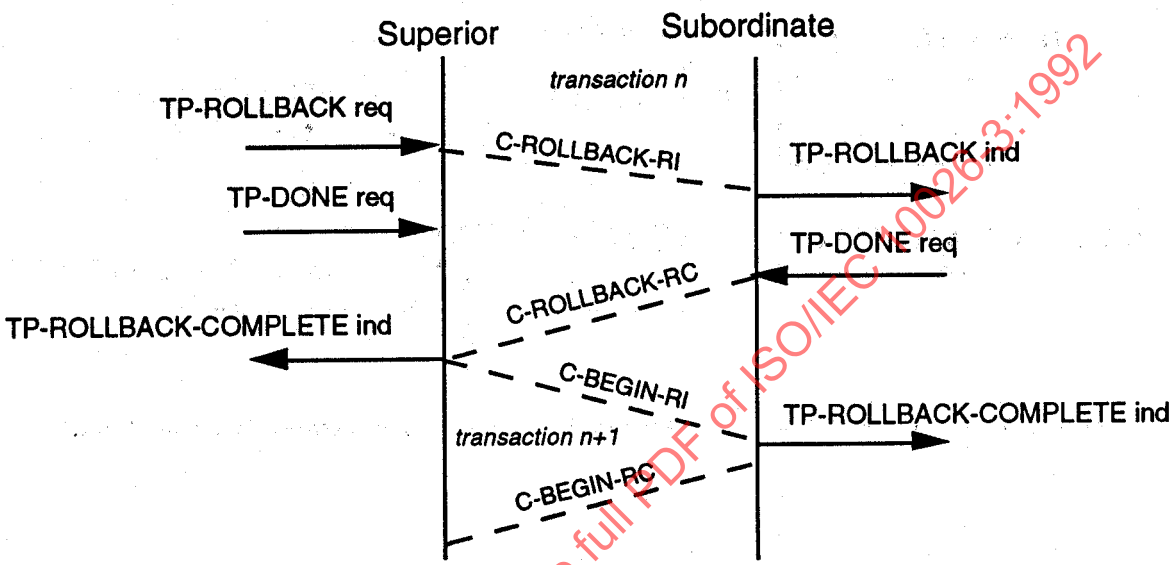


Figure C.16 — TP-ROLLBACK with the Chained Transactions functional unit

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.2.2 TP-ROLLBACK with Unchained Transactions

The scenario for TP-ROLLBACK with unchained transactions, shown in figure C.17, is identical to the scenario for chained transactions except that a new transaction is not begun until a TP-BEGIN-TRANSACTION request has been issued.

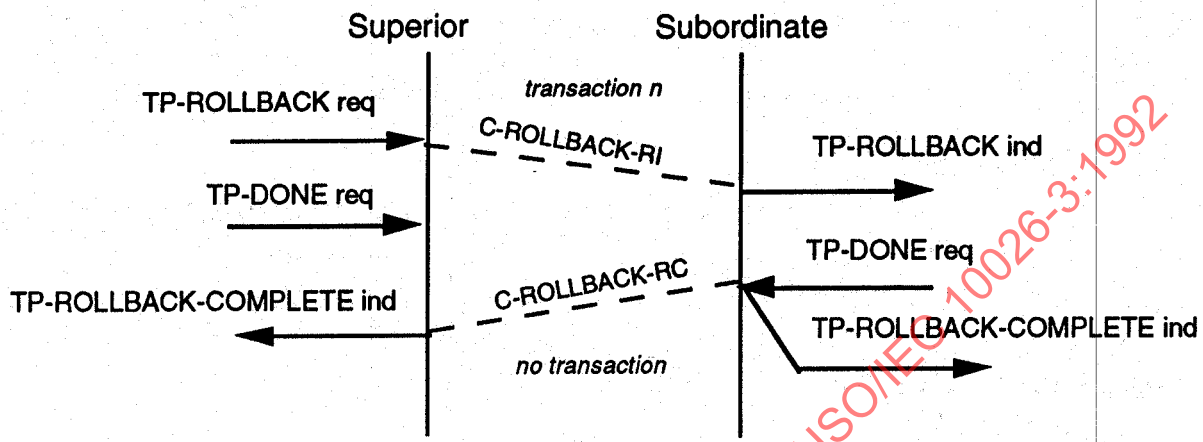


Figure C.17 — TP-ROLLBACK with the Unchained Transactions functional unit

C.3.3 Rejection of a TP-BEGIN-TRANSACTION request

The scenario in figure C.18 describes a situation in which an attempt to include a subordinate TPSUI in the current transaction failed because the subordinate is already participating in another provider-supported transaction. TP-BEGIN-TRANSACTION is rejected, the dialogue between A and B is aborted; the transactions are not rolled back.

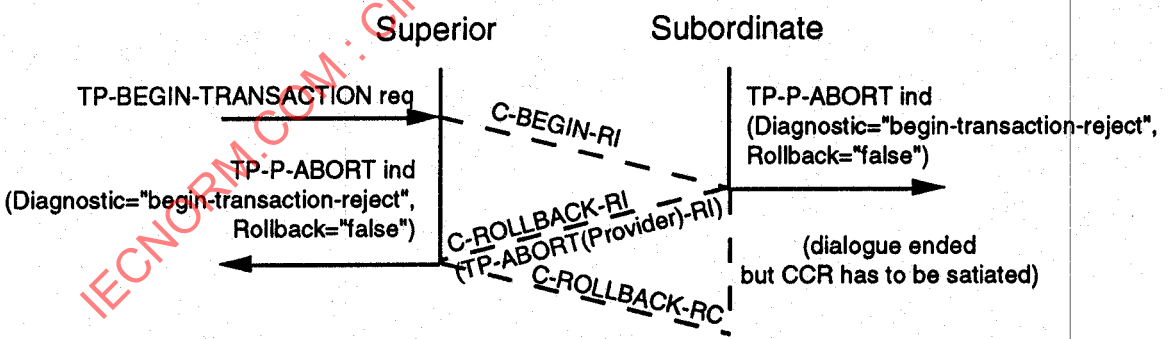


Figure C.18 — Rejection of a TP-BEGIN-TRANSACTION request

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.4 TP-U-ERROR collision with TP-DATA

C.3.4.1 TP-U-ERROR collision with TP-DATA in Polarized Control

The scenario in figure C.19 describes a sequence of primitives in the case of a collision between a TP-DATA request and a TP-U-ERROR request in Polarized Control. TPSUI B is not allowed to issue a TP-DATA request until the TP-GRANT-CONTROL indication is received.

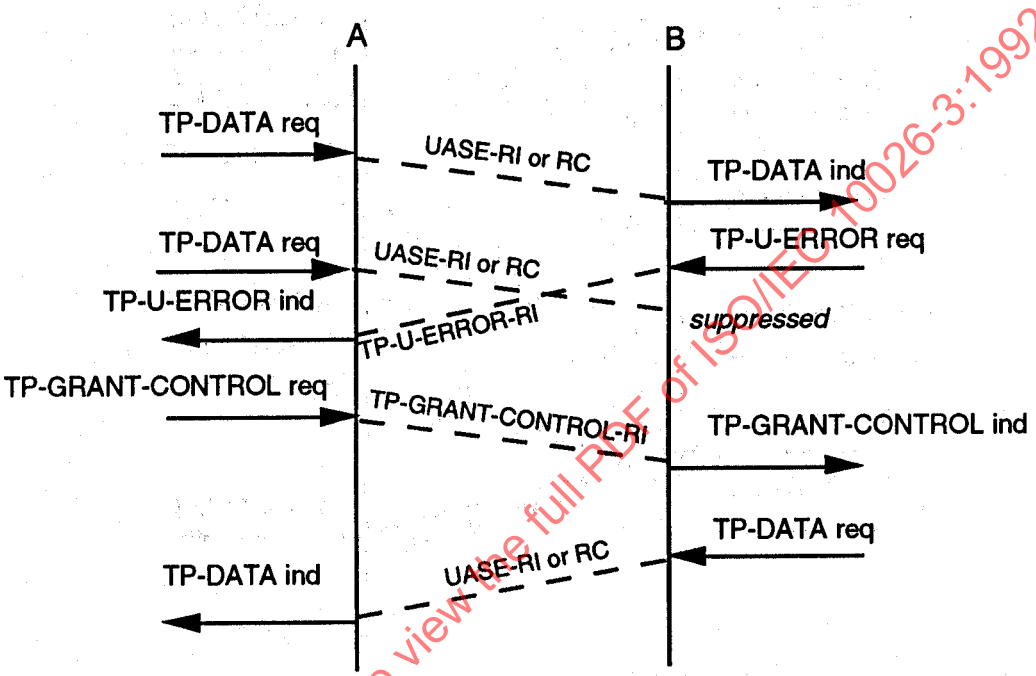


Figure C.19 — TP-U-ERROR collision with TP-DATA in Polarized Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.4.2 TP-U-ERROR Collision with TP-DATA in Shared Control

The scenario of figure C.20 describes a sequence of primitives in the case when a TP-DATA exchange between two TPSUIs is followed by a collision between a TP-DATA request and a TP-U-ERROR request in Shared Control.

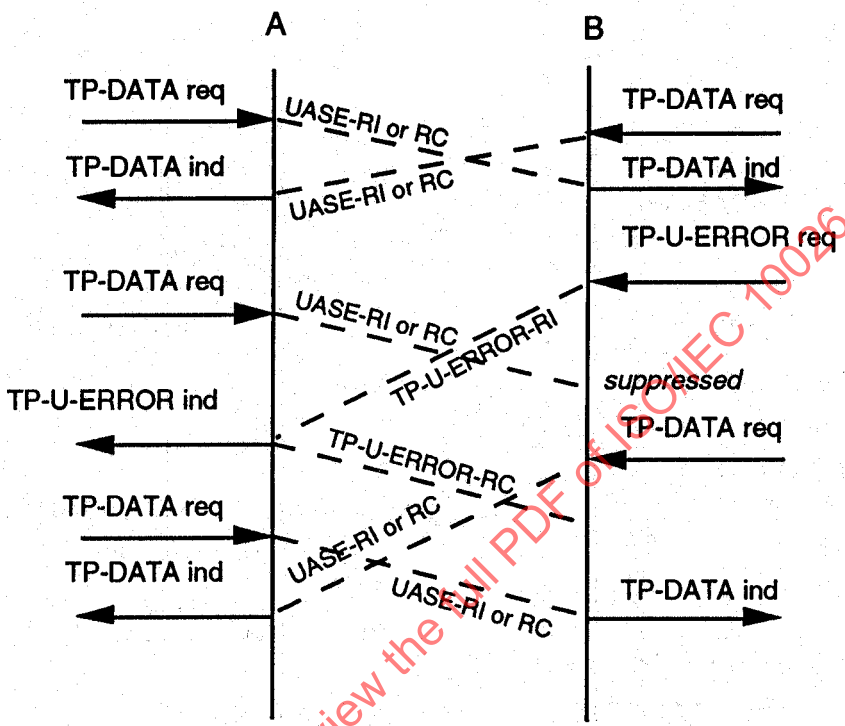


Figure C.20 — TP-U-ERROR collision with TP-DATA in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.5 TP-ROLLBACK with TP-U-ABORT

The scenario in figure C.21 describes a sequence of primitives in the case when a TP-ROLLBACK indication is followed by a TP-U-ABORT request to abort the dialogue.

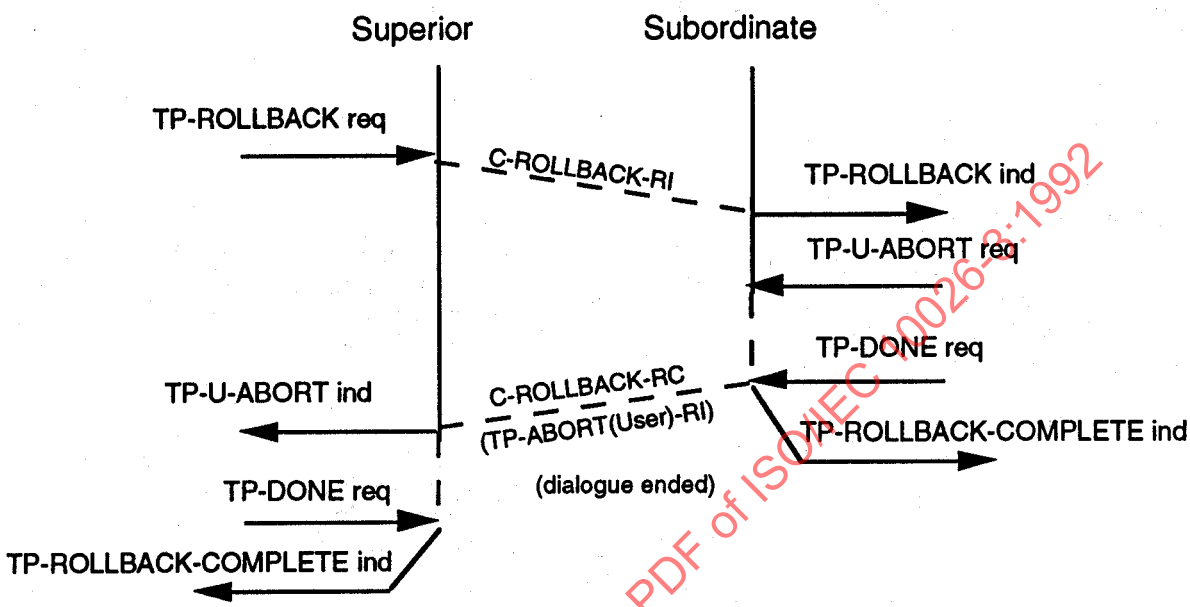


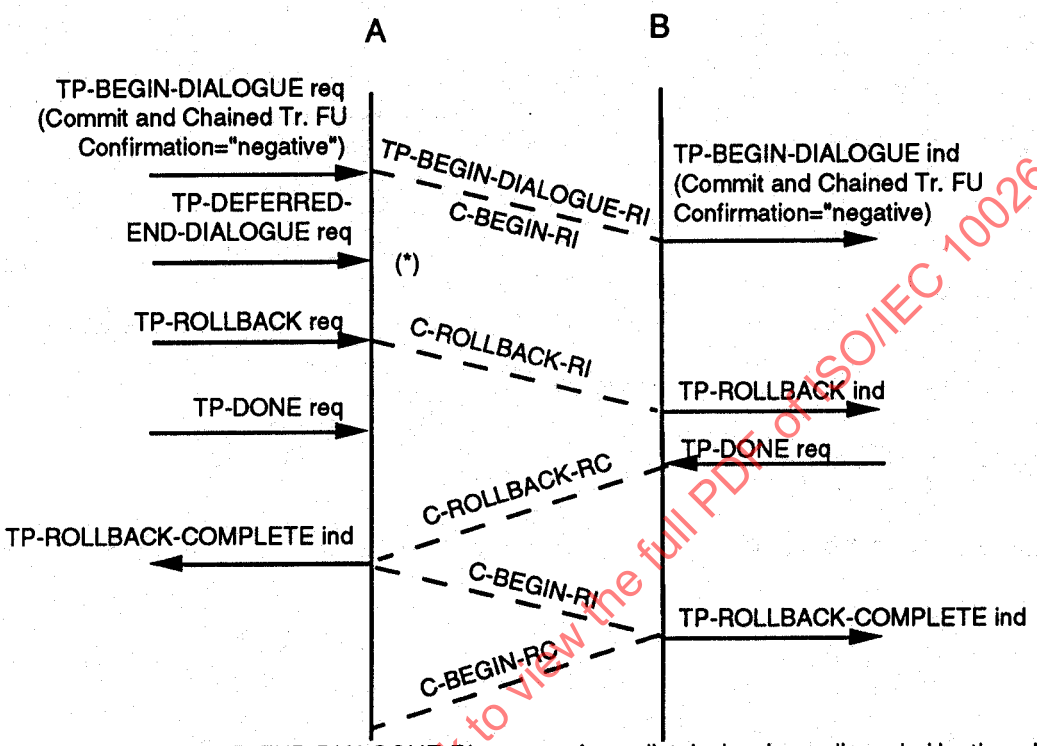
Figure C.21 — TP-U-ABORT in response to a TP-ROLLBACK indication

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.6 TP-DEFERRED-END-DIALOGUE with TP-ROLLBACK

C.3.6.1 TP-ROLLBACK request issued during the active phase

The scenario of figure C.22 describes a sequence of primitives in the case when a TP-DEFERRED-END-DIALOGUE request is cancelled by a TP-ROLLBACK request during the active phase of a provider-supported transaction. The dialogue is not terminated.



(*) The TP-DEFERRED-END-DIALOGUE-RI, not sent immediately, has been discarded by the rollback procedure

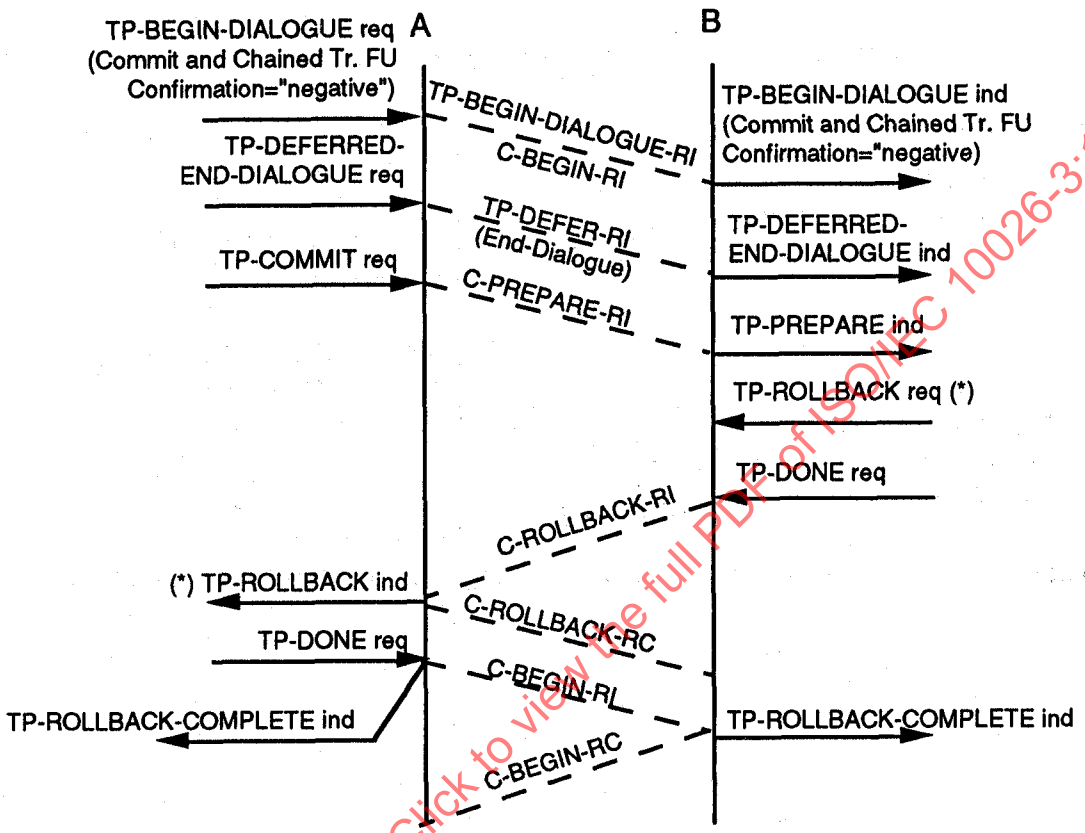
Note : TP-BEGIN-DIALOGUE-RC and C-BEGIN-RC are not required in this case. C-ROLLBACK-RC serves as the confirmation instead.

Figure C.22 — TP-DEFERRED-END-DIALOGUE cancelled by a rollback

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.6.2 TP-ROLLBACK request issued after a TP-PREPARE indication

The scenario of figure C.23 describes a sequence of primitives in the case when a TP-DEFERRED-END-DIALOGUE request is cancelled by a TP-ROLLBACK request during the termination phase of a provider-supported transaction. The dialogue is not terminated.



(*) any other rollback-initiating service primitive would also cancel the effects of the TP-DEFERRED-END-DIALOGUE service.

Note : TP-BEGIN-DIALOGUE-RC and C-BEGIN-RC are not required in this case. C-ROLLBACK-RI serves as the confirmation instead.

Figure C.23 — Effects of a TP-DEFERRED-END-DIALOGUE cancelled by a rollback, after TP-COMMIT request

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7 Dialogue Establishment Scenarios

C.3.7.1 Rejection of the dialogue establishment request

The scenarios of figure C.24, C.25, and C.26 describe a sequence of primitives in the case when dialogue establishment is requested with a TP-BEGIN-DIALOGUE request, with either Confirmation="negative" or Confirmation="always", but is rejected by the recipient.

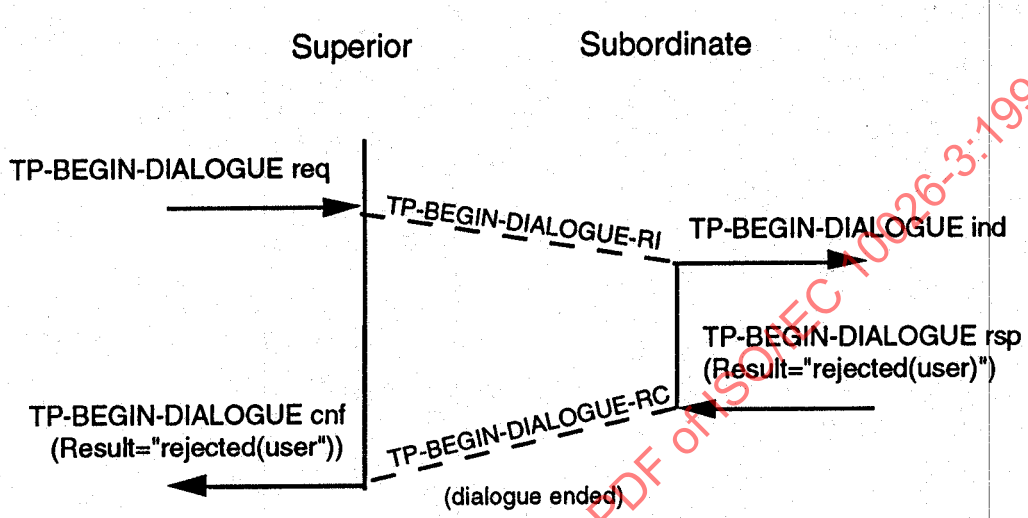


Figure C.24 — Rejection of a TP-BEGIN-DIALOGUE, coordination level "none"

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

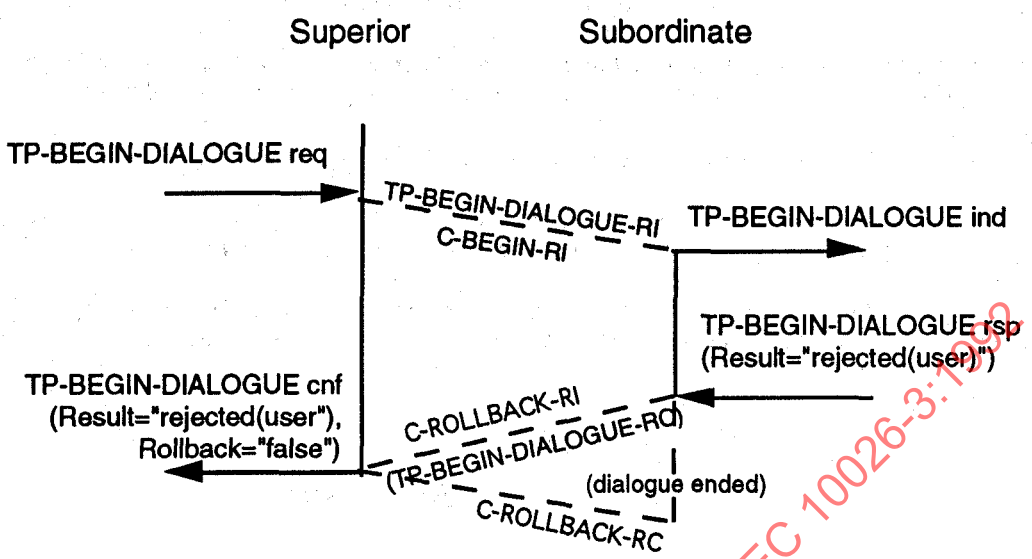


Figure C.25 — Rejection of a TP-BEGIN-DIALOGUE, coordination level "commitment"

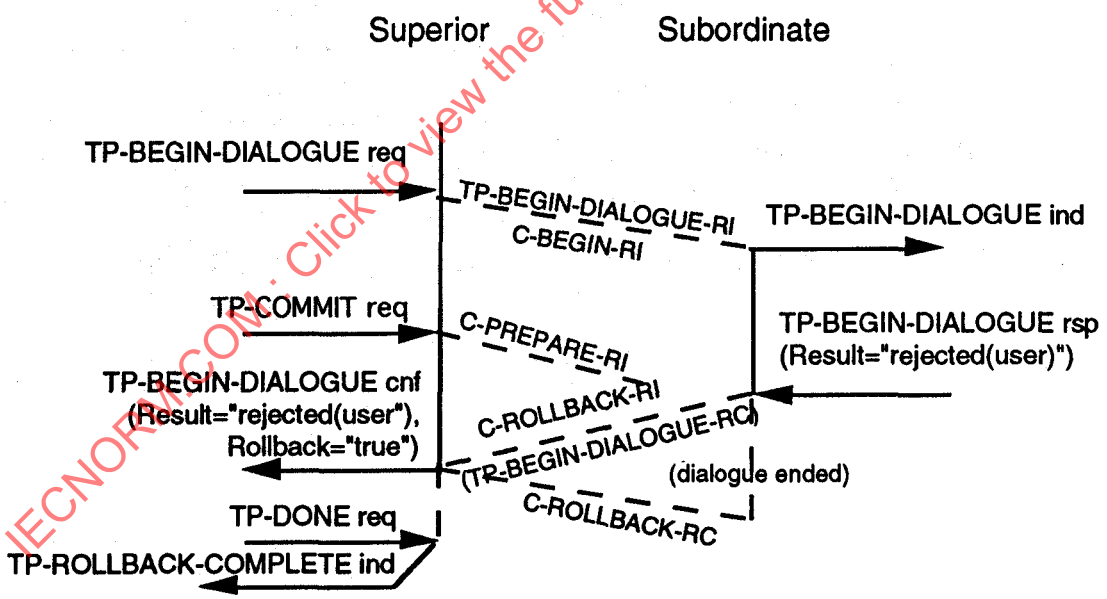


Figure C.26 — Rejection of a TP-BEGIN-DIALOGUE, coordination level "commitment" (TP-COMMIT request issued)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.2 TP-BEGIN-DIALOGUE (Confirmation="negative") scenarios

The following scenarios apply to provider-supported transaction and describe various dialogue establishment sequences using the TP-BEGIN-DIALOGUE (Confirmation=negative) service related to aborting, rejecting, and failure cases.

C.3.7.2.1 Aborting the dialogue establishment by the requestor

The scenario of figure C.27 describes a sequence of primitives in the case when dialogue establishment is requested with a TP-BEGIN-DIALOGUE (Confirmation="negative") request and is then aborted by the requestor with a TP-U-ABORT request.

When a TP-U-ABORT indication is received following a TP-BEGIN-DIALOGUE (Confirmation="negative") indication, the TP-U-ABORT indication will carry the Rollback parameter set to "true" and the recipient TPSUI must issue a TP-DONE request to complete the rollback action.

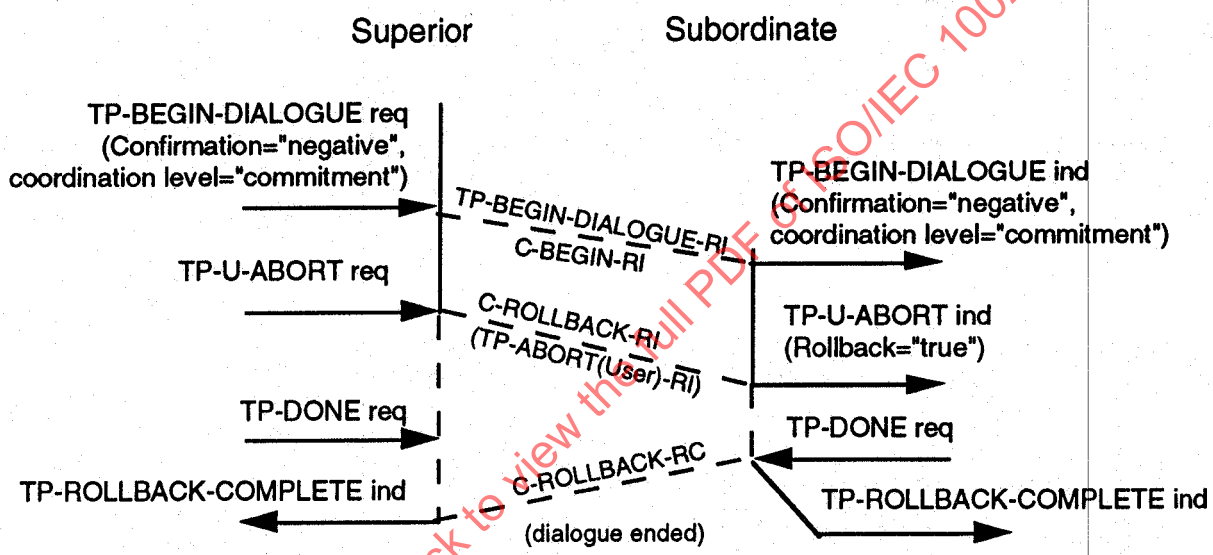


Figure C.27 — Abort of a dialogue establishment by the requestor (Confirmation = "negative")

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.2.2 Dialogue establishment followed by rollback from the requestor

The following scenarios describe two sequences of primitives for the cases when dialogue establishment is attempted with a TP-BEGIN-DIALOGUE (Confirmation="negative") request and then rolled back by the requestor with a TP-ROLLBACK request.

As shown in figure C.28, when a TP-ROLLBACK indication is received following the receipt of a TP-BEGIN-DIALOGUE (Confirmation="negative") indication and prior to issuing any other services on the specific dialogue, a negative response may be issued with a TP-BEGIN-DIALOGUE (Result="rejected(user)") response.

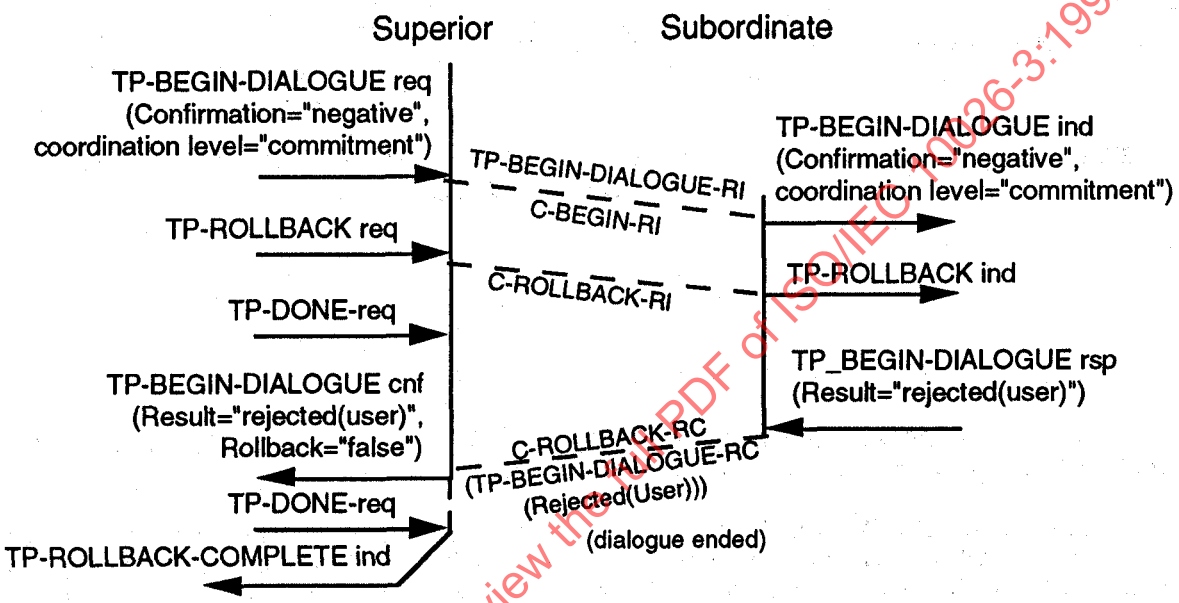


Figure C.28 — Dialogue establishment followed by rollback from the requestor, recipient rejects the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Otherwise, the recipient must issue a TP-DONE request to complete the rollback action, as shown in figure C.29.

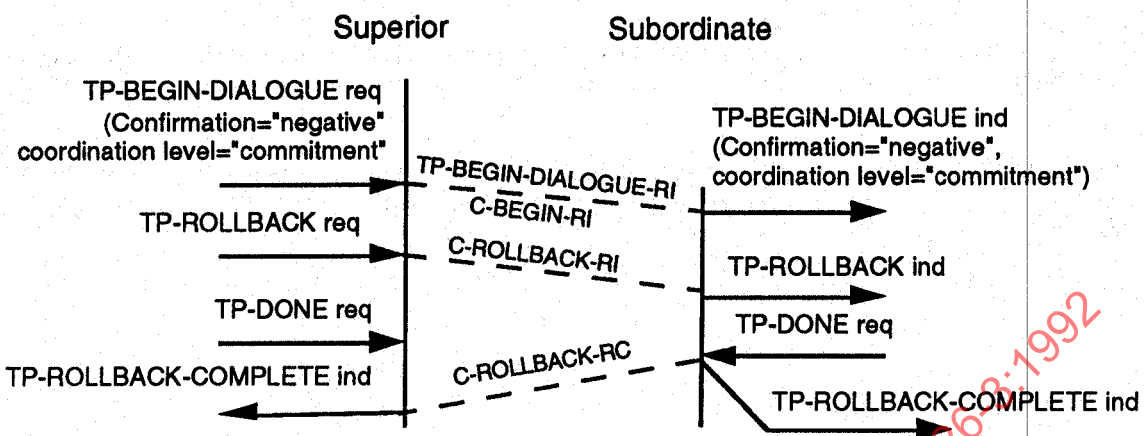


Figure C.29 — Dialogue establishment followed by rollback from the requestor, recipient accepts the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.2.3 Failure after Dialogue Establishment

The scenario of figure C.30 describes a sequence of primitives in the case when dialogue establishment is attempted with a TP-BEGIN-DIALOGUE (Confirmation="negative") request, followed by a failure.

In case of a dialogue failure, a TP-P-ABORT indication with the Rollback parameter set to "true" will be issued to both TPSUIs. When the TP-P-ABORT indication is issued to the superior TPSUI before the TP-BEGIN-DIALOGUE confirm has been received, the TP-P-ABORT indication will have the Rollback parameter set to "true" because it is not known whether the subordinate accepted or rejected the dialogue initialization request.

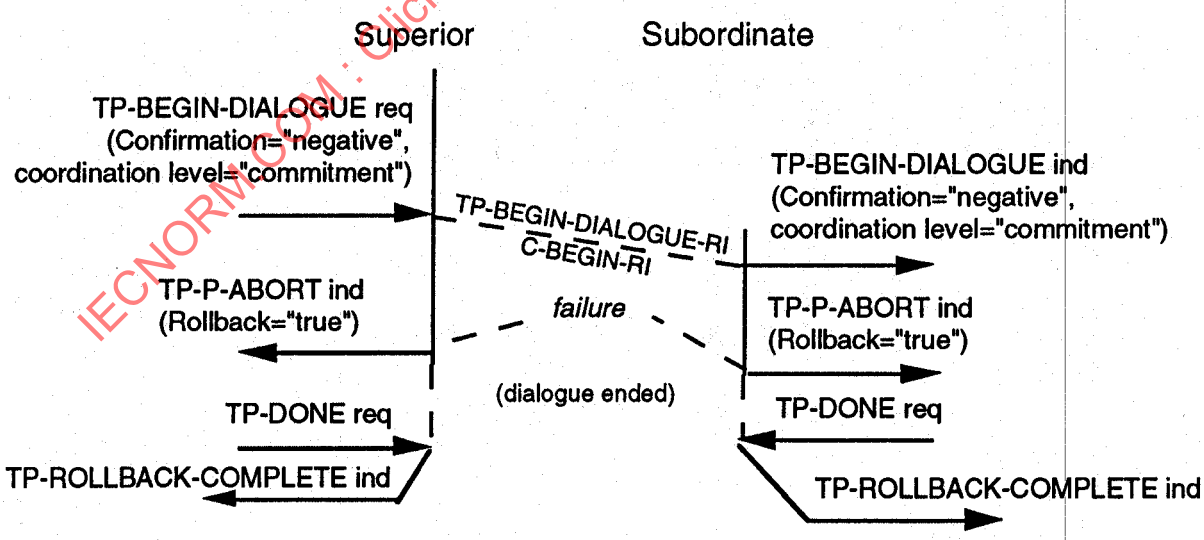


Figure C.30 — Dialogue establishment (Confirmation="negative") followed by dialogue failure; recipient does not reject the dialogue

C.3.7.2.4 Dialogue establishment rejected after TP-COMMIT request

Figure C.31 shows how dialogue establishment rejection received after a TP-COMMIT request, signals that the transaction is being rolled back.

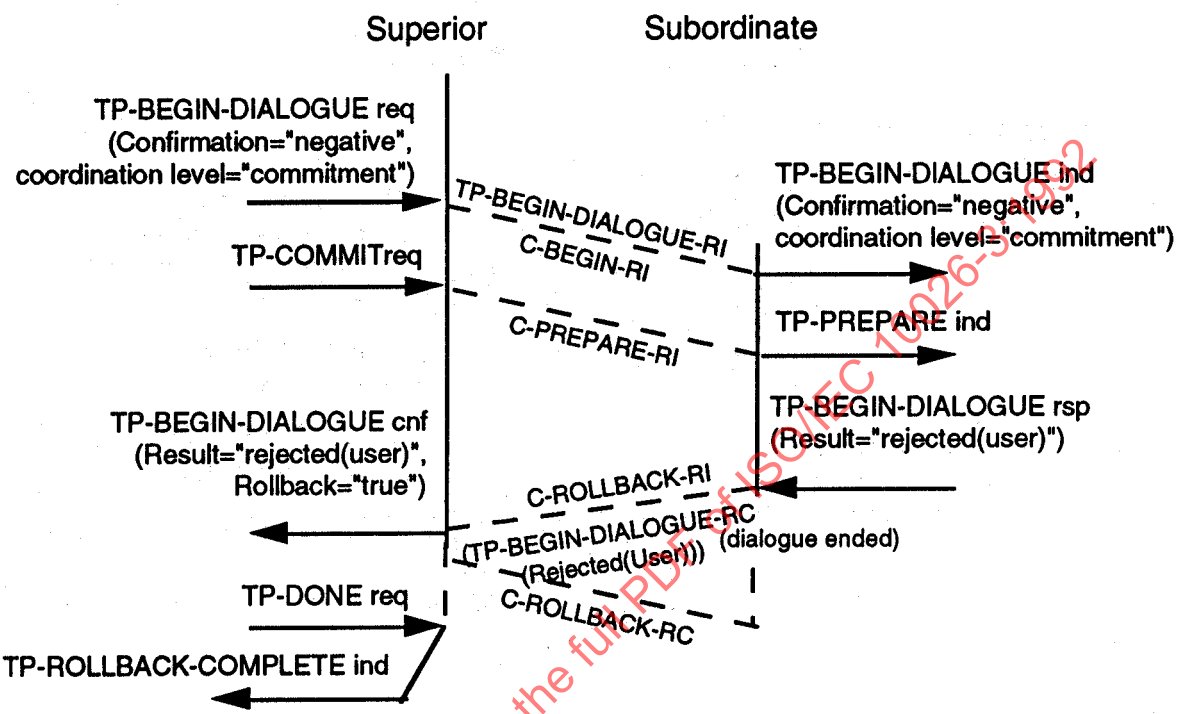


Figure C.31 — Rejection of a Dialogue establishment request causing rollback

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.2.5 Dialogue establishment rejected after TP-COMMIT request

Figure C.32 shows how dialogue establishment rejection received after a TP-PREPARE request does not cause rollback of the transaction.

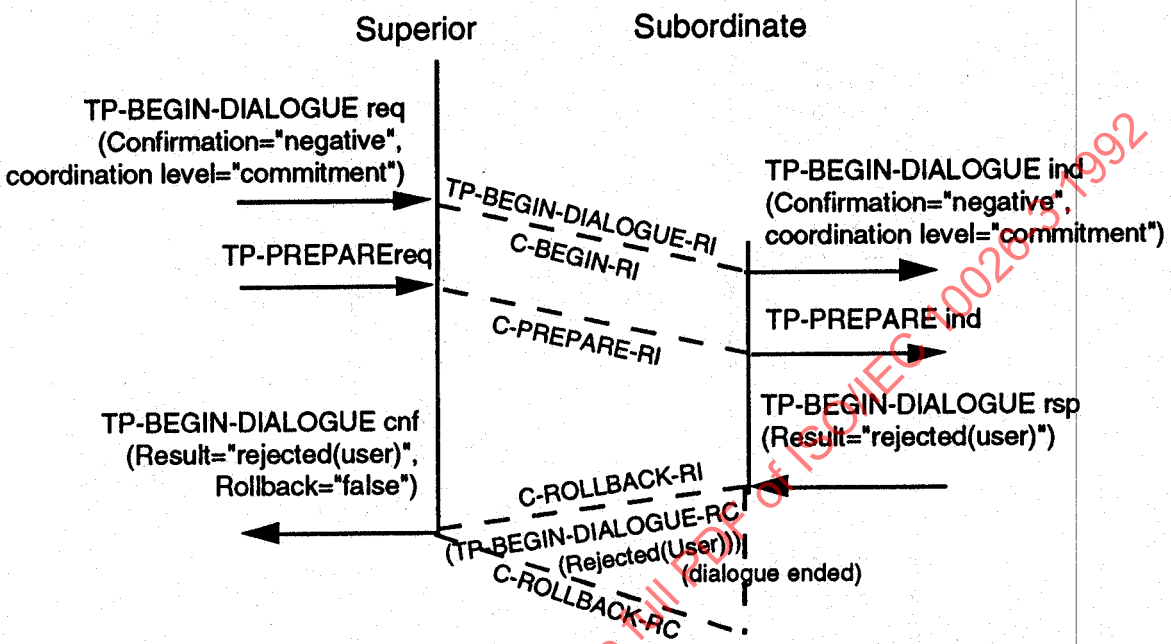


Figure C.32 — Rejection of a Dialogue establishment request after TP-PREPARE

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.3 TP-BEGIN-DIALOGUE (Confirmation="always") scenarios

The following scenarios describe various dialogue establishment sequences using the TP-BEGIN-DIALOGUE (Confirmation="always") service related to aborting, rejecting, and failure cases.

C.3.7.3.1 Aborting the dialogue establishment

The scenario of figure C.33 describes sequences of primitives for the case when dialogue establishment is attempted with a TP-BEGIN-DIALOGUE (Confirmation="always") request and then aborted, by the requestor, with a TP-U-ABORT request.

When a TP-U-ABORT indication is received following a TP-BEGIN-DIALOGUE (Confirmation="always") indication and before the indication has been responded to, the TP-U-ABORT indication will carry the Rollback parameter set to "false", and no further actions are required by the subordinate.

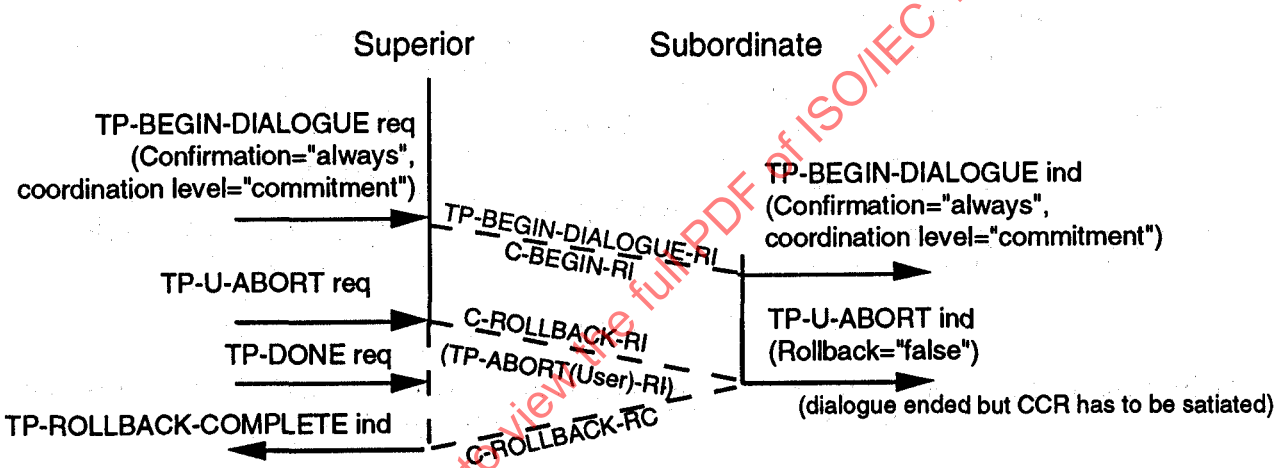


Figure C.33 — Abort of dialogue establishment by the requestor (Confirmation="always"), before the recipient has accepted the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

When a TP-U-ABORT indication is received following a TP-BEGIN-DIALOGUE (Confirmation="always") indication but after the indication has been positively responded to, the TP-U-ABORT indication will carry the Rollback parameter set to "true", and the subordinate must complete normal rollback actions, as shown in figure C.34.

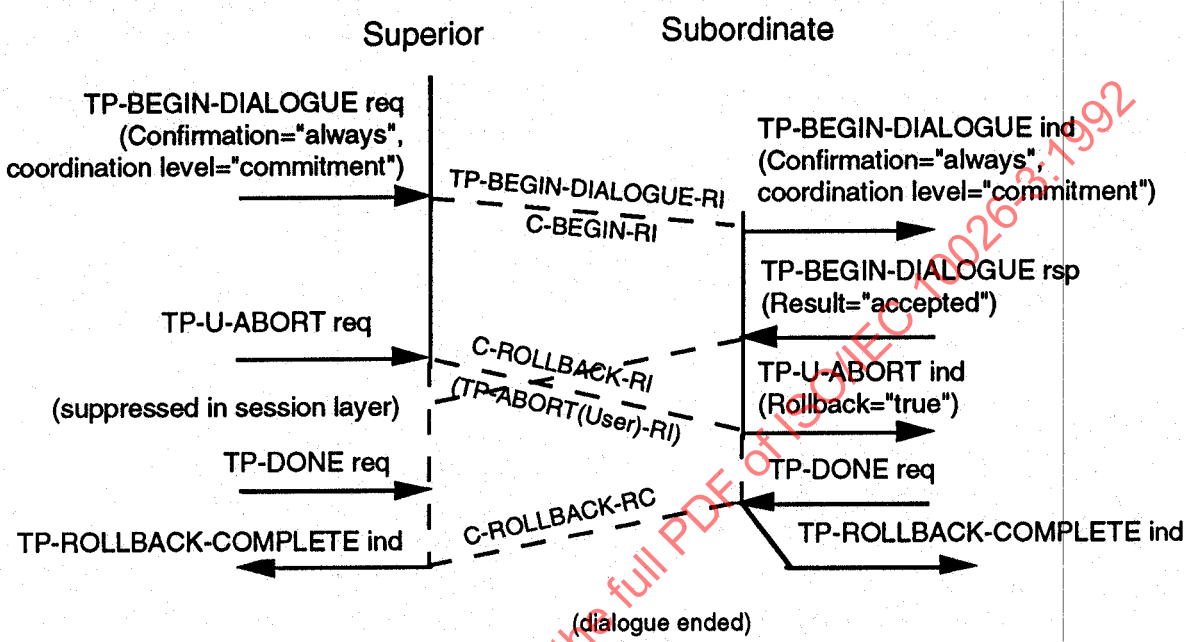


Figure C.34 — Abort of a dialogue establishment (Confirmation="always") by the requestor, after the recipient has accepted the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario in figure C.35 shows that the recipient of a TP-BEGIN-DIALOGUE (Confirmation="always") indication, if it wants to abort the dialogue, shall issue the TP-BEGIN-DIALOGUE response before the TP-U-ABORT request can be issued.

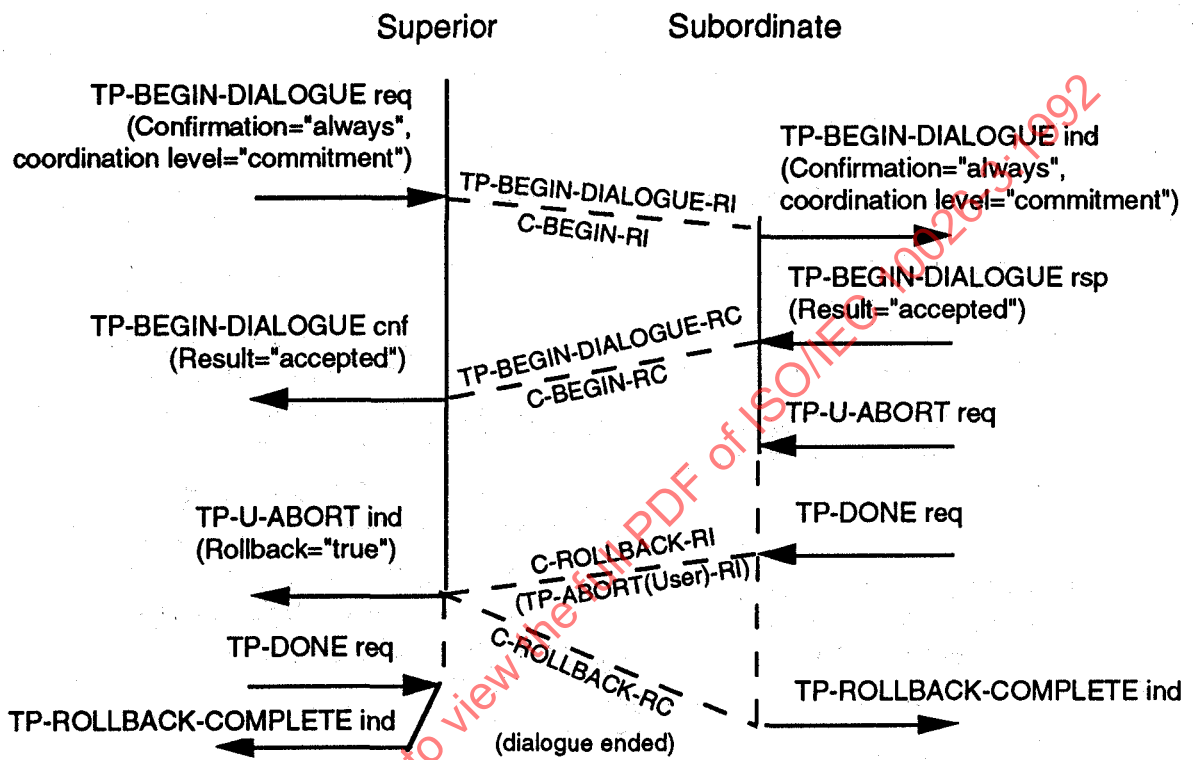


Figure C.35 — Abort of a dialogue establishment (Confirmation="always") by the recipient, after the recipient has accepted the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.3.2 Dialogue establishment followed by rollback from the requestor

These scenarios describe two sequences of primitives in the cases when dialogue establishment is attempted with a TP-BEGIN-DIALOGUE (Confirmation="always") request and then rolled back by the requestor with a TP-ROLLBACK request.

When a TP-ROLLBACK indication is received before a TP-BEGIN-DIALOGUE (Confirmation="always") indication has been responded to, the TP-BEGIN-DIALOGUE response must be issued before the TP-ROLLBACK indication is acted on; a negative response to the TP-BEGIN-DIALOGUE (confirmed) indication completes the subordinate's rollback action, as shown in figure C.36.

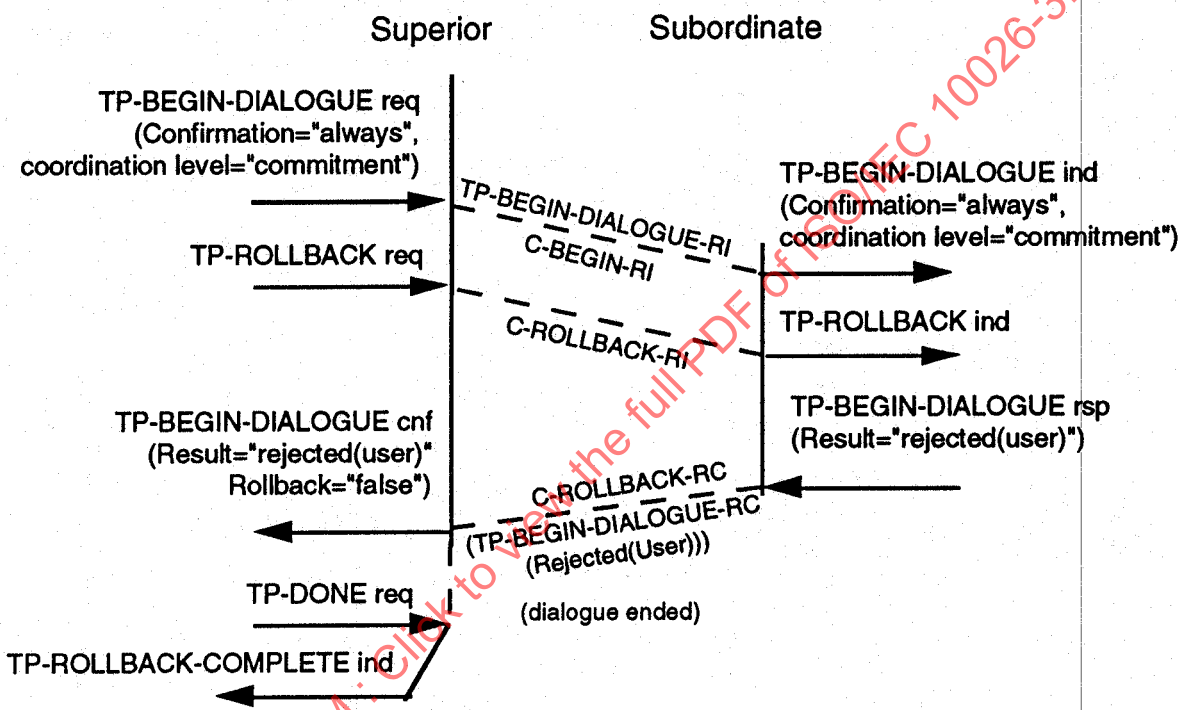


Figure C.36 — Dialogue establishment followed by rollback from the requestor; recipient rejects the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The recipient TPSUI may also issue a positive response to the TP-BEGIN-DIALOGUE (Confirmation="always") indication, followed by a TP-DONE request, as shown in figure C.37.

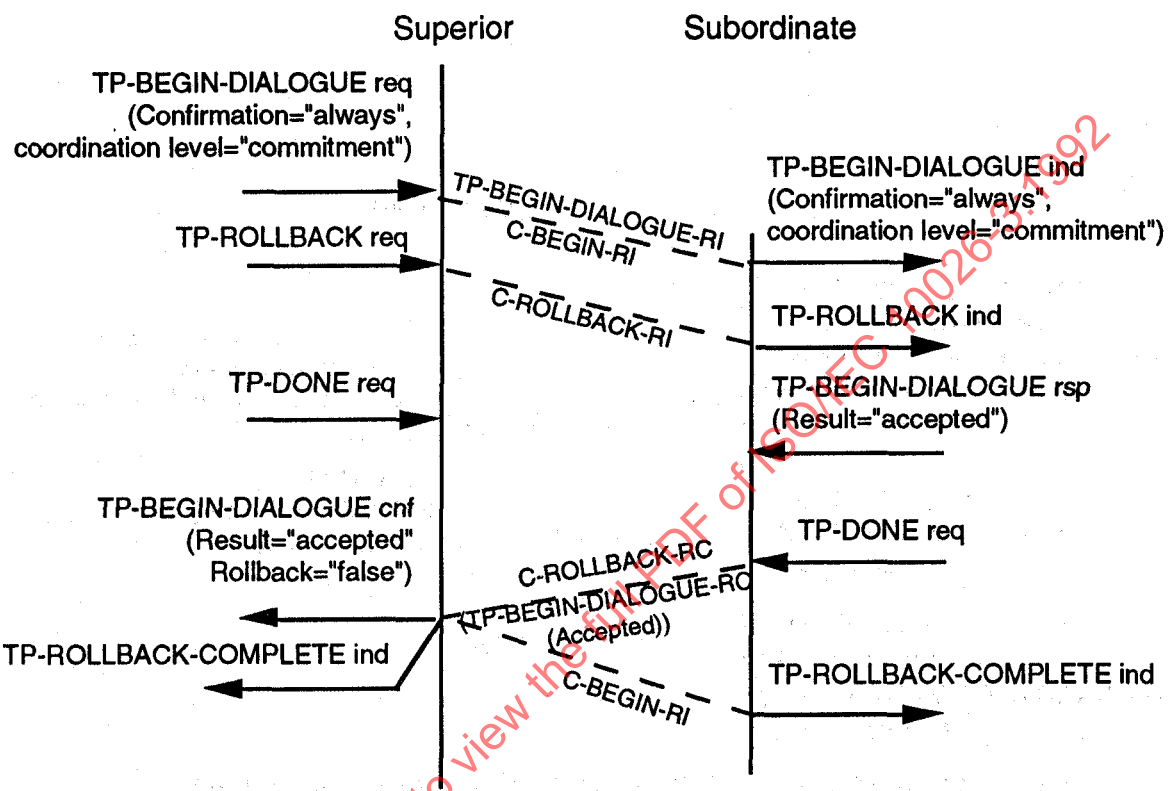


Figure C.37 — Dialogue establishment followed by rollback from the requestor; recipient accepts the dialogue and has to complete the rollback procedure (chained transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.3.3 Failure during dialogue establishment

The scenario of figure C.38 describes a sequence of primitives in the case when dialogue establishment is attempted with a TP-BEGIN-DIALOGUE (Confirmation="always") request, followed by a failure. In case of a dialogue failure, a TP-P-ABORT indication will be issued to the superior with the Rollback parameter set to "true" and to the subordinate with the Rollback parameter set to "false".

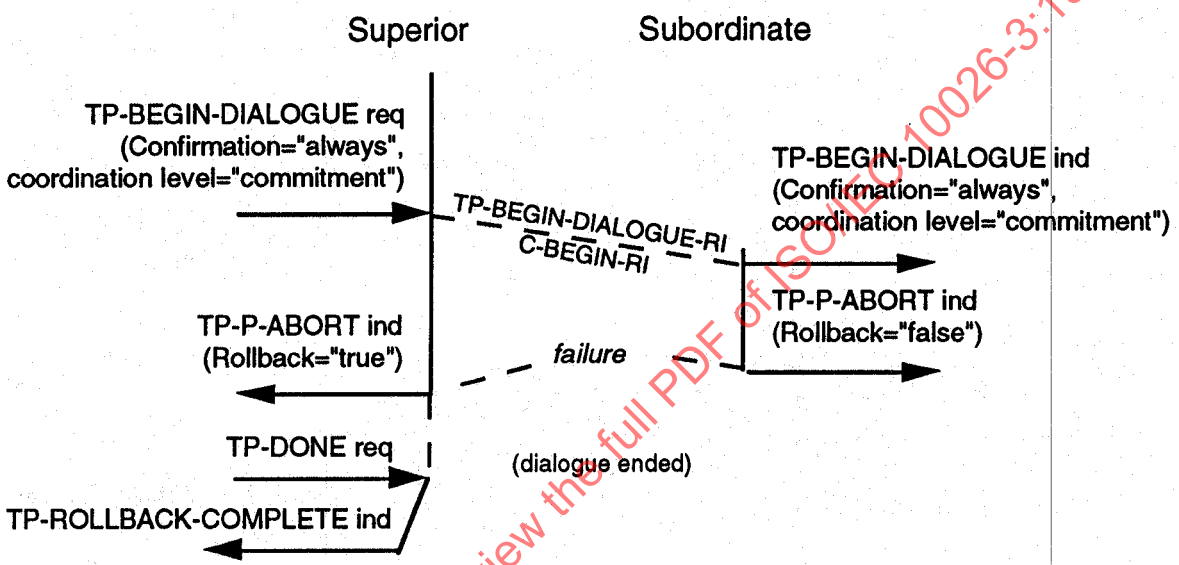


Figure C.38 — Dialogue establishment (Confirmation="always") followed by dialogue failure before the recipient has accepted the dialogue

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

However, if the subordinate has already issued a positive TP-BEGIN-DIALOGUE response, the TP-P-ABORT will carry the Rollback parameter set to "true" and the subordinate shall complete the rollback actions in the usual manner, as shown in figure C.39.

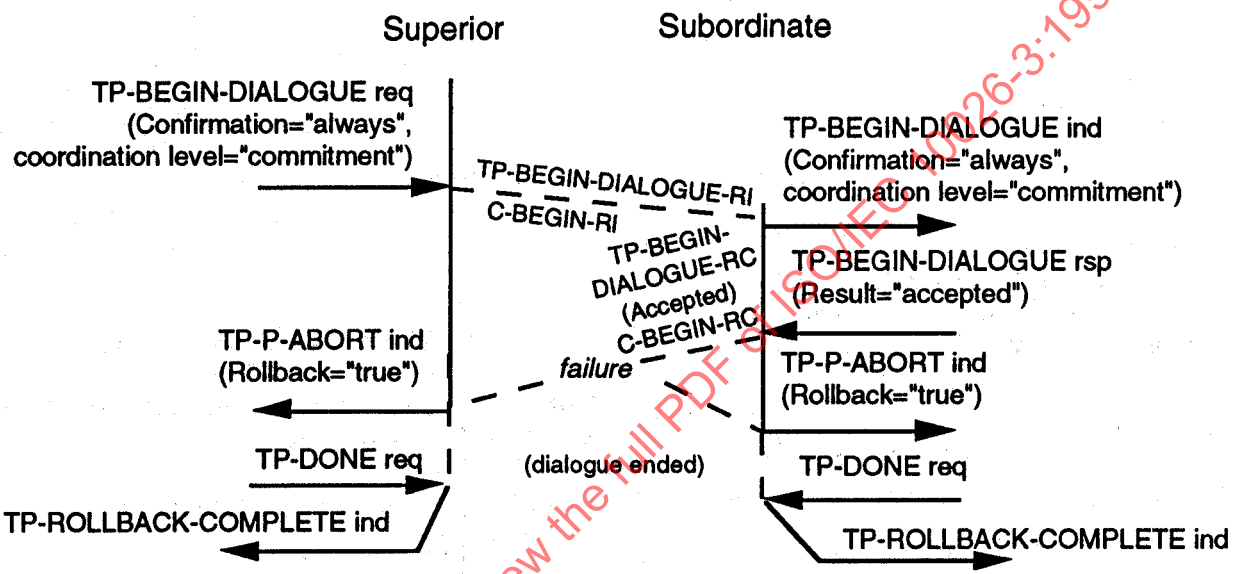


Figure C.39 — Dialogue establishment (Confirmation="always") followed by failure after the recipient has accepted the dialogue; the recipient has to complete the rollback procedure

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.3.7.3.4 Dialogue establishment with rollback and abort

These scenarios cover the case when dialogue establishment is attempted with a TP-BEGIN-DIALOGUE (Confirmation="always") request and then the transaction is rolled back by the requestor with a TP-ROLLBACK request.

Before a TP-BEGIN-DIALOGUE confirm is received, the dialogue is aborted by either a TP-P-ABORT indication (figure C.40) or a TP-U-ABORT request (figures C.41 and C.42).

When a TP-P-ABORT indication is received, subsequent to the TP-ROLLBACK indication, this ends the termination phase of the transaction as shown in figure C.40.

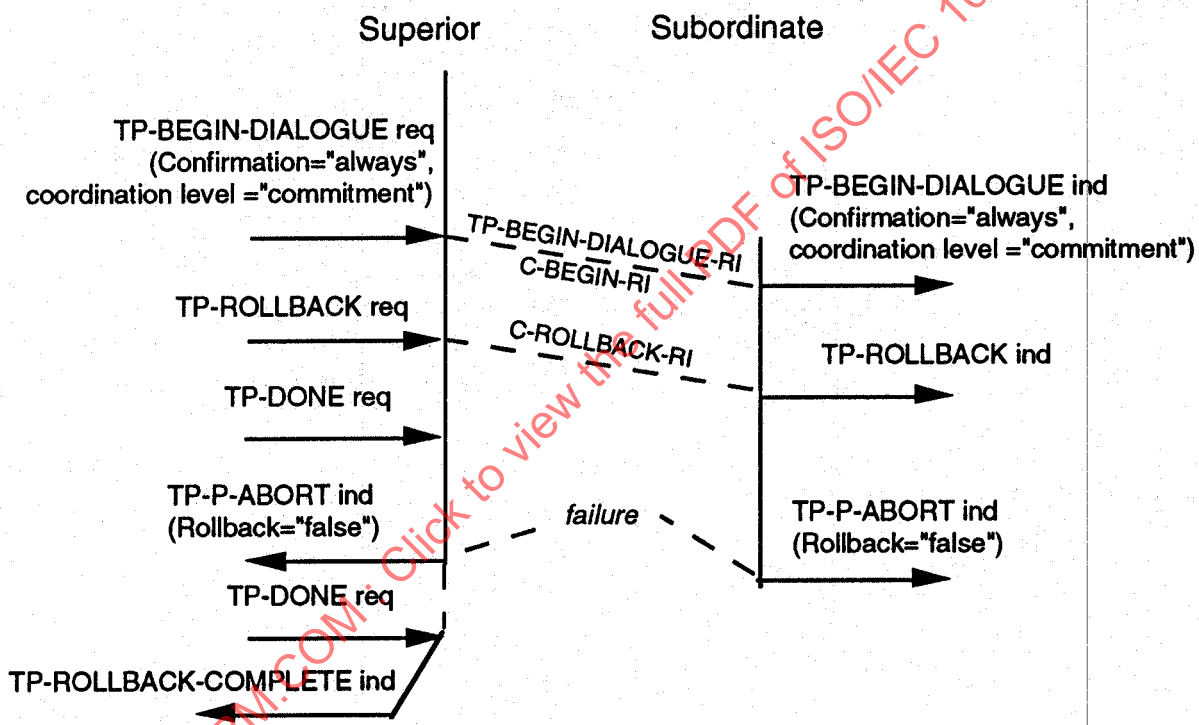


Figure C.40 — Dialogue establishment followed by rollback from the requestor and a subsequent dialogue abort

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario of figure C.41 shows the case where the superior issues a TP-U-ABORT request for the dialogue. Here the TP-U-ABORT indication is withheld until the positive dialogue establishment confirmation is received.

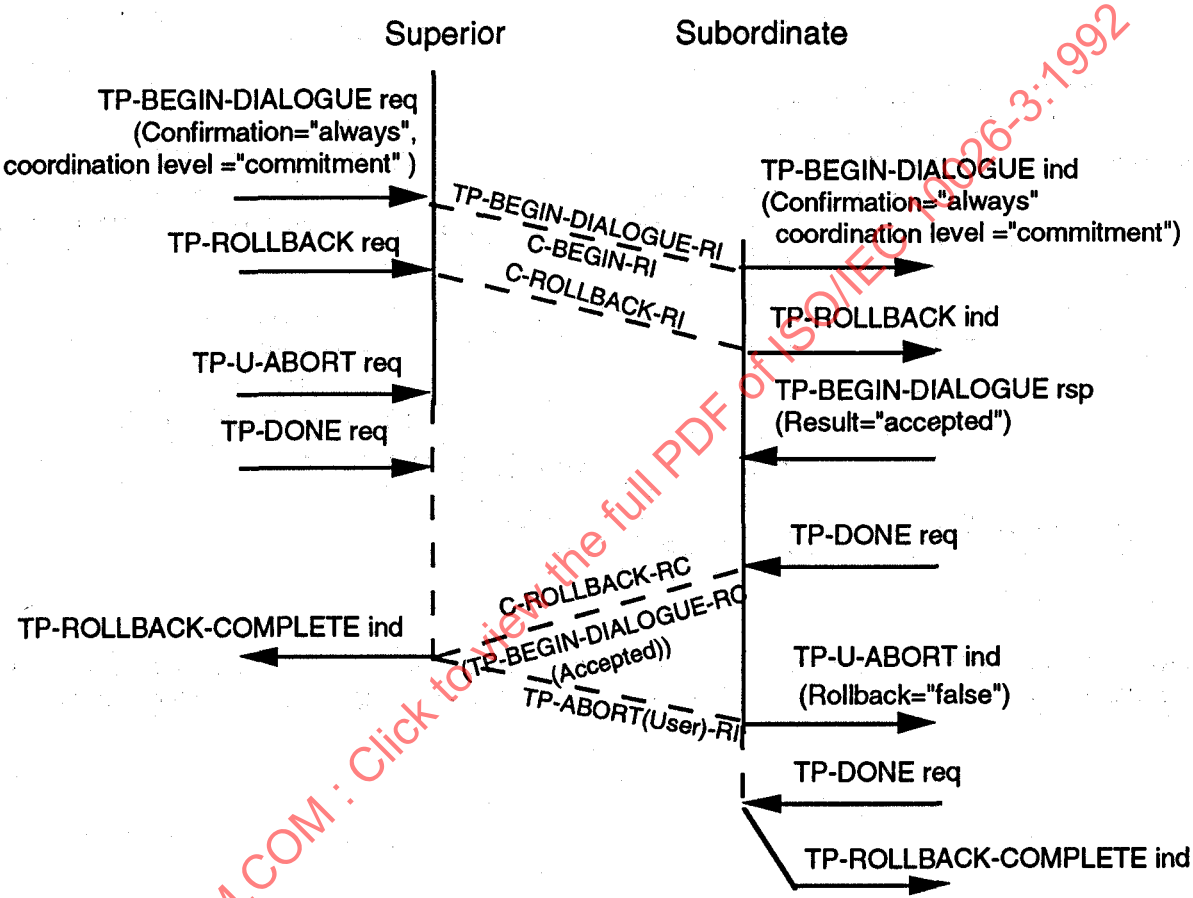


Figure C.41 — Dialogue establishment followed by rollback by the requestor and a subsequent collision of TP-U-ABORT req and TP-BEGIN-DIALOGUE rsp (Result="accepted"). The Chained Transactions functional unit is selected

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario of figure C.42 is similar to that shown on figure C.41 except that the subordinate rejects the dialogue after receiving the TP-ROLLBACK indication.

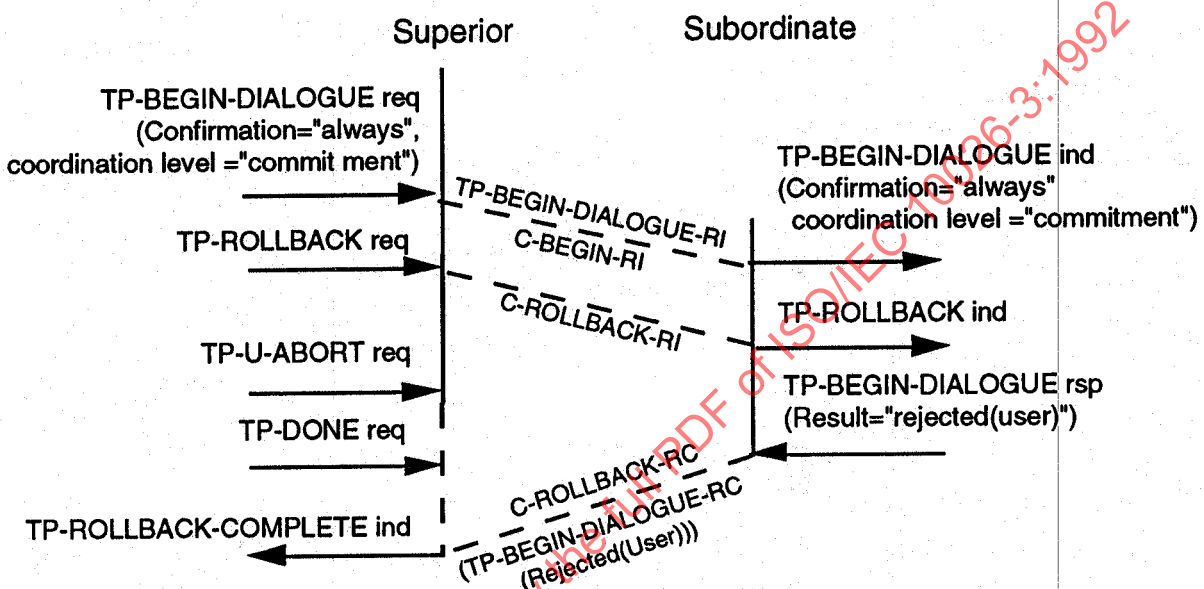


Figure C.42 — Dialogue establishment followed by rollback by the requestor and a subsequent collision of TP-U-ABORT req and TP-BEGIN-DIALOGUE rsp (Result="rejected(user)")

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.4 Scenarios with a single dialogue (failure cases)

C.4.1 TP-P-ABORT

These scenarios apply to dialogues with a coordination level of "commitment" with either chained and unchained transactions unless otherwise specified.

C.4.1.1 TP-P-ABORT during active phase

The scenario of figure C.43 describes a sequence of primitives in the case when a provider-supported transaction is aborted by a communications failure of the dialogue during the active phase.

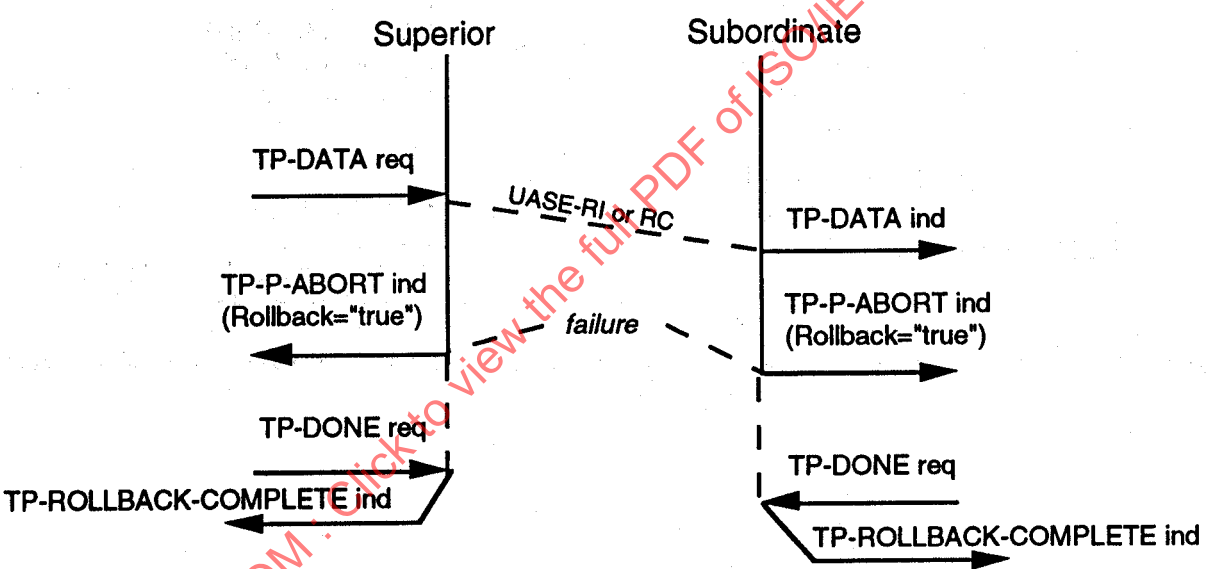


Figure C.43 — Failure for a provider-supported transaction in the active phase

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.4.1.2 TP-P-ABORT during the first phase of commitment

The scenario of figure C.44 describes a sequence of primitives in the case when a provider-supported transaction is aborted by a communications failure of the dialogue during the first phase of commitment.

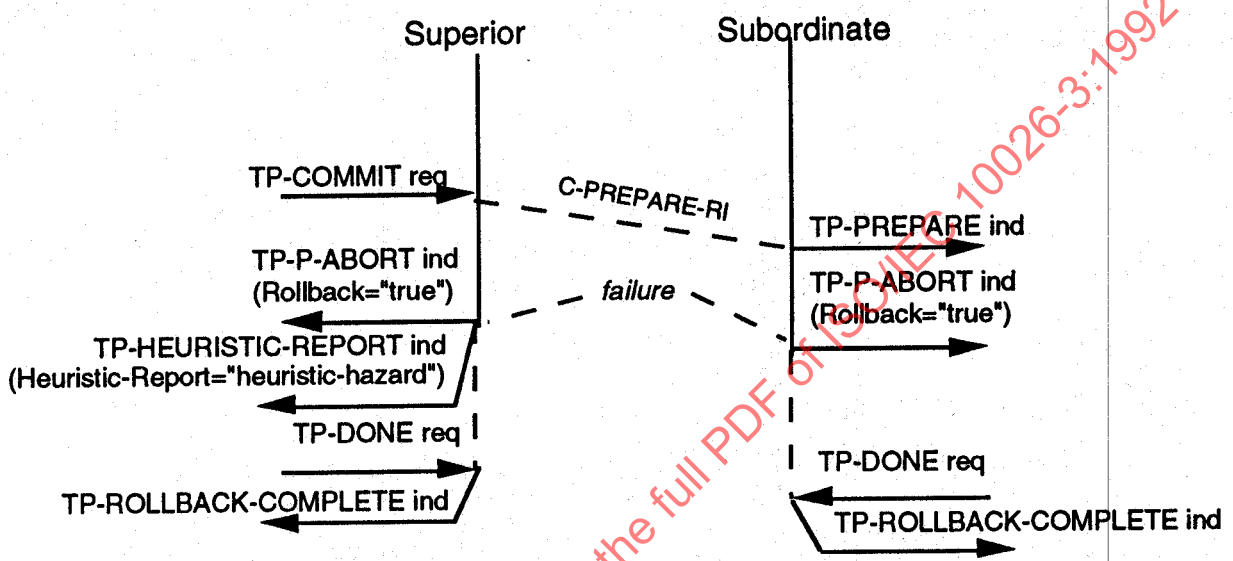
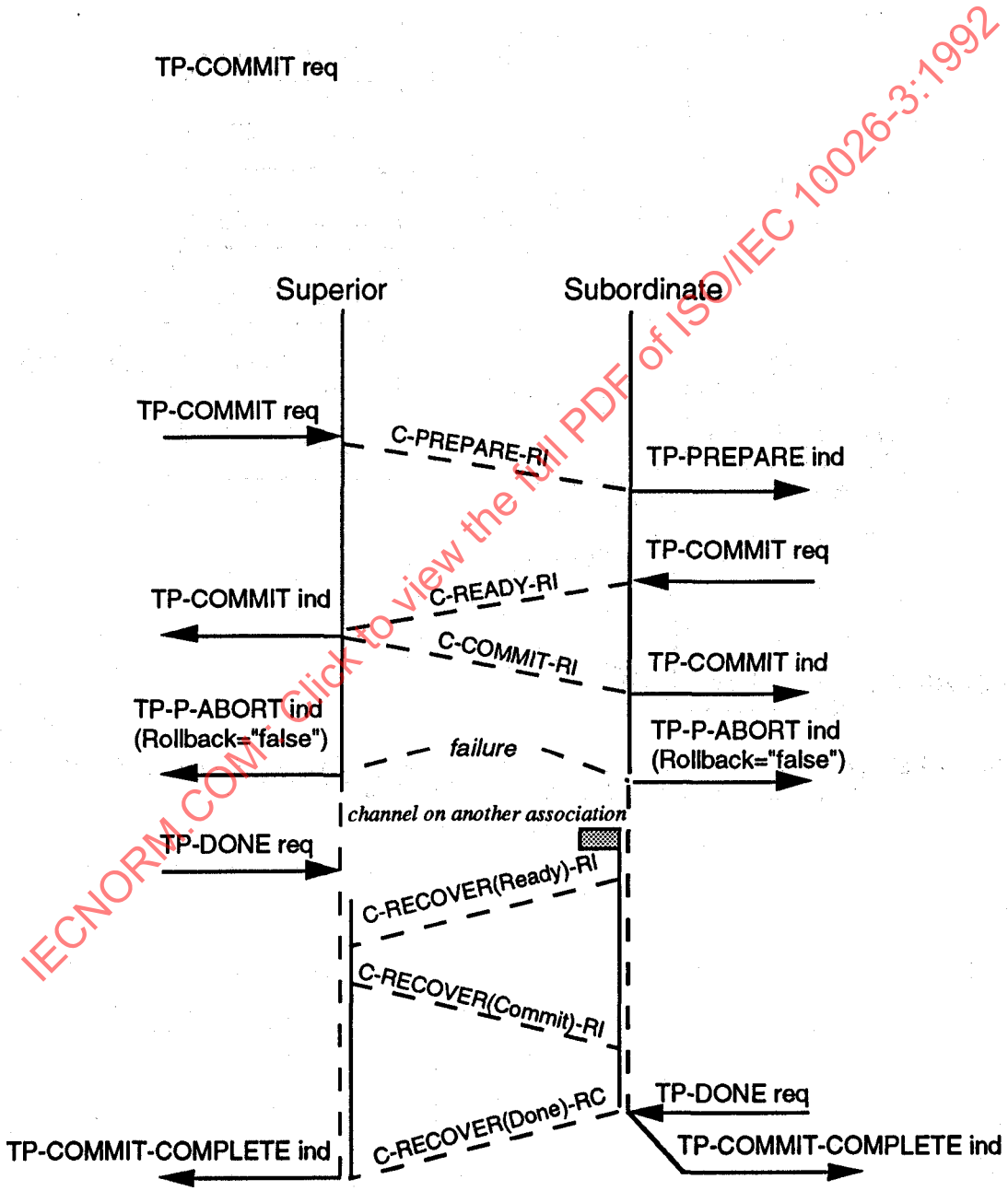


Figure C.44 — Failure for a provider-supported transaction during the first phase of commitment

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.4.1.3 TP-P-ABORT during the second phase of commitment

The scenarios of figure C.45, C.46 and C.47 describe sequences of primitives in the cases when a dialogue is aborted during the second phase of commitment. The three scenarios illustrate the effect that the precise timing of a failure has during commitment.



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure C.45 — Failure for a provider-supported transaction during the second phase of commitment

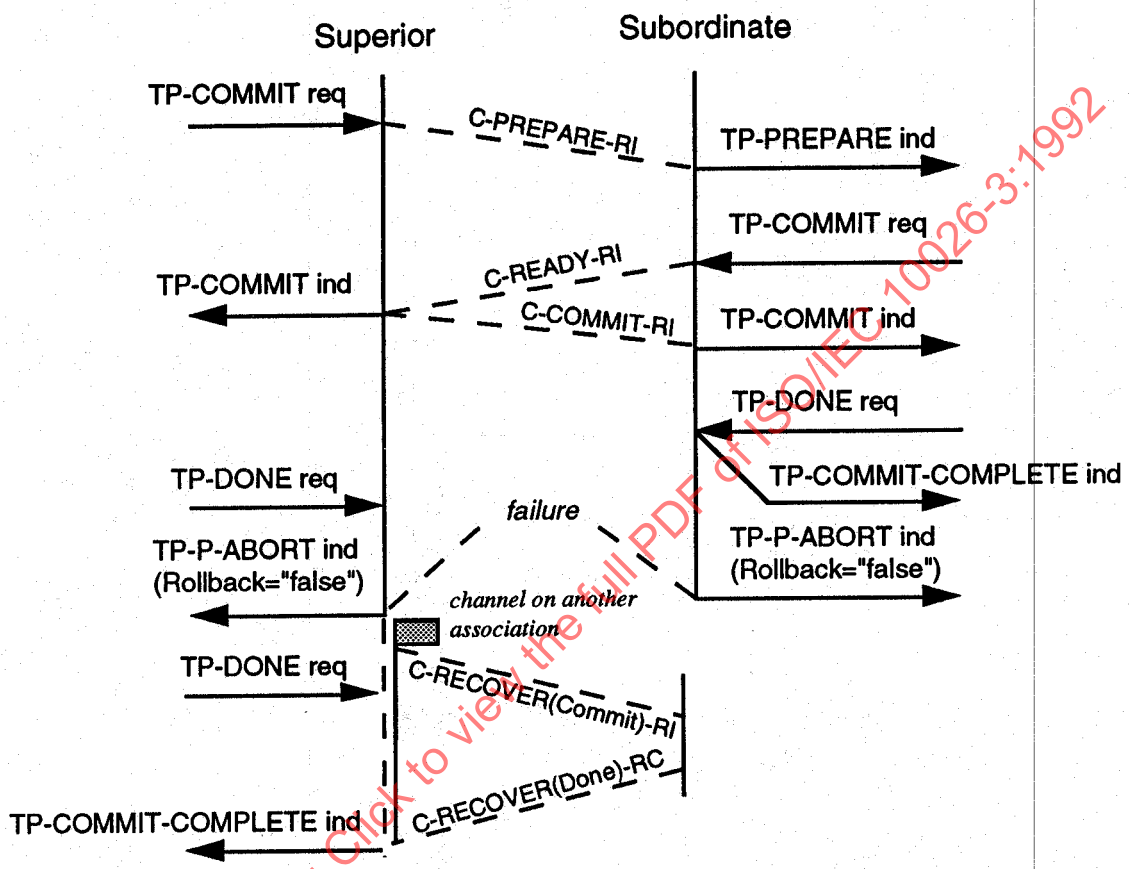


Figure C.46 — Failure for a provider-supported transaction during the second phase of commitment when the Unchained Transactions functional unit is selected

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

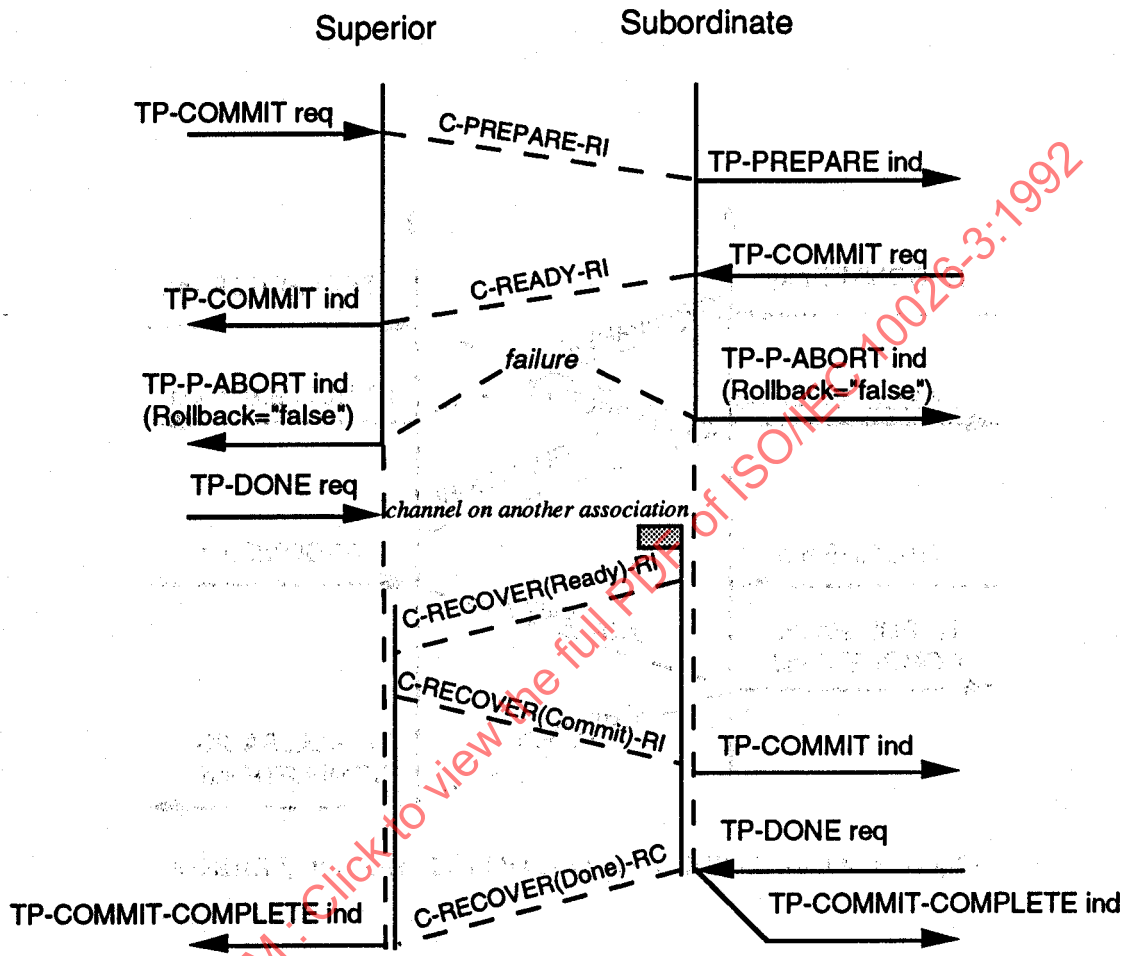


Figure C.47 — Failure for a provider-supported transaction during the second phase of commitment

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5 Collision scenarios on a single dialogue

C.5.1 Collisions of TP-U-ERROR with TP-COMMIT

The scenario of figure C.48 shows that, if a TP-U-ERROR and a TP-COMMIT collide, the TP-U-ERROR indication is suppressed and a rollback procedure is triggered.

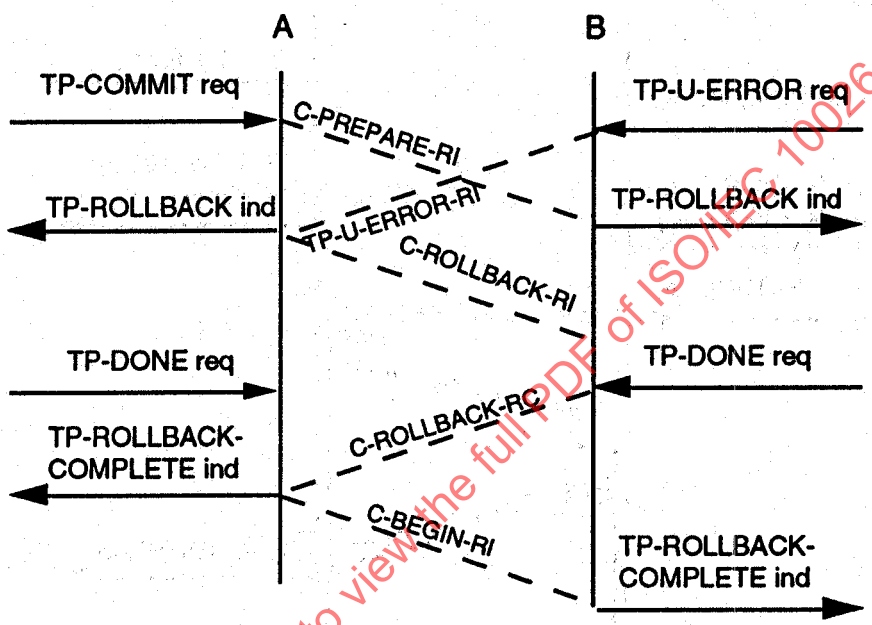


Figure C.48 — Collision of TP-COMMIT and TP-U-ERROR

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.2 Collisions with TP-U-ERROR in Polarized Control

After a TP-U-ERROR request issued when the TPSUI does not have the control of the dialogue, any TP-U-ERROR indications and TP-DATA indication are suppressed according to the scenario of figure C.49.

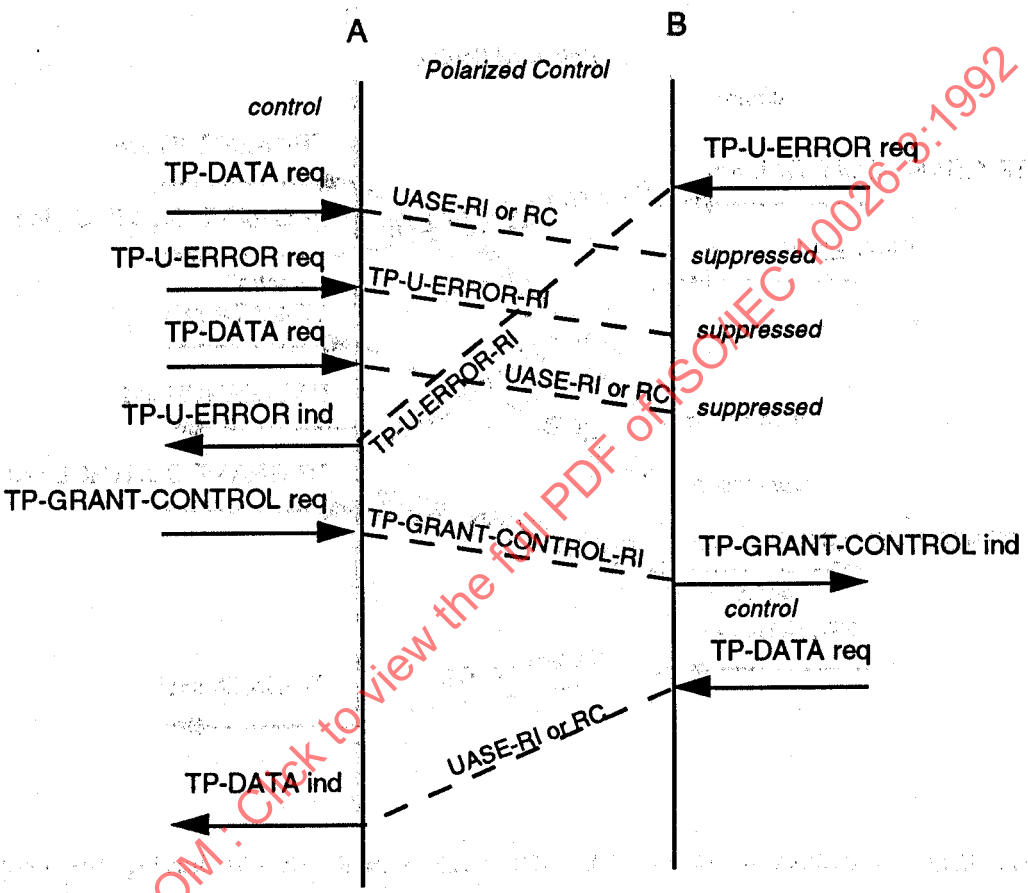


Figure C.49 — Suppression effects of TP-U-ERROR

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Due to the transit delay of the Grant Control service, it may happen that two TP-U-ERROR requests sent by TPSUIs not having control collide, this collision is shown by the scenario of figure C.50.

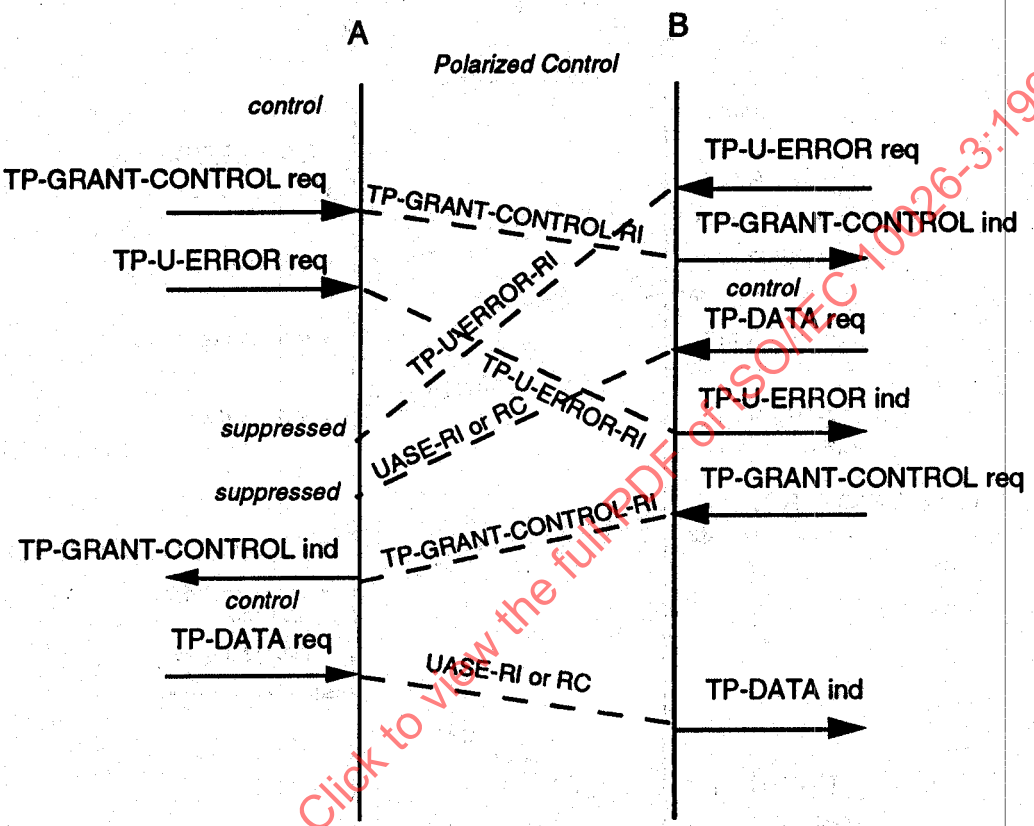


Figure C.50 — Collisions of two TP-U-ERRORs issued without having the control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.3 Collisions with TP-U-ERROR in Shared Control

The scenario of figure C.51 shows the suppression effect of TP-U-ERROR in Shared Control.

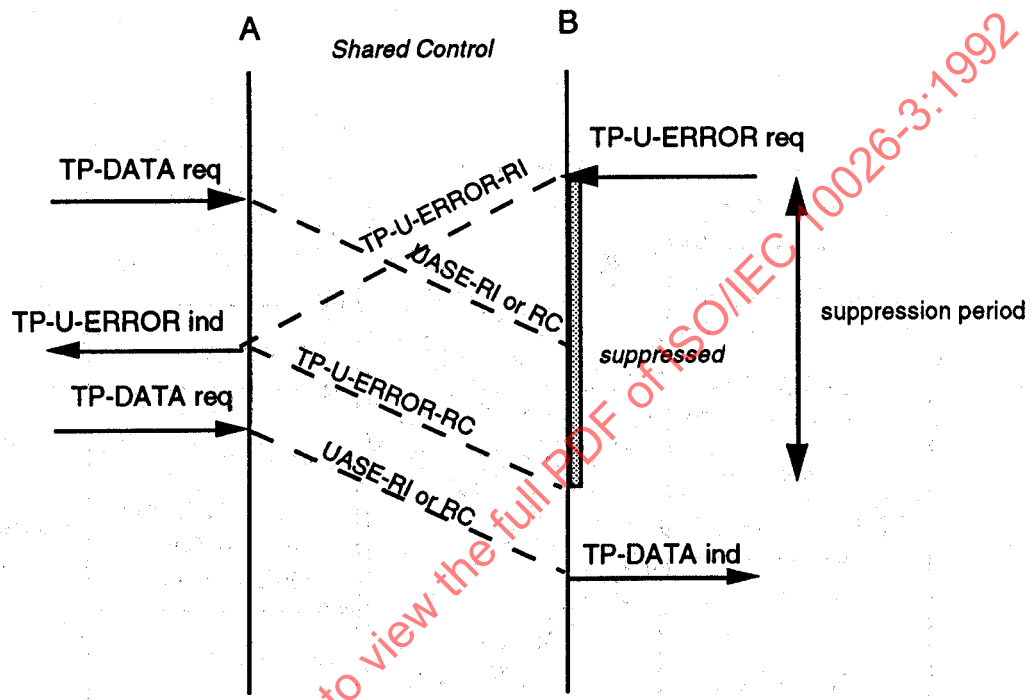


Figure C.51 — Suppression effect of TP-U-ERROR in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

If a collision occurs between two TP-U-ERROR request, the suppression is shown in figure C.52.

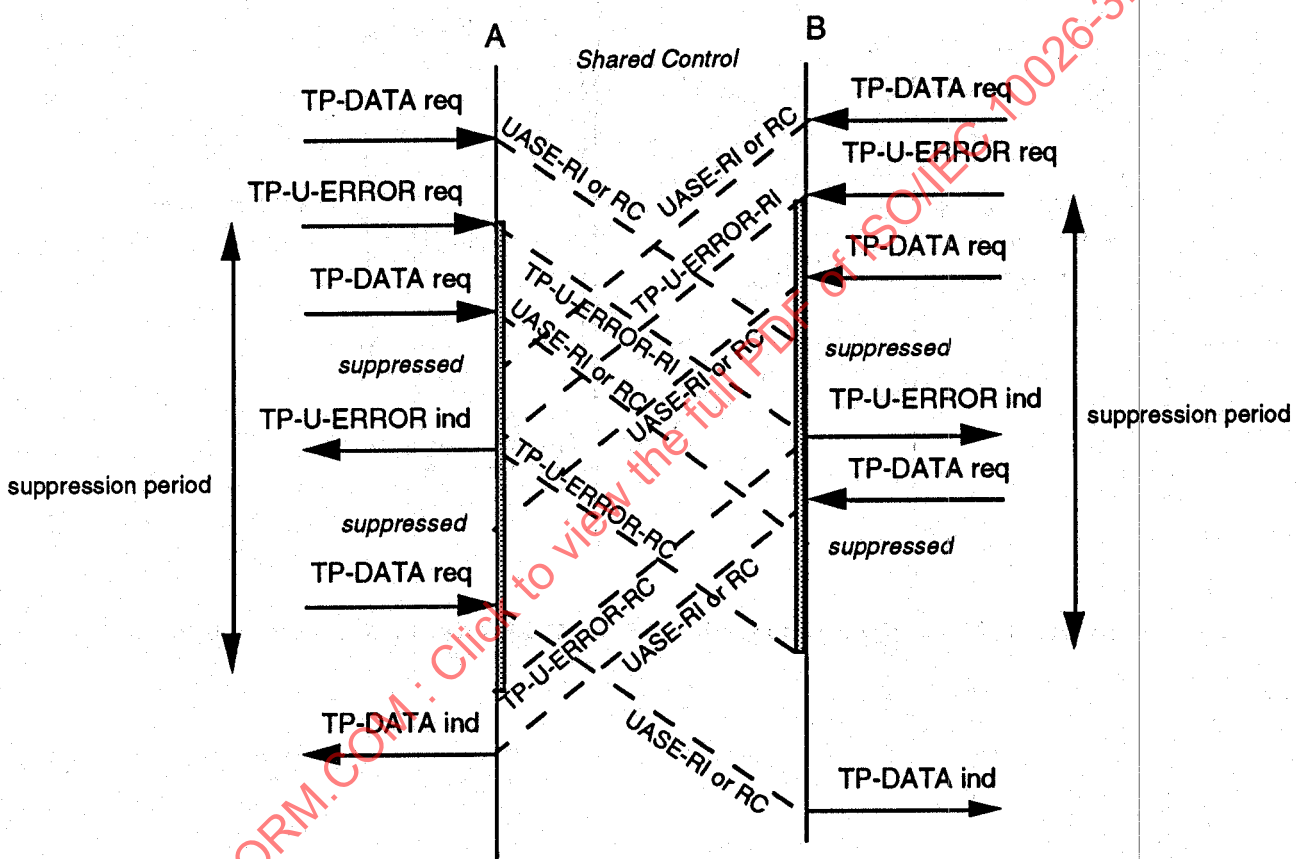


Figure C.52 — Collision of TP-U-ERRORs in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenarios of figure C.53 and C.54 shows that if TP-U-ERROR request is issued between a TP-HANDSHAKE request and the corresponding confirmation, the requestor of TP-U-ERROR request cannot issue a TP-DATA request until it has received the handshake confirmation.

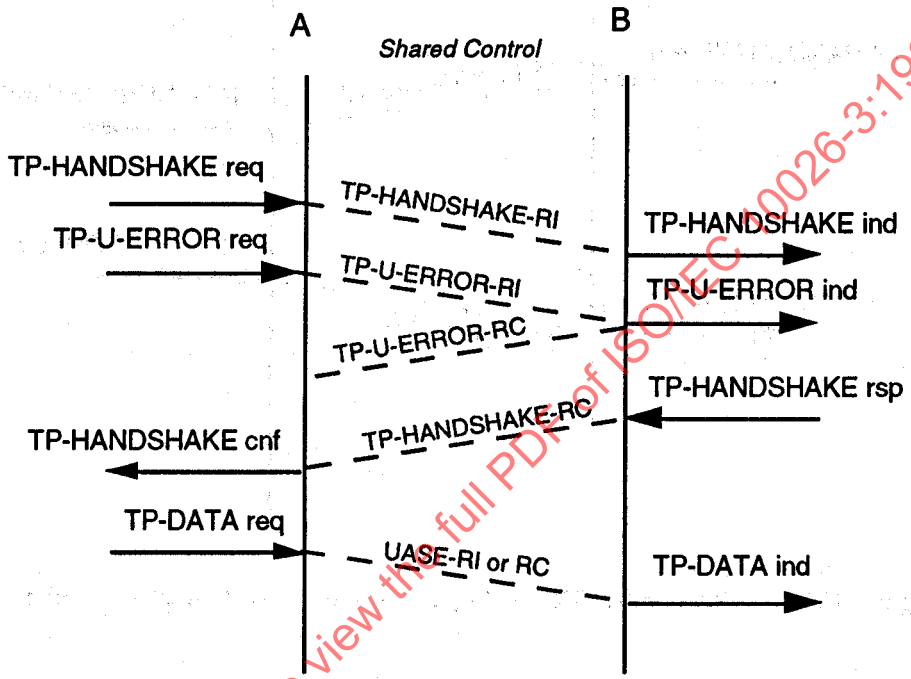


Figure C.53 — TP-U-ERROR during an outstanding handshake request

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

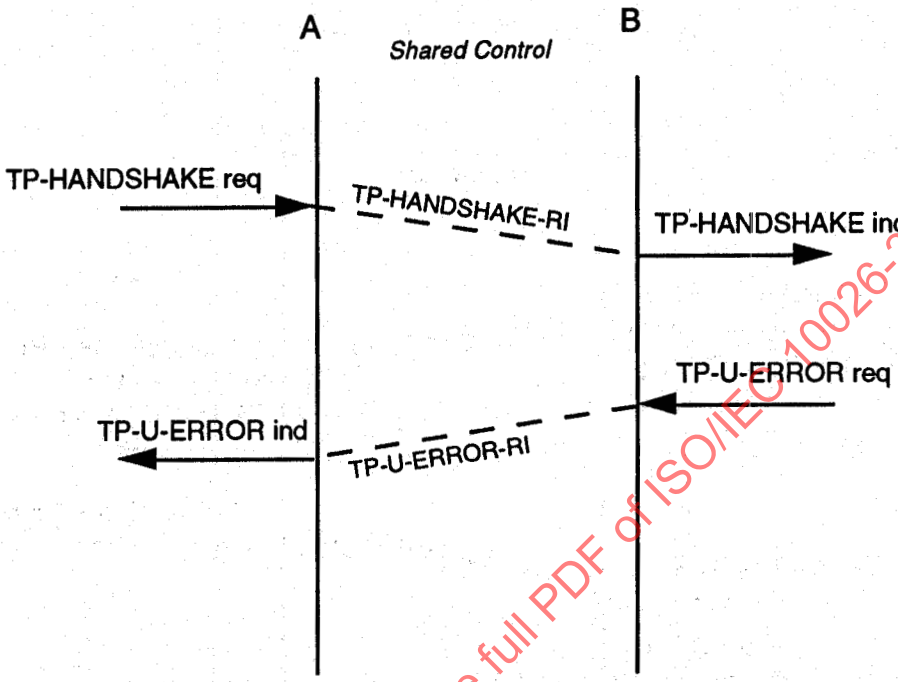


Figure C.54 — TP-U-ERROR during an outstanding handshake indication

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario in figure C.55 shows that, in Shared Control, a TP-U-ERROR request is not confirmed when colliding with a TP-END-DIALOGUE request (Confirmation="true"). It would be the same for the TP-U-ERROR request / TP-HANDSHAKE request collision.

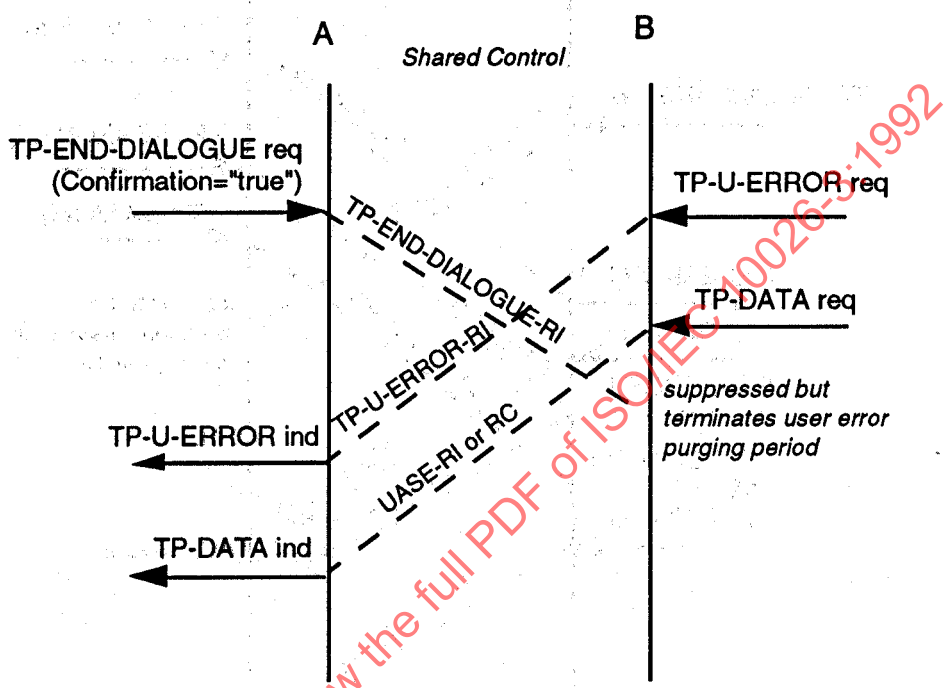


Figure C.55 — Collision of a TP-U-ERROR request with a TP-END-DIALOGUE indication

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario in figure C.56 shows a case when, in Shared Control, two TP-U-ERROR requests collide with a TP-HANDSHAKE request; only the first TP-U-ERROR-RI is not confirmed.

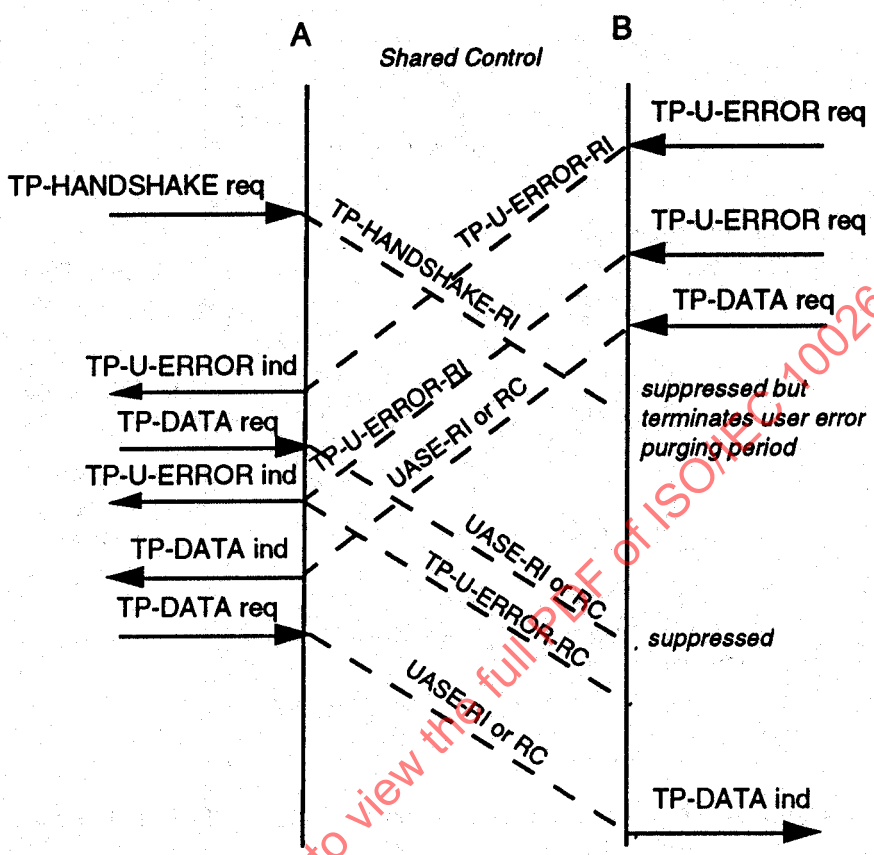


Figure C.56 — Collision of two TP-U-ERROR requests with a TP-HANDSHAKE indication in Shared control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario in figure C.57 shows that TP-U-ERROR-RC are not sent when a dialogue establishment indication is outstanding; they are sent after the TP-BEGIN-DIALOGUE-RC, and after a C-BEGIN-RC (if any).

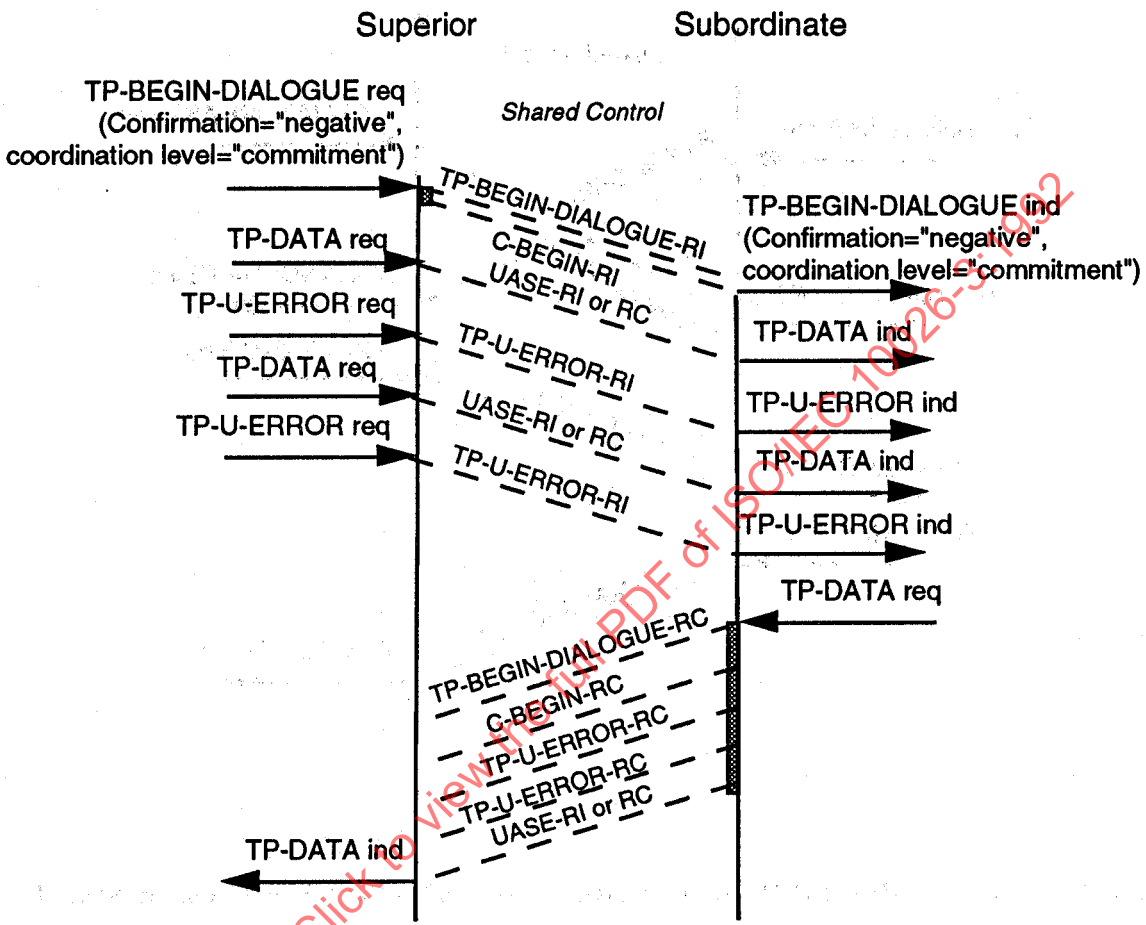


Figure C.57 — Queuing TP-U-ERROR-RC before TP-BEGIN-DIALOGUE-RC is sent in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario in figure C.58 shows that two TP-U-ERROR requests, sent in response to two colliding TP-HANDSHAKE request, are not confirmed.

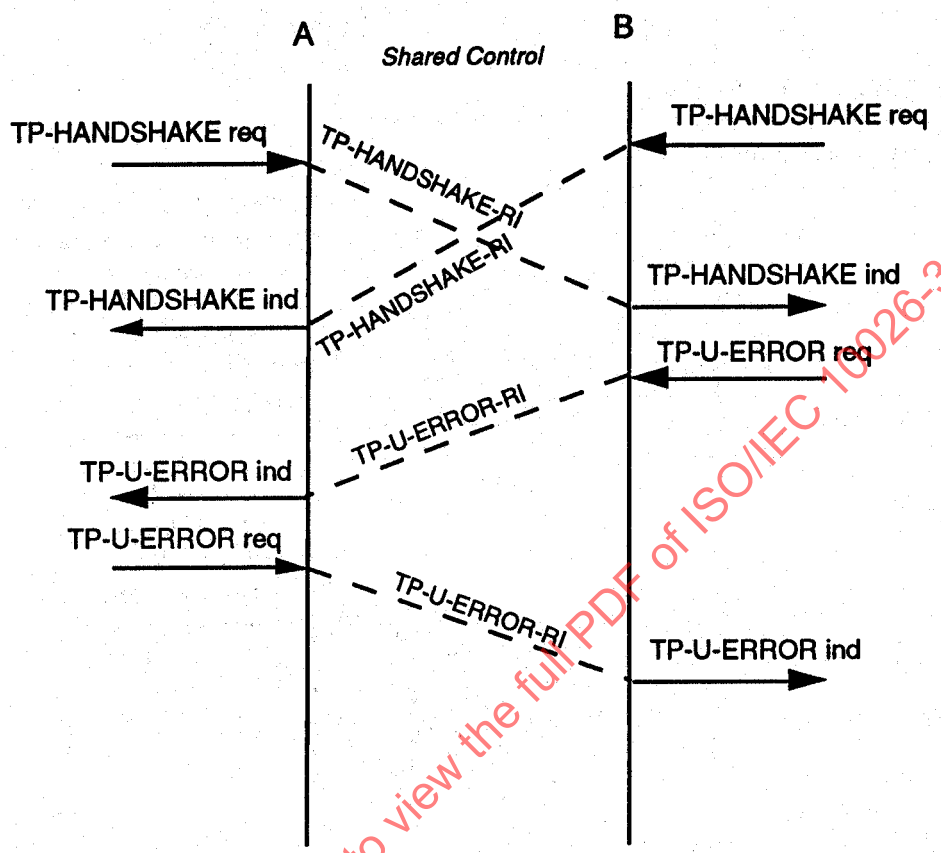


Figure C.58 — Two TP-U-ERROR requests after a TP-HANDSHAKE collision in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.4 TP-END-DIALOGUE Collision Scenarios

C.5.4.1 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-END-DIALOGUE (Confirmation="false")

The scenario of figure C.59 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request collides with a TP-END-DIALOGUE (Confirmation="false") request in Shared Control.

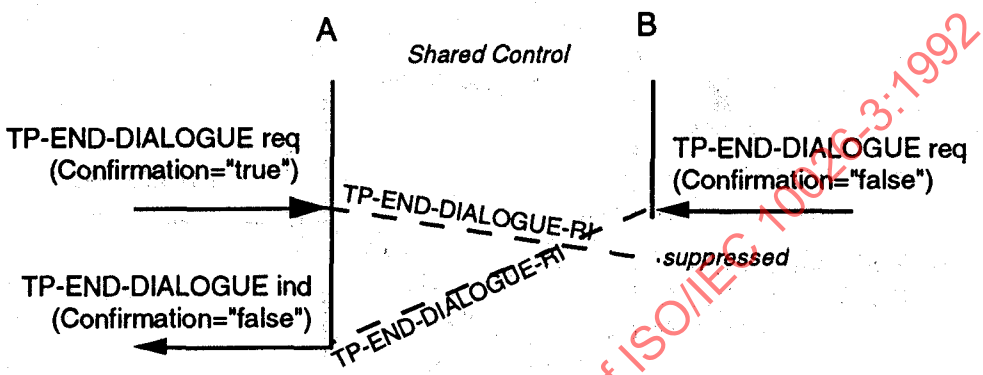


Figure C.59 — Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-END-DIALOGUE (Confirmation="false")

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.4.2 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-END-DIALOGUE (Confirmation="true")

The scenario of figure C.60 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request collides with another TP-END-DIALOGUE (Confirmation="true") request in Shared Control.

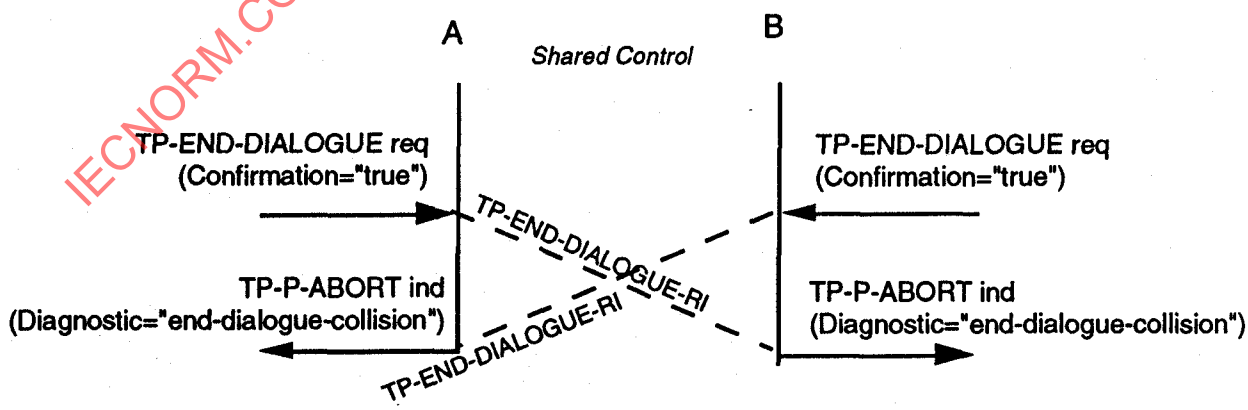


Figure C.60 — Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-END-DIALOGUE (Confirmation="true")

C.5.4.3 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-U-ERROR

The scenario of figure C.61 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request collides with a TP-U-ERROR request in Polarized Control.

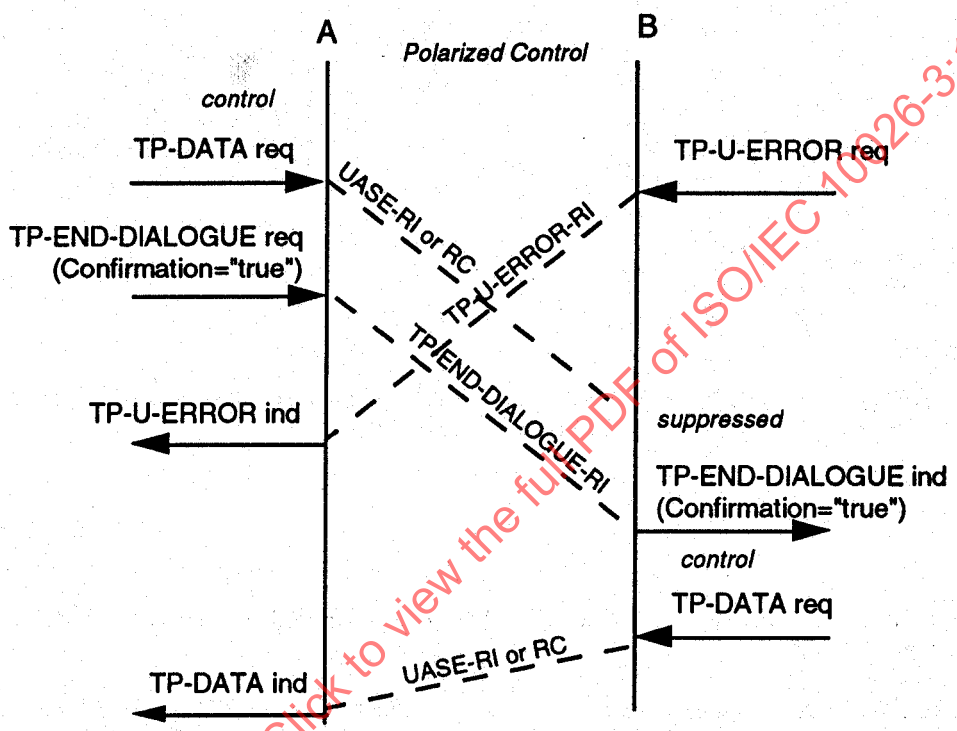


Figure C.61 — Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-U-ERROR in Polarized Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

The scenario of figure C.62 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request is involved in a collision with a TP-U-ERROR request in Shared Control. The TP-END-DIALOGUE indication is suppressed.

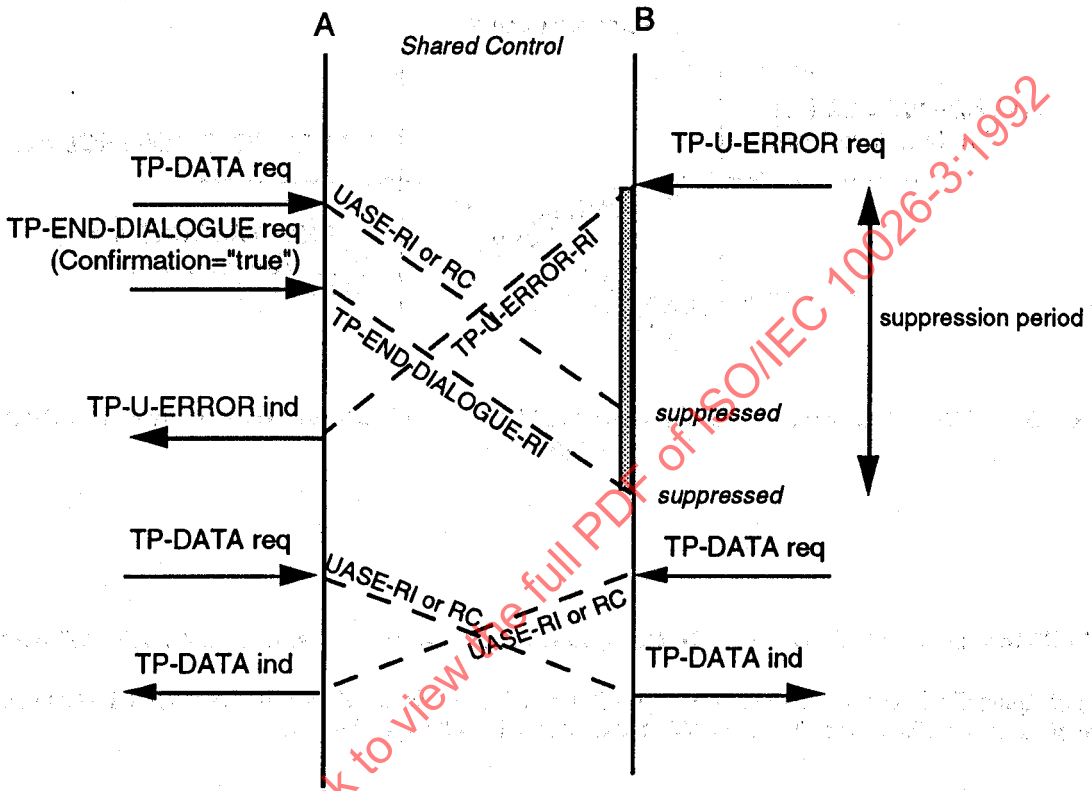


Figure C.62 — Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-U-ERROR in Shared Control

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.4.4 Collision of a TP-END-DIALOGUE (Confirmation="true") with a TP-REQUEST-CONTROL

The scenario of figure C.63 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request is involved in a collision with a TP-REQUEST-CONTROL request in Polarized Control.

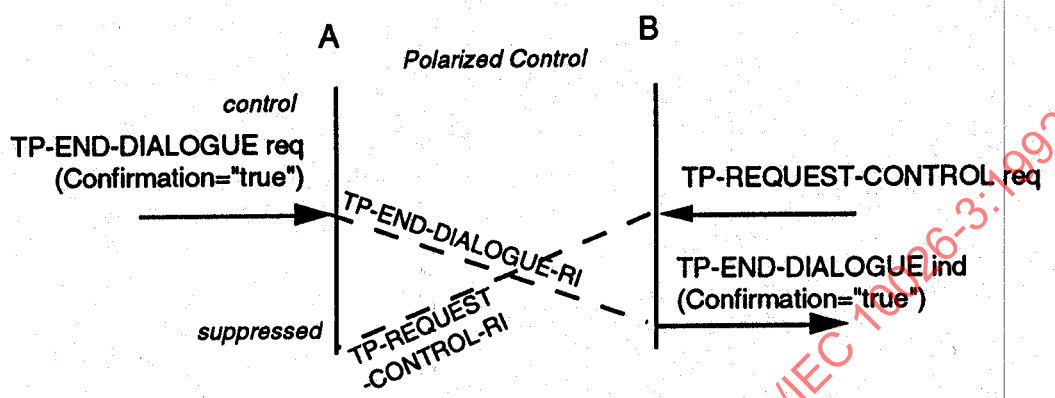


Figure C.63 — Collision of a TP-END-DIALOGUE (Confirmation="true") with a TP-REQUEST-CONTROL

C.5.4.5 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-BEGIN-TRANSACTION

The scenario of figure C.64 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request is involved in a collision with TP-BEGIN-TRANSACTION in Shared Control.

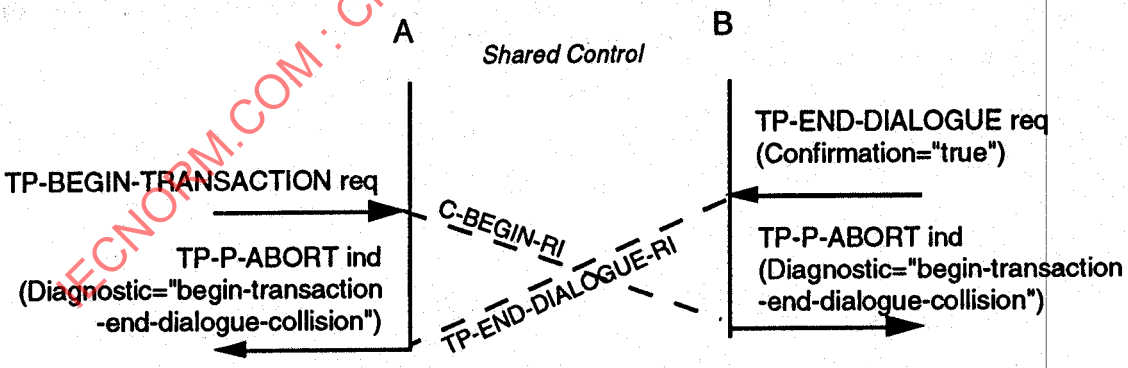


Figure C.64 — Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-BEGIN-TRANSACTION

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.4.6 Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-HANDSHAKE

The scenario of figure C.65 describes a sequence of primitives in the case when a TP-END-DIALOGUE (Confirmation="true") request is involved in a collision with TP-HANDSHAKE in Shared Control. TPSUI B shall wait for the TP-HANDSHAKE confirm before issuing the TP-END-DIALOGUE response.

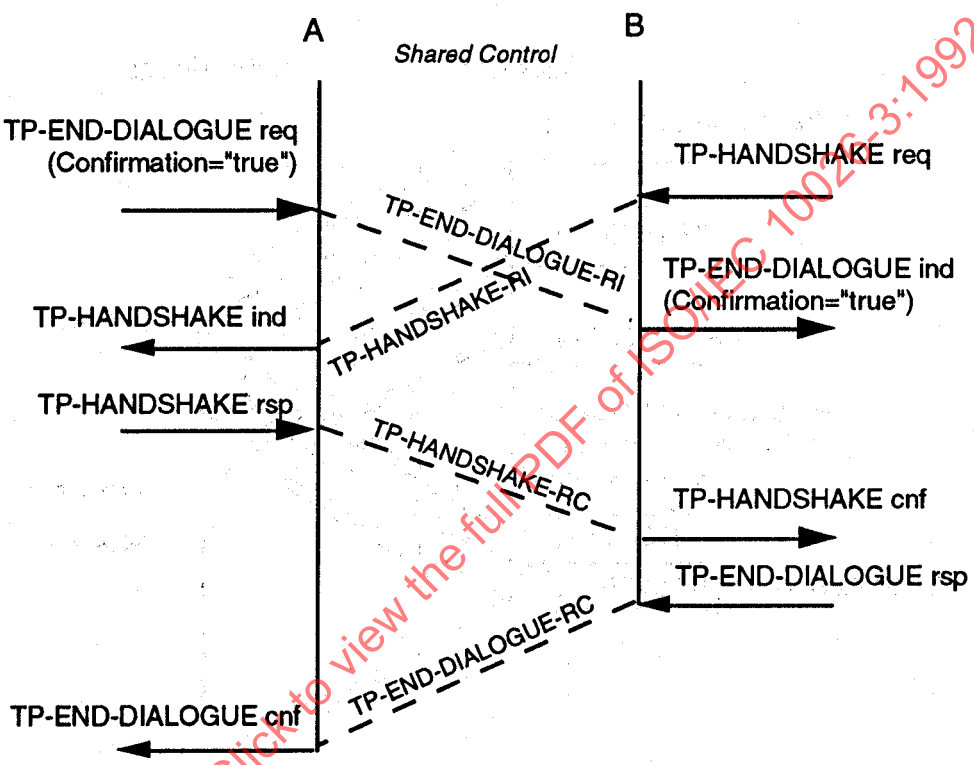


Figure C.65 — Collision of a TP-END-DIALOGUE (Confirmation="true") and a TP-HANDSHAKE

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.5 Other collisions

C.5.5.1 Collision of a TP-COMMIT and a dialogue establishment rejects.

The scenarios of figures C.66 and C.67 show that if TP-COMMIT request is issued before a TP-BEGIN-DIALOGUE confirm (Result="rejected"), the transaction is rolled back.

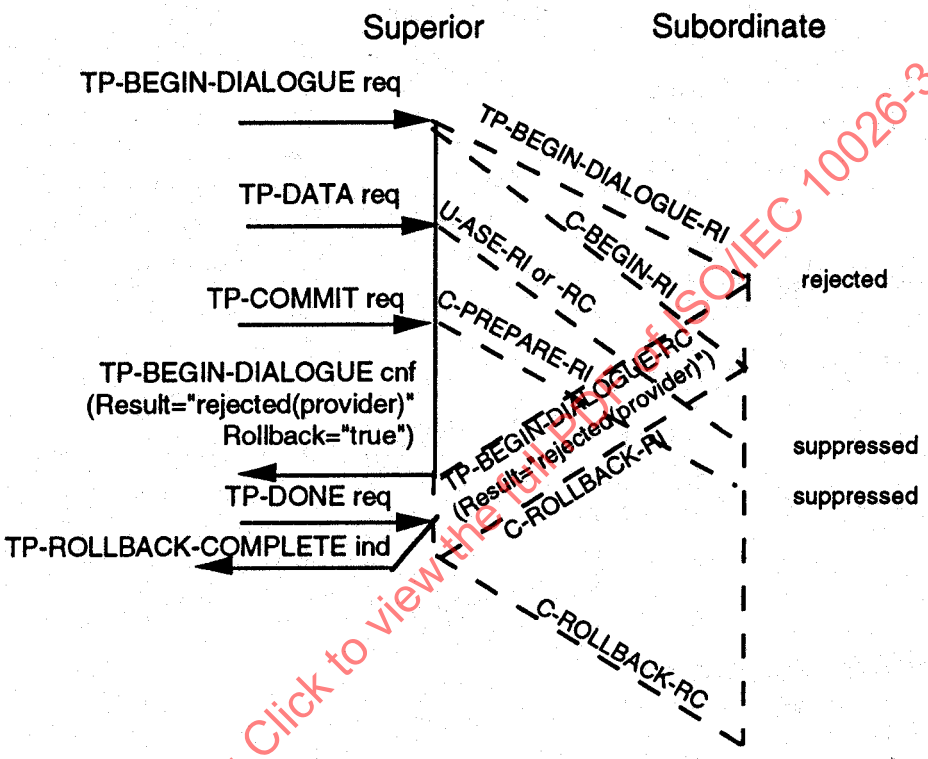
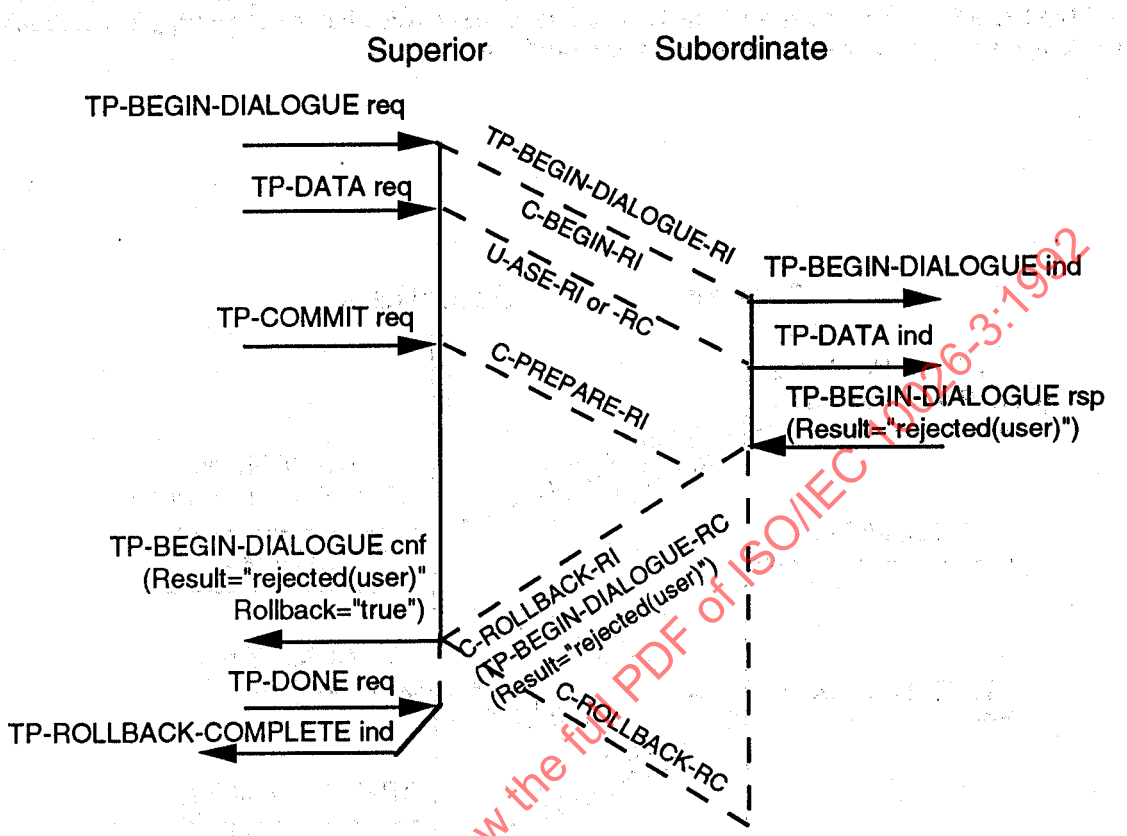


Figure C.66 — Collision of a TP-COMMIT request and a TP-BEGIN-DIALOGUE confirm (rejected(provider))

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



NOTE — the C-PREPARE-RI could result in a TP-PREPARE indication at the subordinate before the TP-BEGIN-DIALOGUE response; the final result would be exactly the same.

Figure C.67 — Collision of a TP-COMMIT request and a TP-BEGIN-DIALOGUE confirm (rejected(user))

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.5.5.2 Collision of a TP-COMMIT and TP-DATA

The scenario of figure C.68 shows a case where a TP-DATA issued after the Data-Permitted parameter has been set to "true" collides with a TP-COMMIT request; the subordinate is not aware of the collision but the TPSP triggers a rollback because the superior is no longer willing to accept indications from its subordinate.

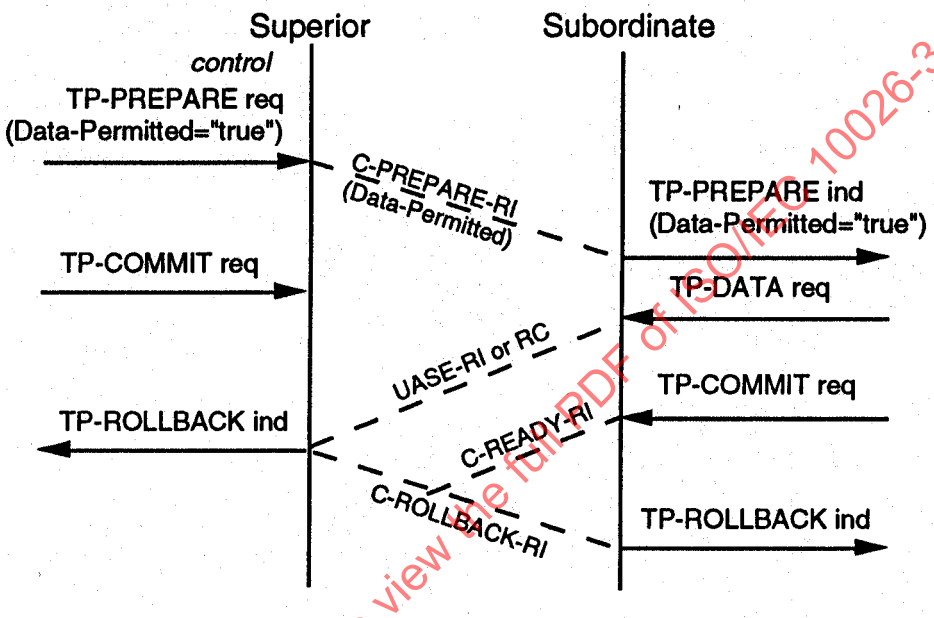


Figure C.68 — Collision of TP-COMMIT and TP-DATA

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.6 Tree with multiple dialogues (successful cases)

These scenarios show how the commitment procedures operate within a transaction tree containing several intermediate and leaf nodes.

In each case at the end of the sequences:

— for unchained transactions, the provider-supported transaction has ended and the coordination level is "none";

— for chained transactions, the new transaction will have begun and the transaction tree would retain the same shape as in the previous transaction.

C.6.1 The committing phase driven by TP-COMMIT

The scenario of figure C.69 shows the sequence of primitives following issuance of the TP-COMMIT request.

C.6.2 The committing phase with TP-PREPARE and TP-READY

The scenario of figure C.70 shows the sequence of primitives following invocation of the TP-PREPARE request by the root TPSUI for one of its subordinates (C); this illustrates how a TPSUI(TPSUI A) can initiate commitment on some branches(in this case, the single branch A-C) while remaining active, by using the TP-PREPARE request.

The scenario additionally shows the same function being used by node C on its branch C-D.

C.7 Tree with multiple dialogues (unsuccessful cases)

C.7.1 Rollback from the root during the active phase

The scenario of figure C.71 shows the sequence of primitives following invocation of TP-ROLLBACK request during the active phase of a transaction, for example as the result of a TPSUI error. This scenario covers the case where TP-ROLLBACK request is invoked in the root node of the transaction tree.

At the end of this scenario, all bound data will have been returned to their initial state. If the chained transactions functional unit has been selected on the dialogues, a new transaction will have begun.

C.7.2 Rollback from a subordinate during the active phase

The scenarios of figures C.72 and C.73 are similar to the scenario of figure C.71; however the rollback originates from an intermediate node. Figures C.72 and C.73 show the unchained and chained cases respectively.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

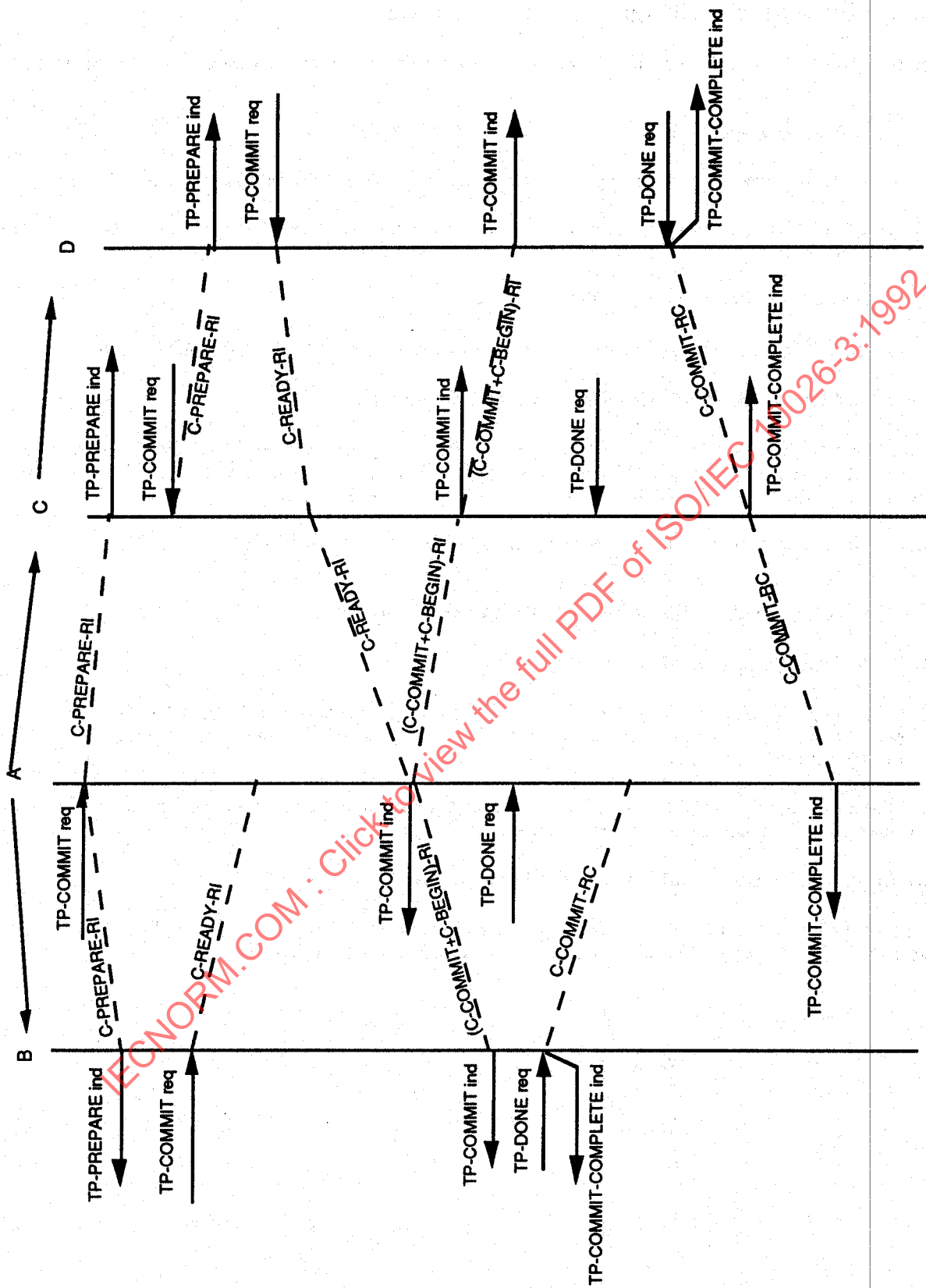


Figure C.69 — Commitment in a multi-dialogue tree (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

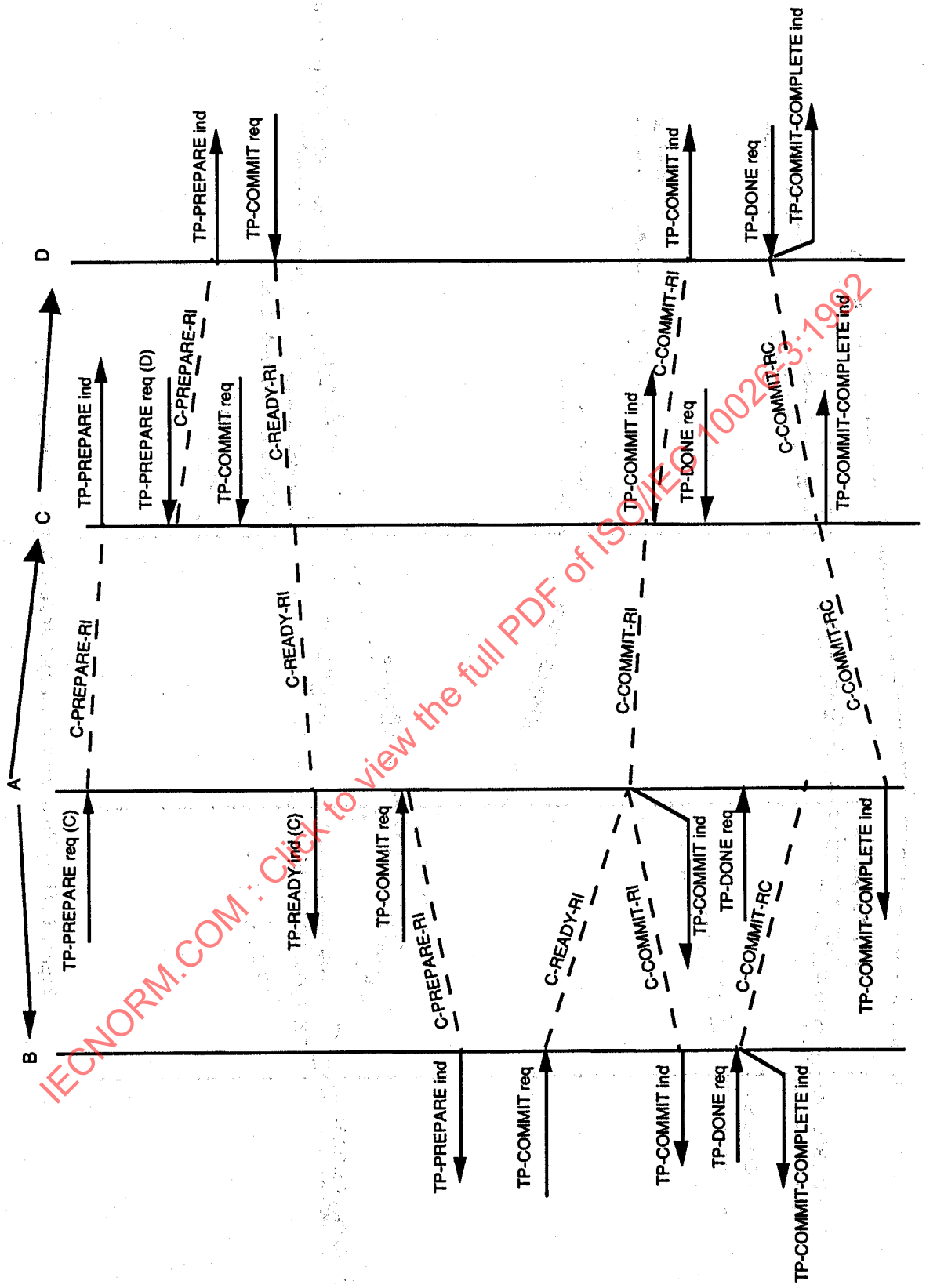


Figure C.70 — Commitment in a multi-dialogue tree with use of TP-PREPARE (Unchained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

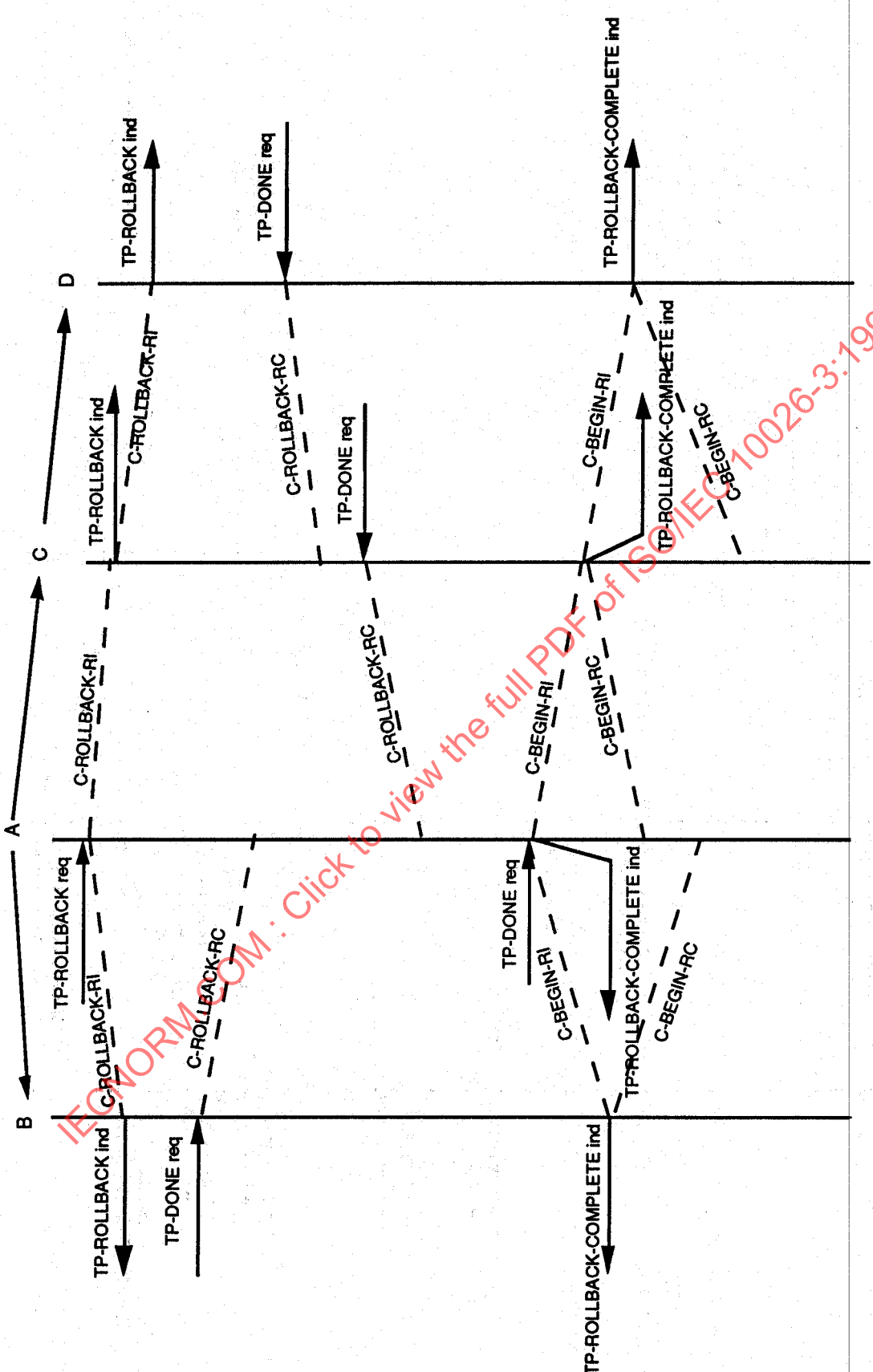


Figure C.71 — Rollback from a root node (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

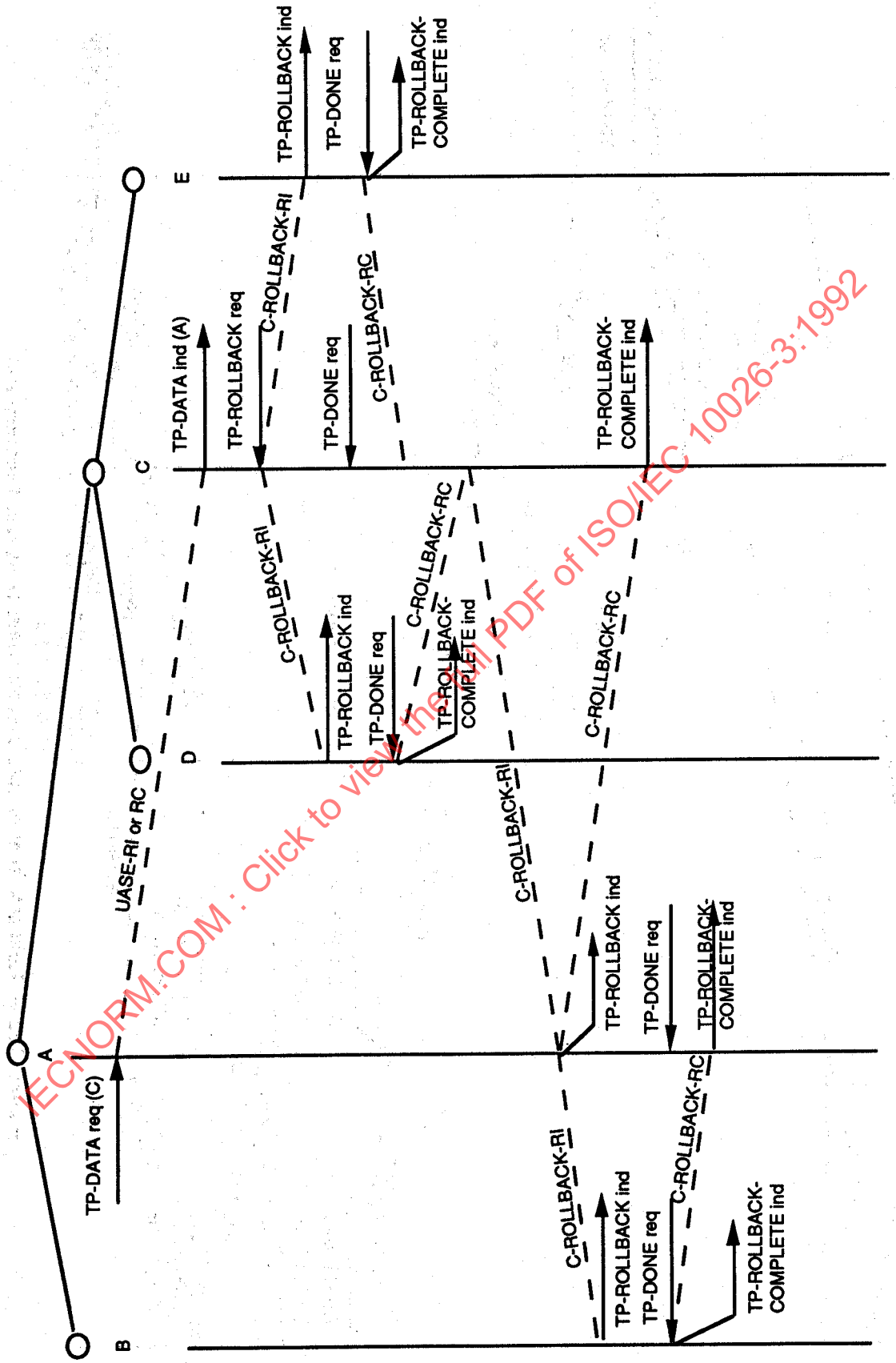


Figure C.72 — Rollback from an intermediate node (Unchained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

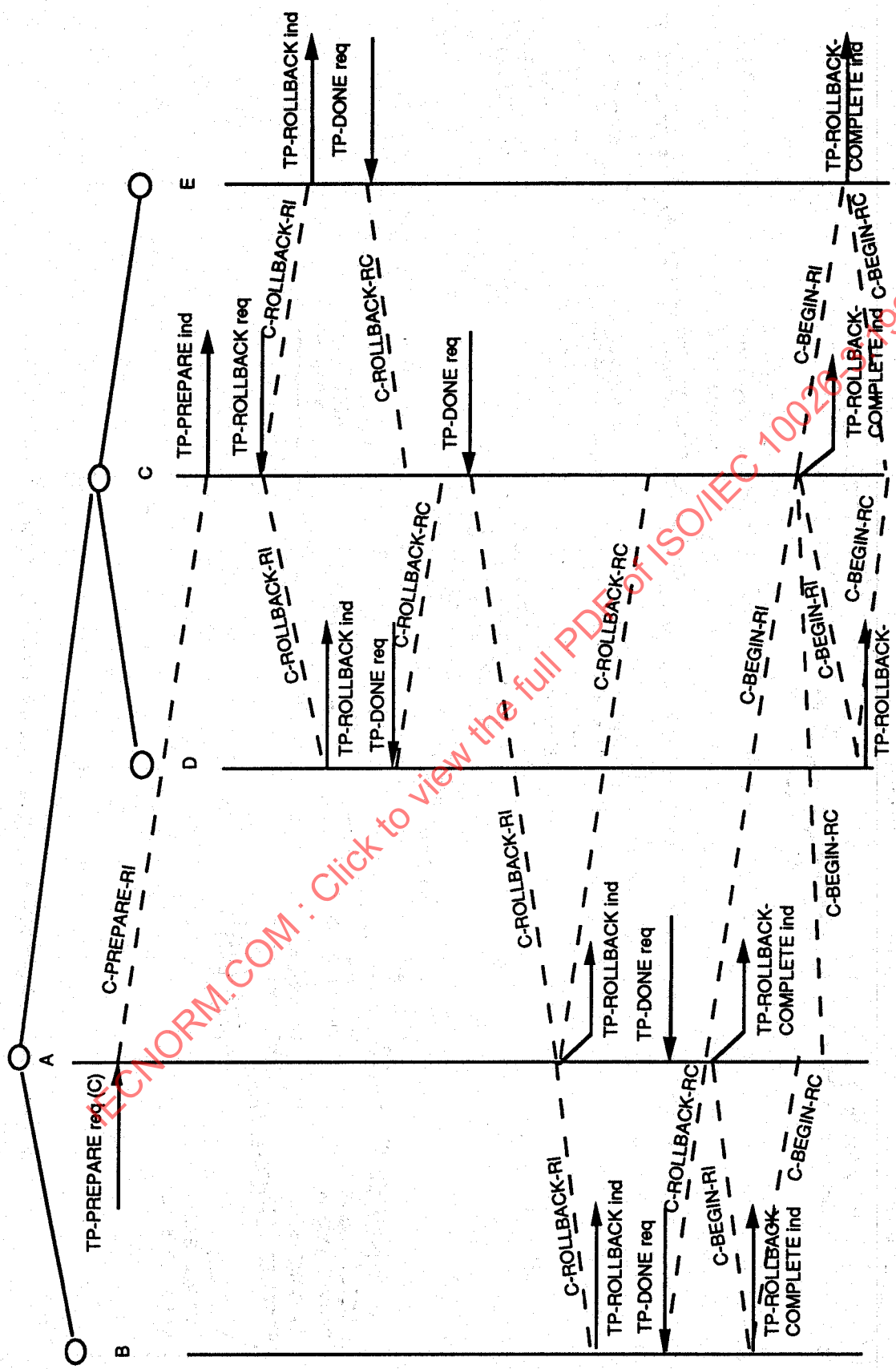


Figure C.73 — Rollback from an intermediate node (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.7.3 Dialogue abort during phase I of commitment

The scenario of figure C.74 is based on scenario of figure C.69 except that a communications failure (causing TP-P-ABORT indications) occurs between A and B, and C and D, during the first phase of commitment.

As the failure between C and D occurs before the commitment sequence begins, this is equivalent to a dialogue abort in the active phase and so no later recovery actions are required. Between B and A, transaction recovery is initiated from B and takes place on a new association.

At the end of the sequence, the dialogue tree contains only A and C; then, depending on the functional units selected:

— unchained transactions: the transaction has been rolled back and no new transaction has been started;

— chained transactions: the transaction has been rolled back and a new transaction tree established containing A and C.

Further action is determined by the TPSUIs themselves..

C.7.4 Rollback-related actions

The scenario of figure C.75 shows a rollback initiated from a leaf node. In reaction to the rollback two other leaf nodes abort their superior dialogue.

In turn, their superior does not accept the situation and aborts its superior dialogue.

At the end of the scenario the tree is limited to two nodes.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

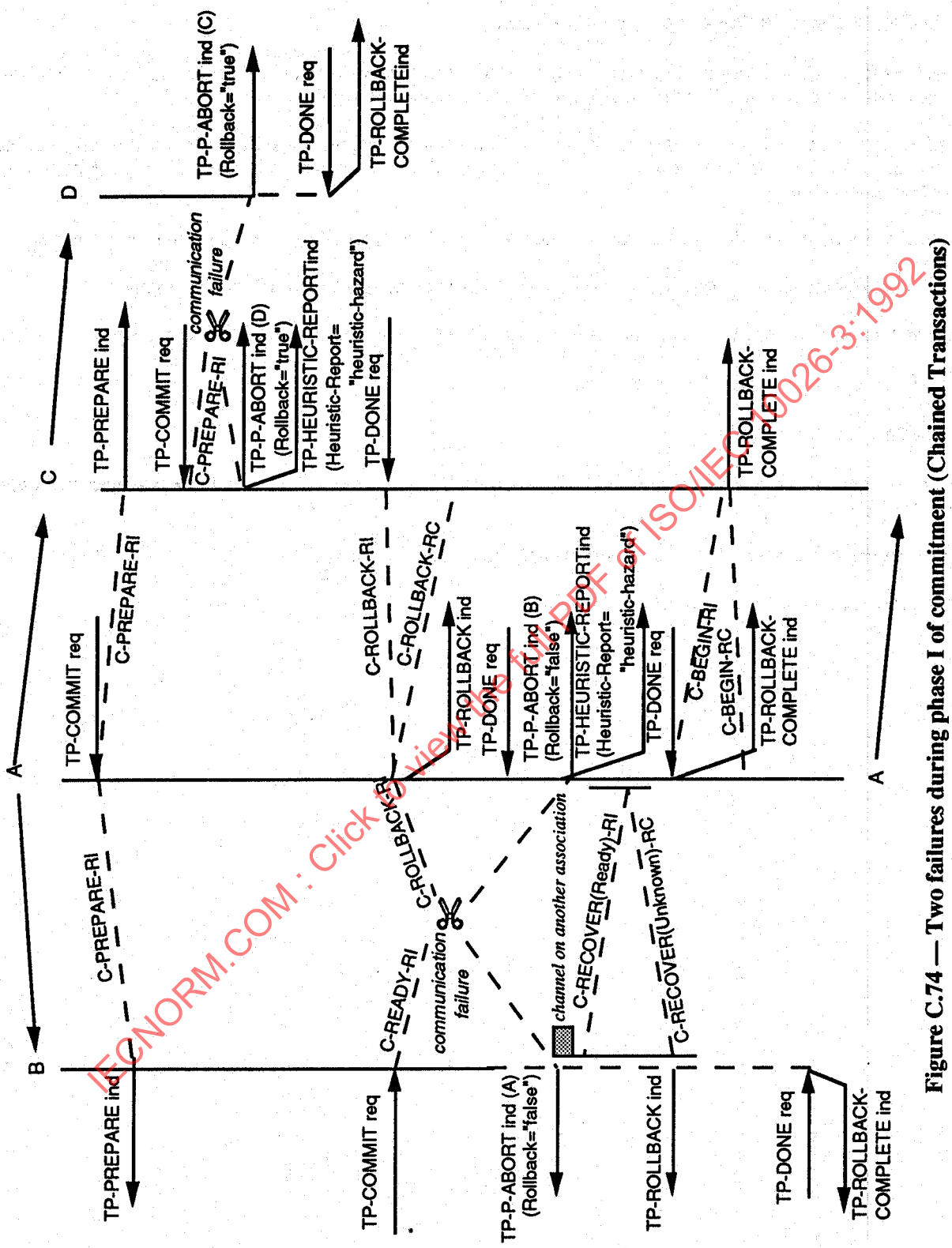


Figure C.74 — Two failures during phase I of commitment (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

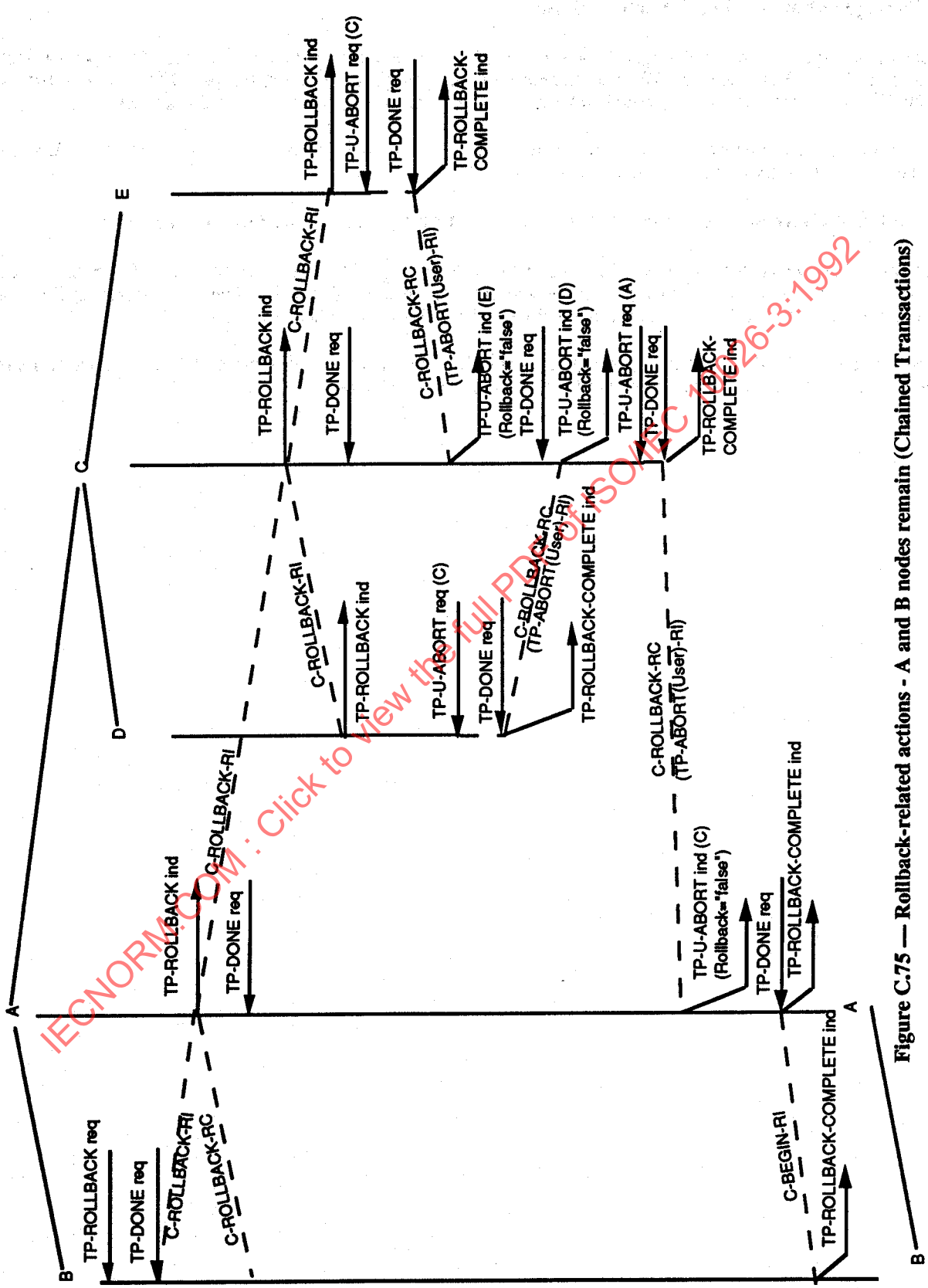


Figure C.75 — Rollback-related actions - A and B nodes remain (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.7.5 Dialogue abort during the active phase

In the scenario of figure C.76, the dialogue between C and E is aborted during the active phase of the provider-supported transaction in which A, B, C, D, E and E's subordinates are participating. In this case, this provider-supported transaction is rolled back (TP-P-ABORT indication is issued with the Rollback parameter set to "true" to both C and E).

At the end of the scenario, two isolated transaction trees exist : 1) a transaction tree with A as the root and which consists of A, B, C and D; 2) a transaction tree with E as the root.

C.7.6 Dialogue abort during the active phase, transaction tree above the failure disbanded

The beginning of the scenario of figure C.77 is similar to the beginning of the previous one. But after being aware of the failure with its superior, C aborts the dialogues it has with its superior (A) and its subordinate (D) by issuing TP-U-ABORT request on these dialogues. A, in return, aborts the dialogue with its last subordinate (B).

At the end of this scenario, the transaction tree superior to the failure is completely disbanded and A, B, C, and D are isolated. E has been isolated from C by the failure but is still the root of a transaction tree.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

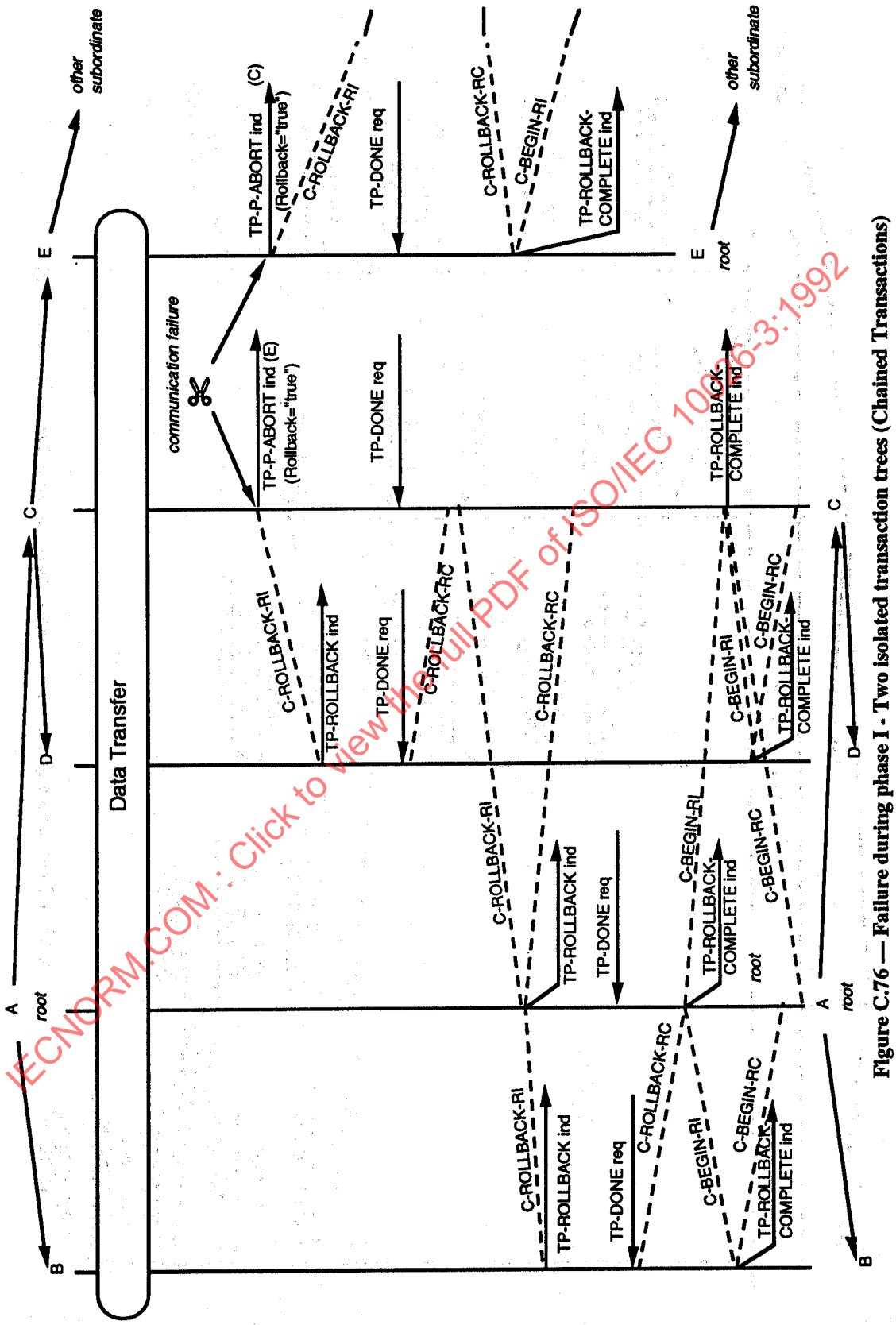


Figure C.76 — Failure during phase I - Two isolated transaction trees (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

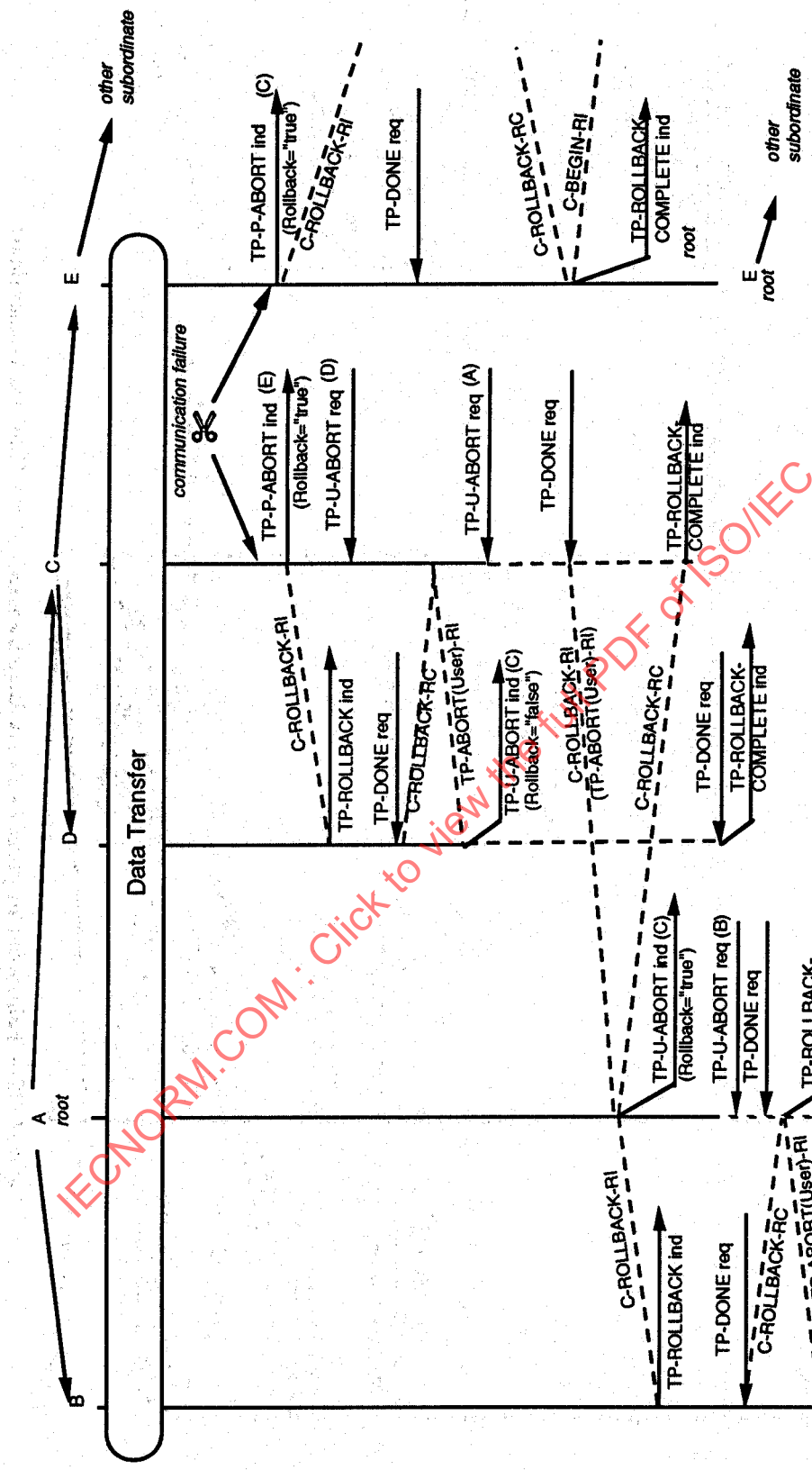


Figure C.77 — Failure during phase I - superior tree aborted (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.7.7 Dialogue abort during the second phase of commitment

In the scenario of figure C.78, the failure between C and E occurs after C has received TP-COMMIT indication, but before E has received TP-COMMIT indication. Therefore, the TP-P-ABORT indication issued to C and E is delivered with the Rollback parameter set to "false" (C knows that the terminating provider-supported transaction is committed, E is in doubt).

After recovery takes place, E receives TP-COMMIT indication and completes the commitment of its subtree.

Since the dialogue between C and E was in chained mode, the next transaction has to be rolled back from node C.

At the end of the scenario, two isolated transaction trees exist: 1) a transaction tree with A as the root and which consists of A, B, C and D; 2) a transaction tree with E as the root.

C.7.8 Dialogue abort during the second phase of commitment, subordinate of the failed dialogue aborts its other dialogues

In the scenario of figure C.79, the failure between A and C occurs after A has received TP-COMMIT indication, but before C has received TP-COMMIT indication. Therefore, the TP-P-ABORT indication issued to A and C is delivered with the Rollback parameter set to "false" (A knows that the terminating provider-supported transaction is committed, C is in doubt).

After having received TP-P-ABORT indication, C decides to abort the dialogue it has with its two subordinates (D and E) by issuing TP-U-ABORT request on this dialogues.

Since the dialogue between A and C was in chained mode, the next transaction has to be rolled back from node A.

At the end of the scenario, C, D and E are isolated, A and B form a transaction tree with A as the root.

C.7.9 Dialogue abort with the superior after TP-COMMIT indication has been issued

In the scenario of figure C.80, a failure occurs between A and its subordinate, C, after C has received TP-COMMIT indication, in this case the next provider-supported transaction has to be rolled back in the subtree which is isolated from the root of the transaction tree (C, D and E).

D and E, which has already received TP-COMMIT-COMPLETE indication and may have begun to perform some actions in the context of the next provider-supported transaction, receive TP-ROLLBACK indication and enter the rollback procedure. A enters the rollback procedure as well.

At the end of the scenario, C is the root of a transaction tree consisting of C, D and E. Another transaction tree consisting of A and B also exists.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

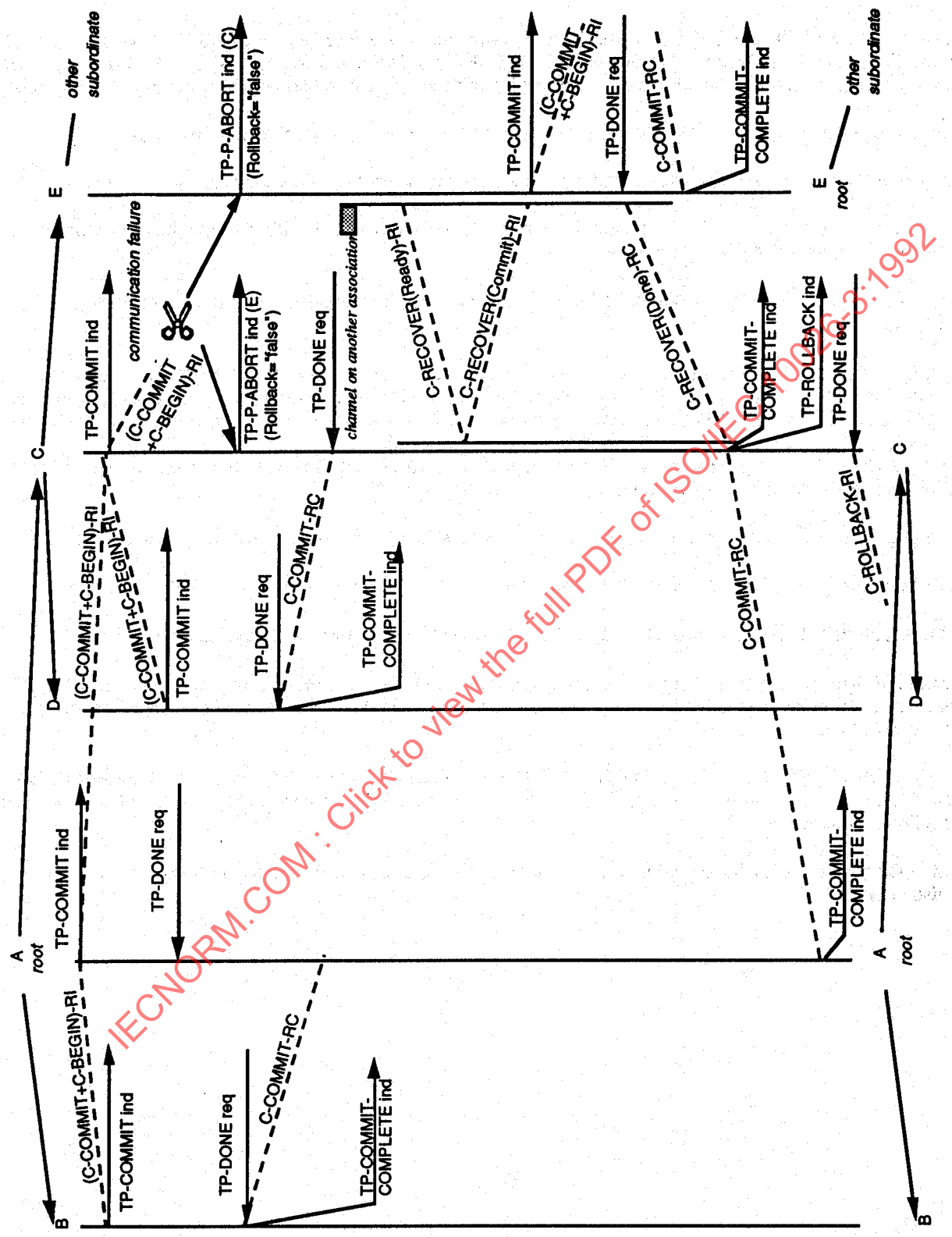


Figure C.78 — Failure during phase II - two isolated transaction trees (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

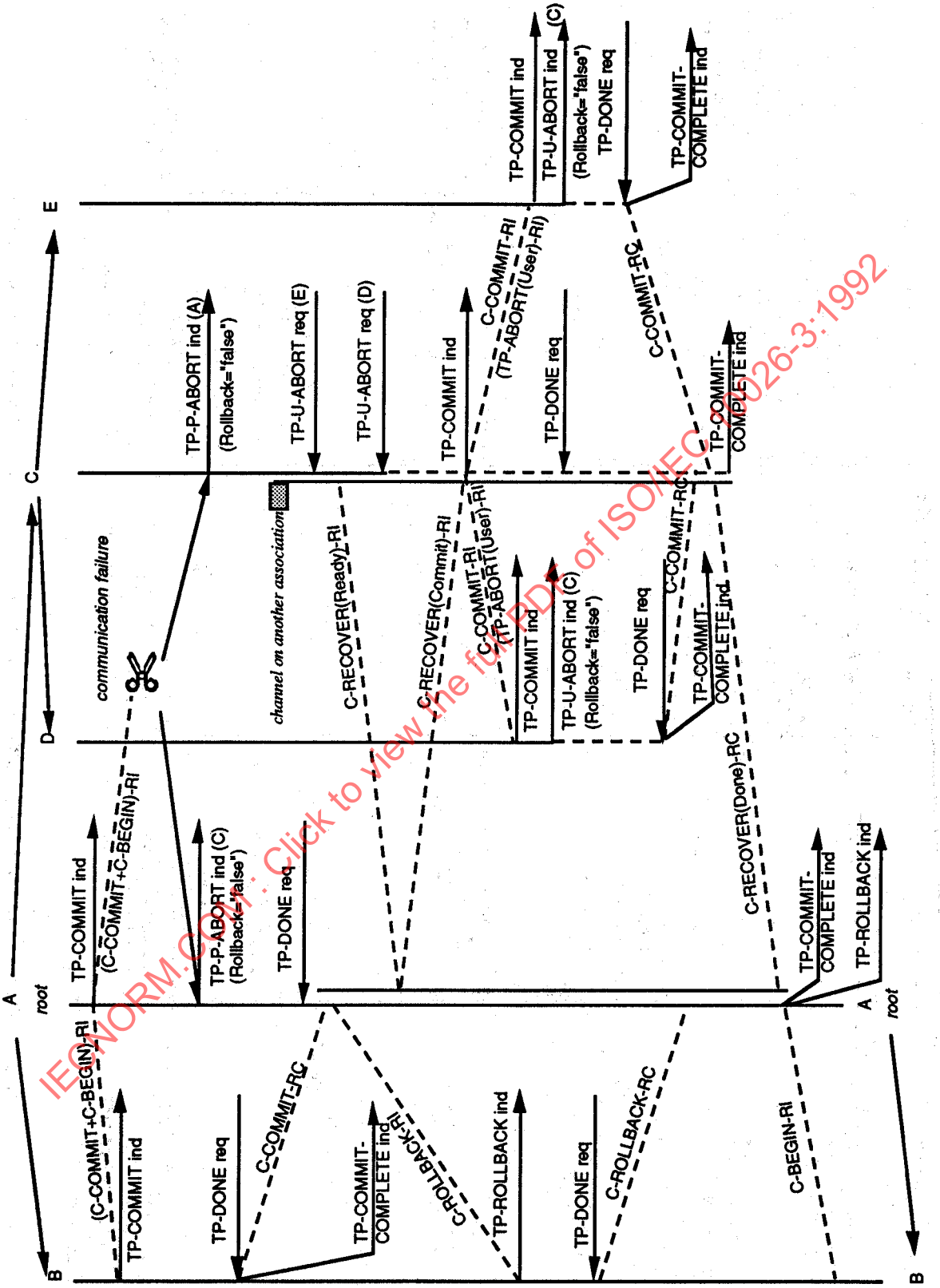


Figure C.79 — Failure during phase II - Subordinate refuses to become root (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

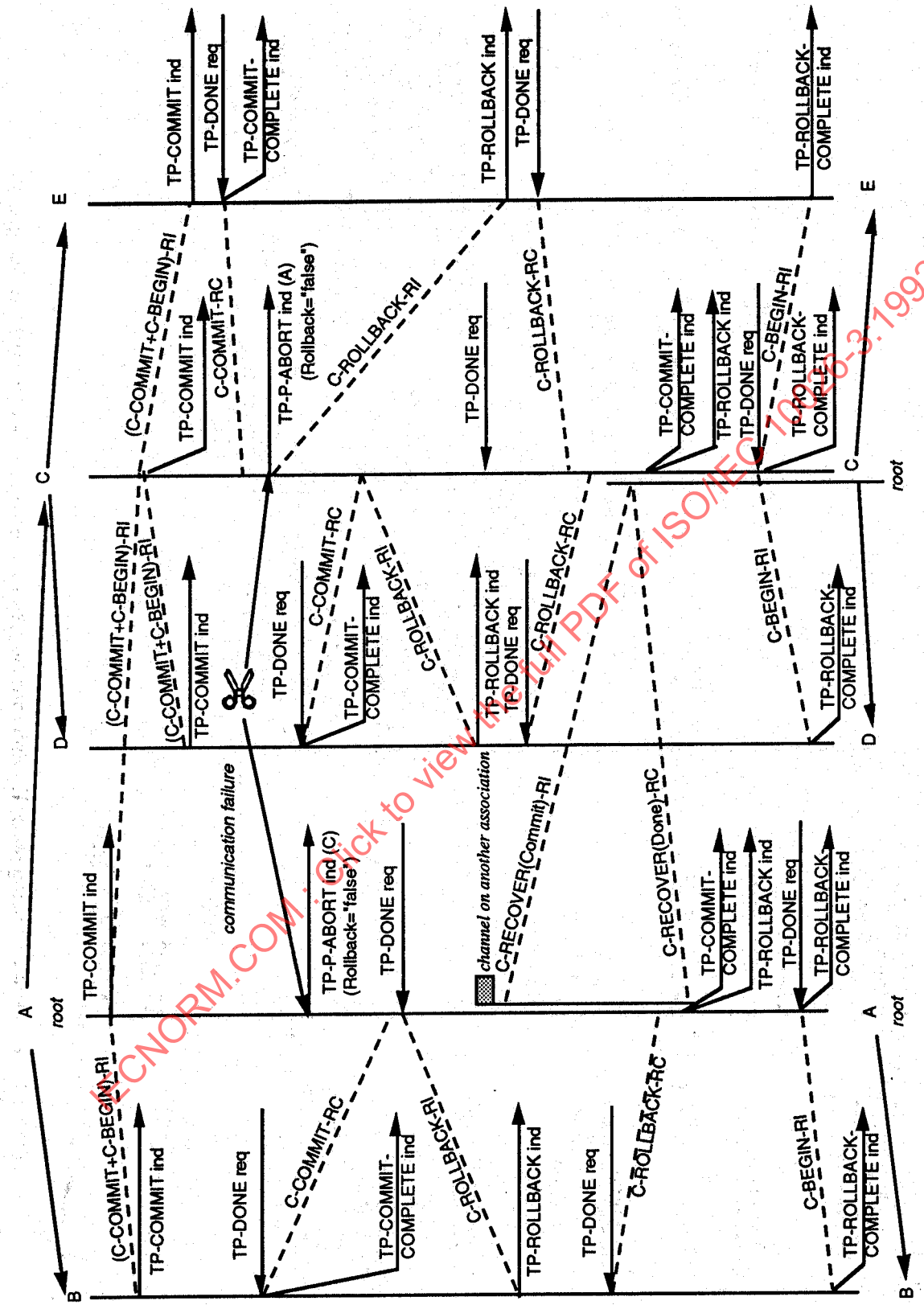


Figure C.80 — Failure during phase II - Isolated node becomes root (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.7.10 Dialogue abort with the superior after TP-COMMIT indication has been issued, node subordinate to the failed dialogue aborts its other dialogues

The beginning of the scenario of figure C.81 is similar to the beginning of the previous one. But after being aware of the failure with its superior, C decides to abort the dialogue it has with its two subordinates (D and E) by issuing TP-U-ABORT request on these dialogues.

The next transaction is rolled back in both transaction trees.

At the end of the scenario, C, D and E are isolated, A and B form a transaction tree with A as the root.

C.7.11 Dialogue abort with the superior after TP-COMMIT indication has been issued, node superior to the failed dialogue aborts its other dialogues

In the scenario of figure C.82, a failure occurs between C and its subordinate, E, after C has received TP-COMMIT indication. E has not received TP-COMMIT indication and will receive it after recovery.

D and E, which have already received TP-COMMIT-COMPLETE indication and can have begun to perform some actions in the context of the next provider-supported transaction, receive TP-ROLLBACK indication and enter the rollback procedure. A also enters the rollback procedure.

C decides to abort the dialogues it has with its superior (A) and its subordinate (D) by issuing TP-U-ABORT request on these dialogues.

At the end of the scenario, C and D are isolated, A and B form a transaction tree with A as the root, and the E, since it has not performed any action related to this failure, is the root of a new transaction tree.

C.7.12 Dialogue abort with the superior after TP-COMMIT-COMPLETE indication has been issued, node superior to the failed dialogue aborts its other dialogues

In the scenario of figure C.83, a failure occurs between B and its subordinate, C, after C has received TP-COMMIT-COMPLETE indication.

C receives the TP-P-ABORT indication with the Rollback parameter set to "true" while being in the next transaction.

B receives, the TP-P-ABORT indication with the Rollback parameter set to "false" before receiving TP-COMMIT-COMPLETE indication, but the next transaction is rolled back.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

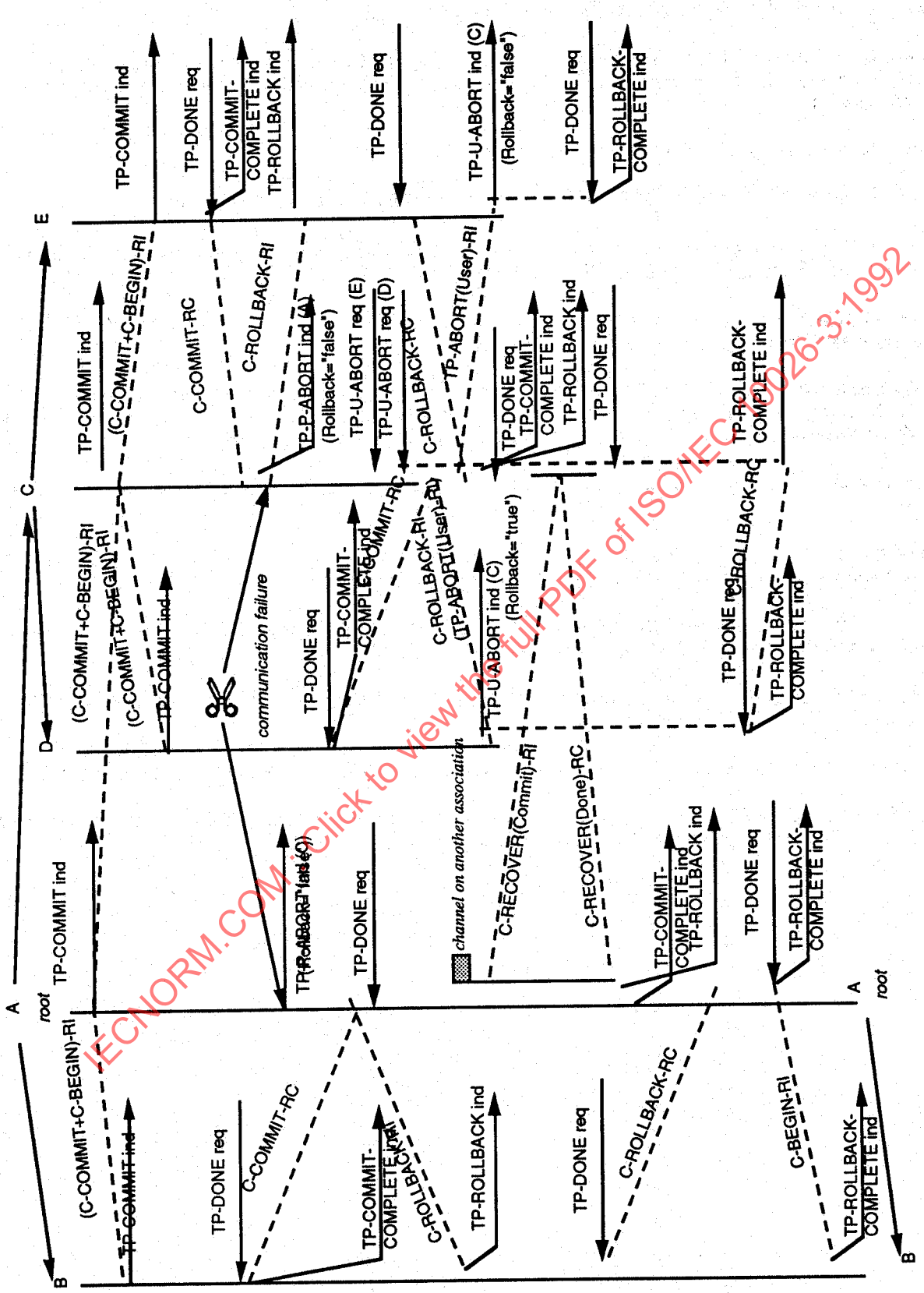


Figure C.81 — Failure during phase II - Intermediate aborts all dialogues (Chained Transactions)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

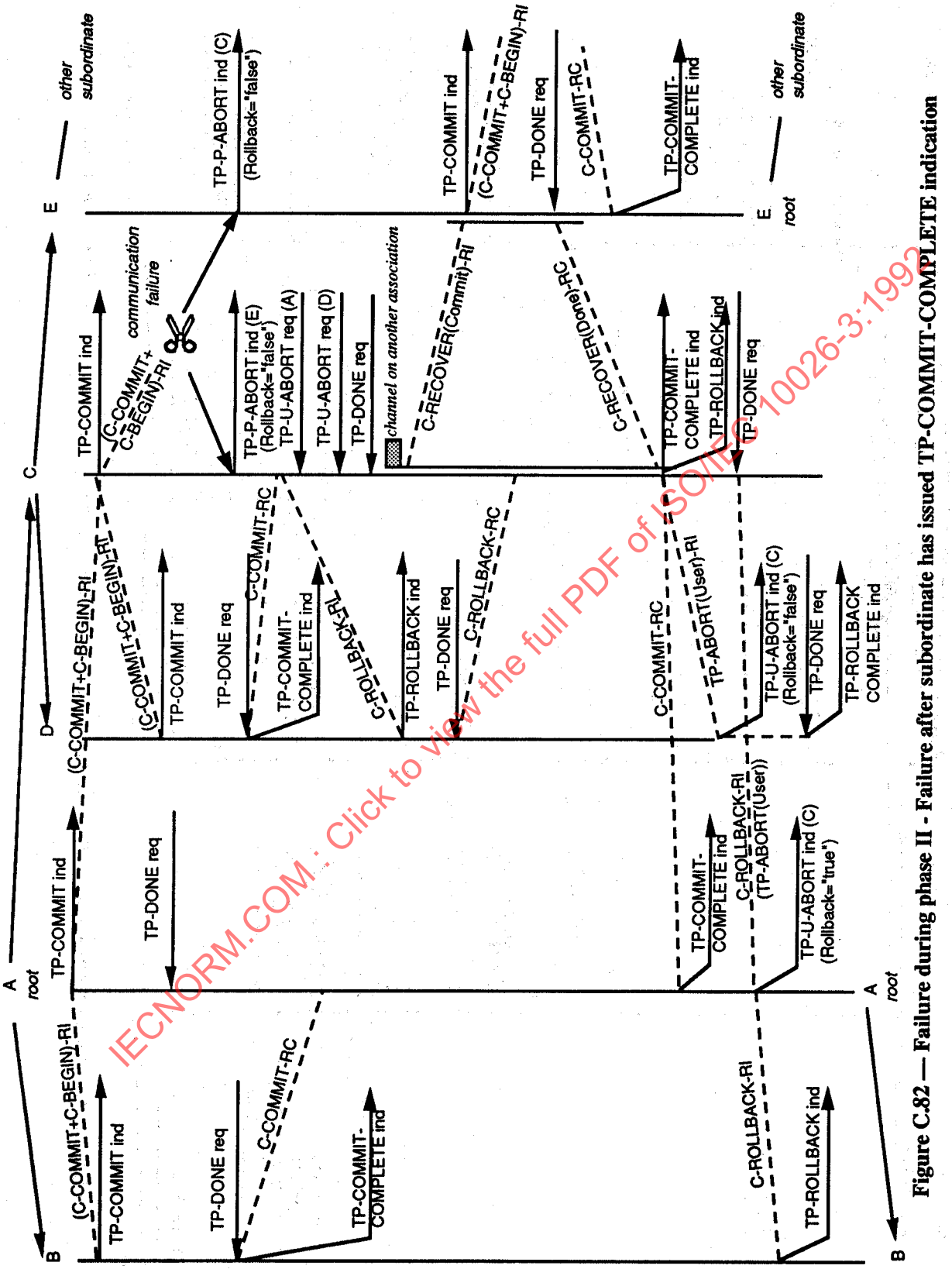


Figure C.82 — Failure during phase II - Failure after subordinate has issued TP-COMMIT-COMPLETE indication

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

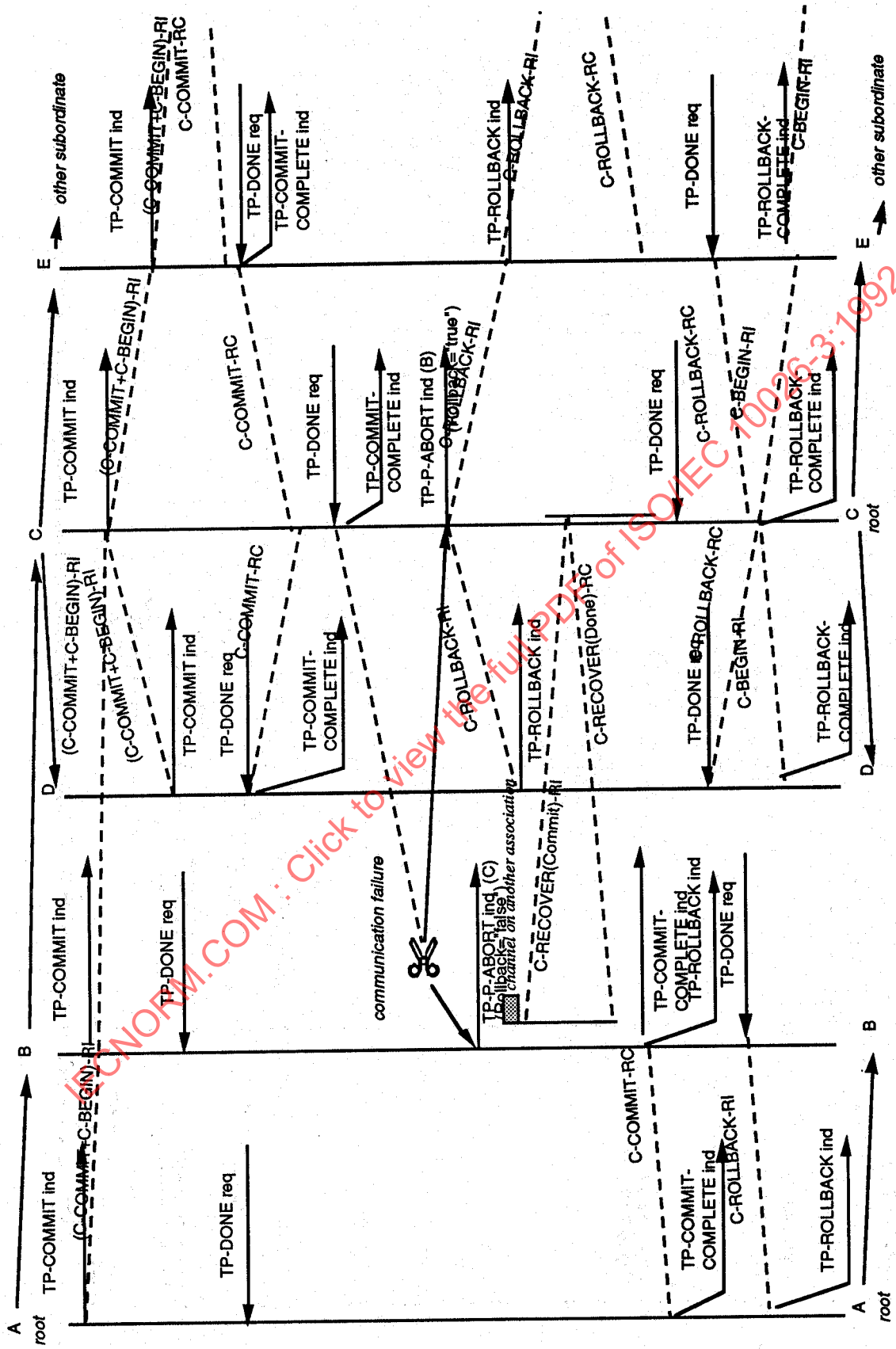


Figure C.83 — Failure during phase II - Failure after subordinate has issued TP-COMMIT-COMPLETE indication

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8 Heuristic decisions and reporting

In all cases other than scenario of figure C.90, a subordinate has issued TP-COMMIT request and then has not received either TP-COMMIT indication or TP-ROLLBACK indication. Its patience exhausted, the subordinate takes a heuristic decision. The following scenarios are provided:

C.8.1 Heuristic decisions when the outcome of the transaction is commit

C.8.1.1 Subordinate rolls back, mix detected

The scenario of figure C.84 shows a case where the subordinate decides to heuristically roll back; since the outcome is commit, a recovery occurs and a heuristic-mix report is given to the superior.

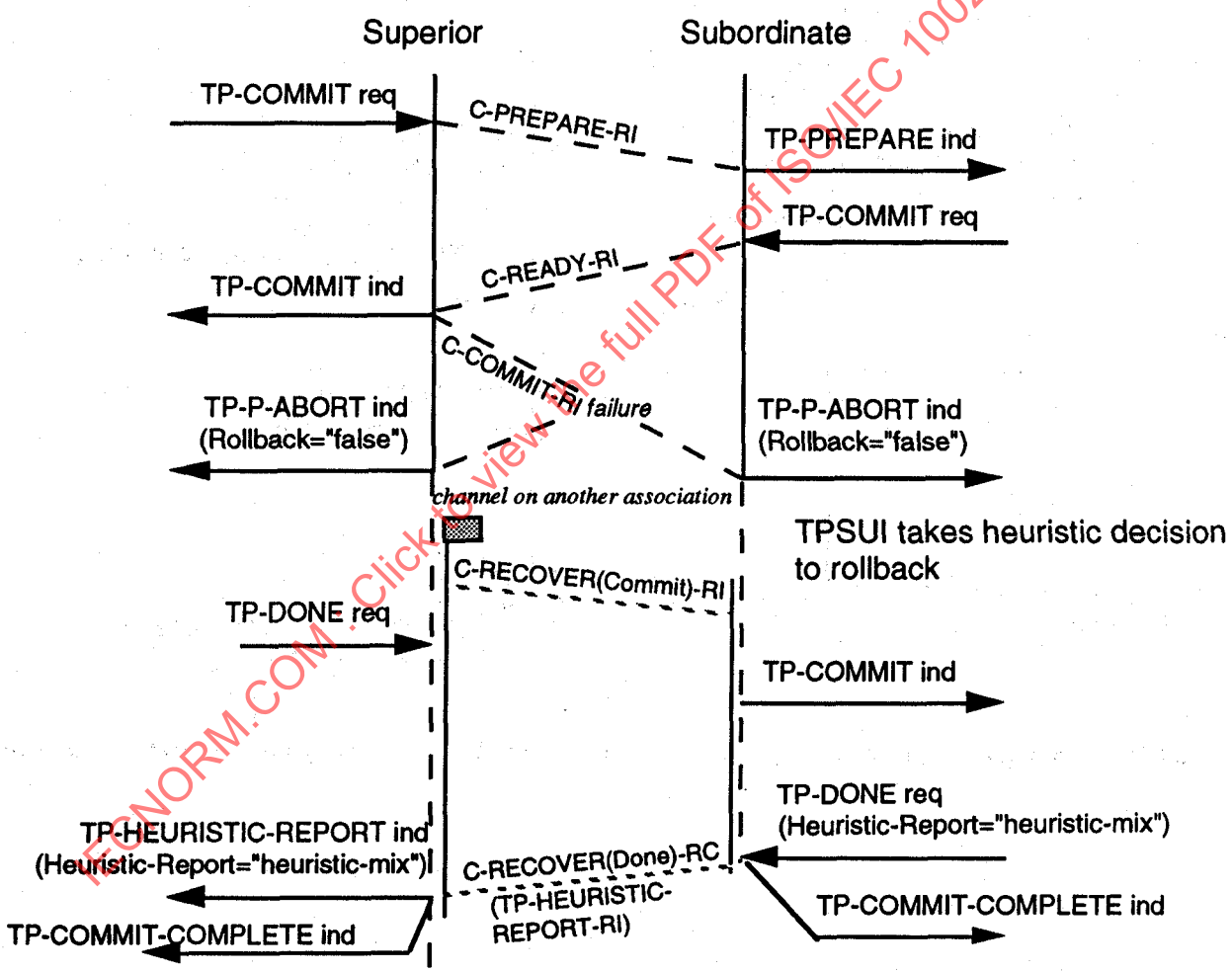


Figure C.84 — Dialogue aborted during commitment, subordinate takes (wrong) decision to rollback

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.1.2 Subordinate commits, no damage

The scenario of figure C.85 shows a case where the subordinate decides to heuristically commit; since the outcome is commit, a recovery occurs and no damage exists.

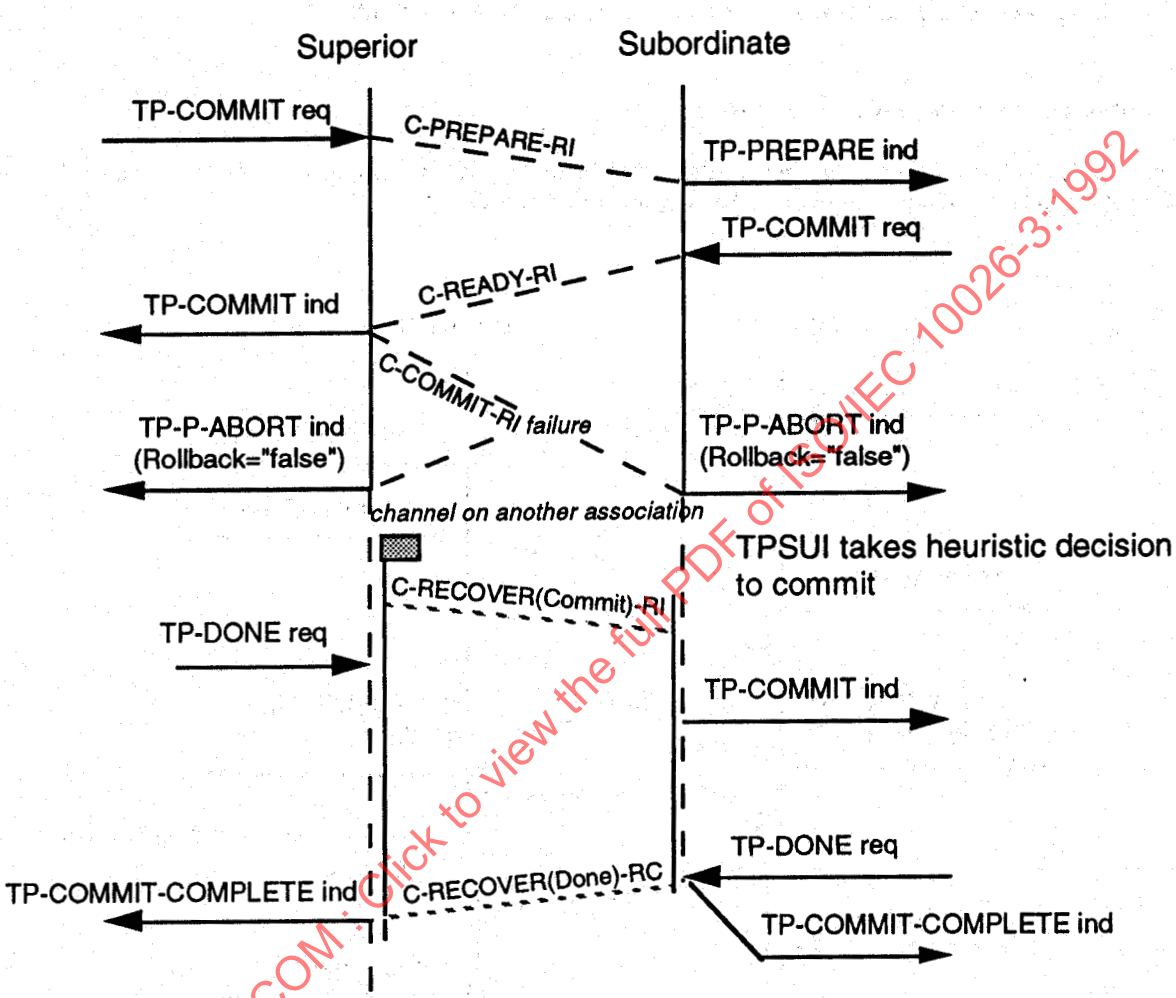


Figure C.85 — Dialogue aborted during commitment; subordinate takes (correct) decision to commit

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.1.3 Subordinate rolls back but is able to compensate, no damage

The scenario of figure C.86 shows a case where the subordinate decides to heuristically rollback; since the outcome is commit, a recovery occurs, but, since the subordinate is able to compensate, no damage is reported.

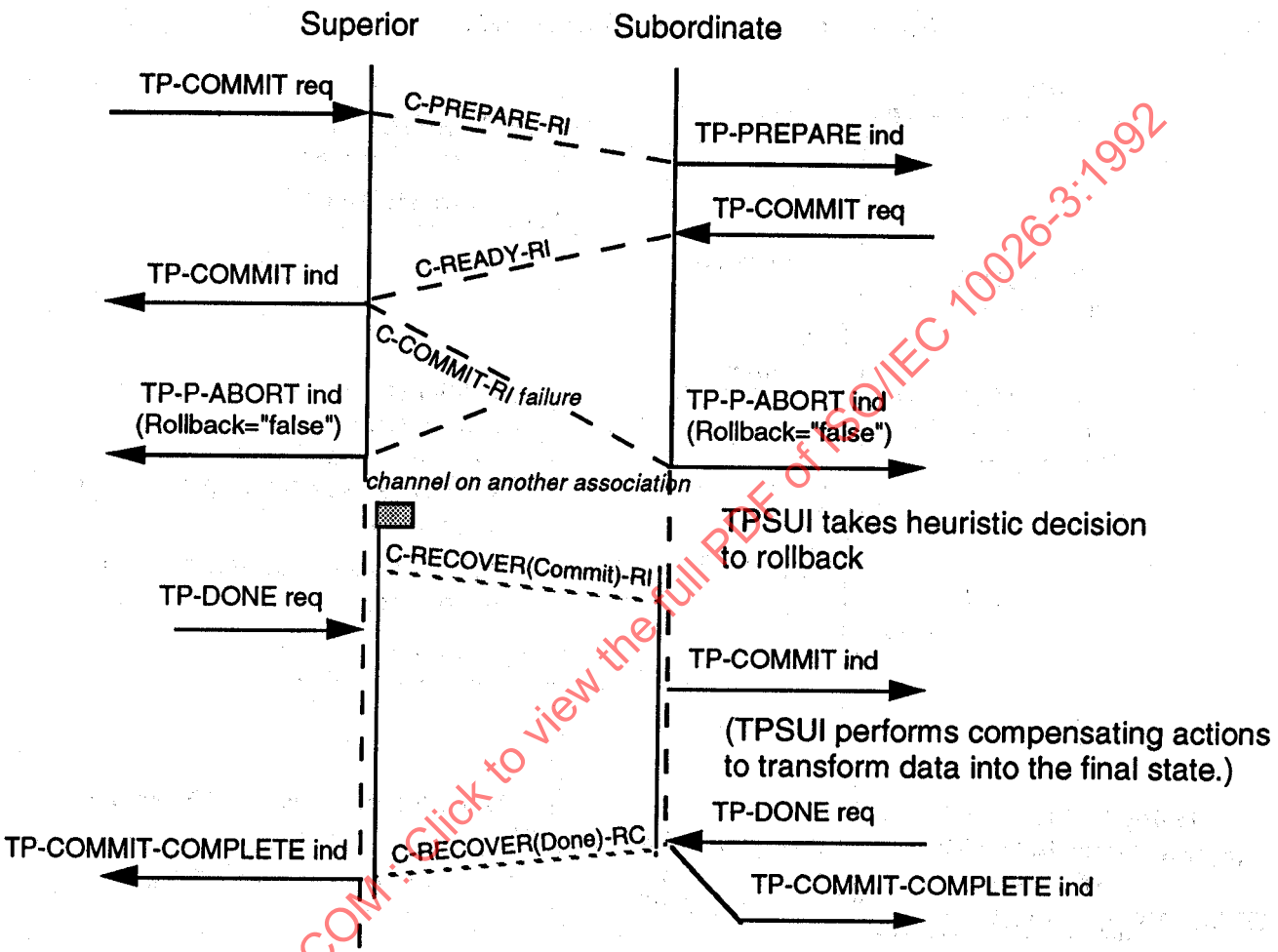


Figure C.86 — Dialogue aborted during commitment; subordinate takes (wrong) decision to rollback but is able to compensate

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.1.4 Subordinate TPSUI commits, TPSP rolls back, mix detected

The scenario of figure C.87 shows a case where the subordinate TPSUI decides to heuristically commit whereas the TPSP decides to roll back; since the outcome is commit, a recovery occurs and a heuristic-mix report is given to the superior.

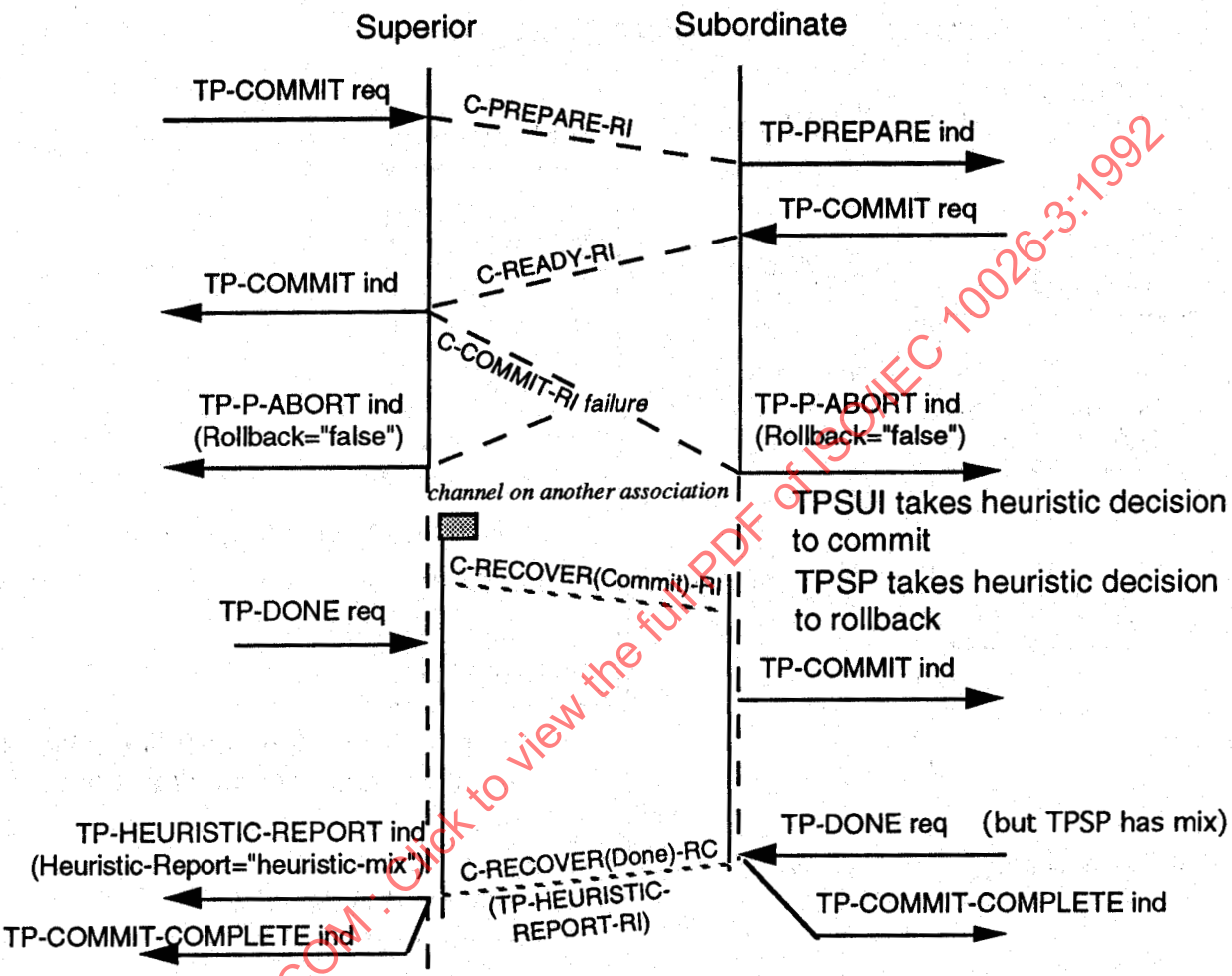


Figure C.87 — Dialogue aborted during commitment; subordinate takes contradicting heuristic decisions producing an internal mix

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.2 Heuristic decisions when the outcome of the transaction is rollback

C.8.2.1 Subordinate commits, heuristic-hazard reported

The scenario of figure C.88 shows a case where the subordinate decides to heuristically commit; since the dialogue has failed with the superior before TP-COMMIT indication is issued, the superior rolls back and no recovery occurs, a heuristic-hazard report is given to the superior.

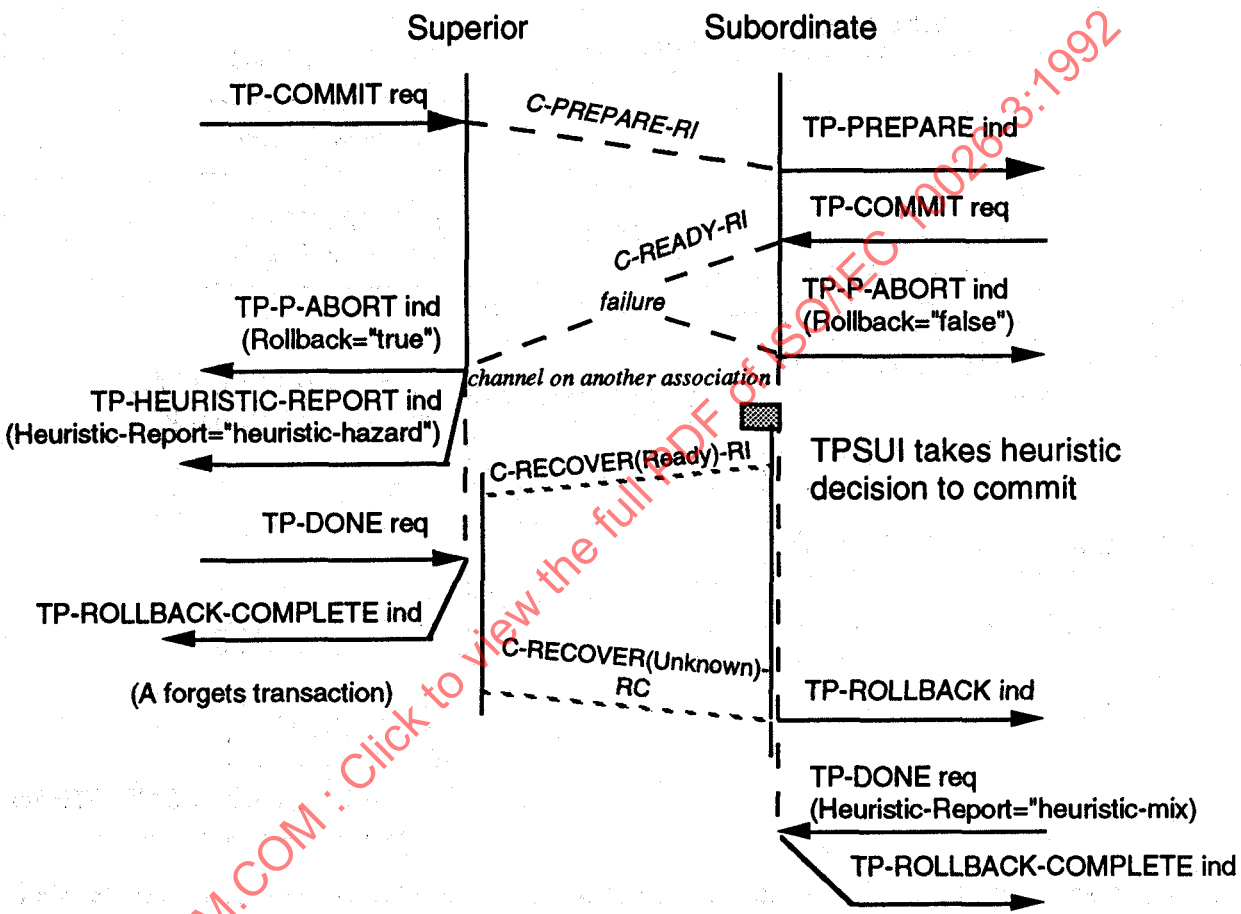


Figure C.88 — Dialogue aborted during commitment; subordinate takes (wrong) heuristic decision to rollback

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.2.2 Subordinate rolls back, no damage, heuristic-hazard reported

The scenario of figure C.89 shows a case where the subordinate decides to heuristically rollback; since the dialogue has failed with the superior before TP-COMMIT indication is issued, the superior rolls back and no recovery occurs, there is no damage but a heuristic-hazard report is given to the superior.

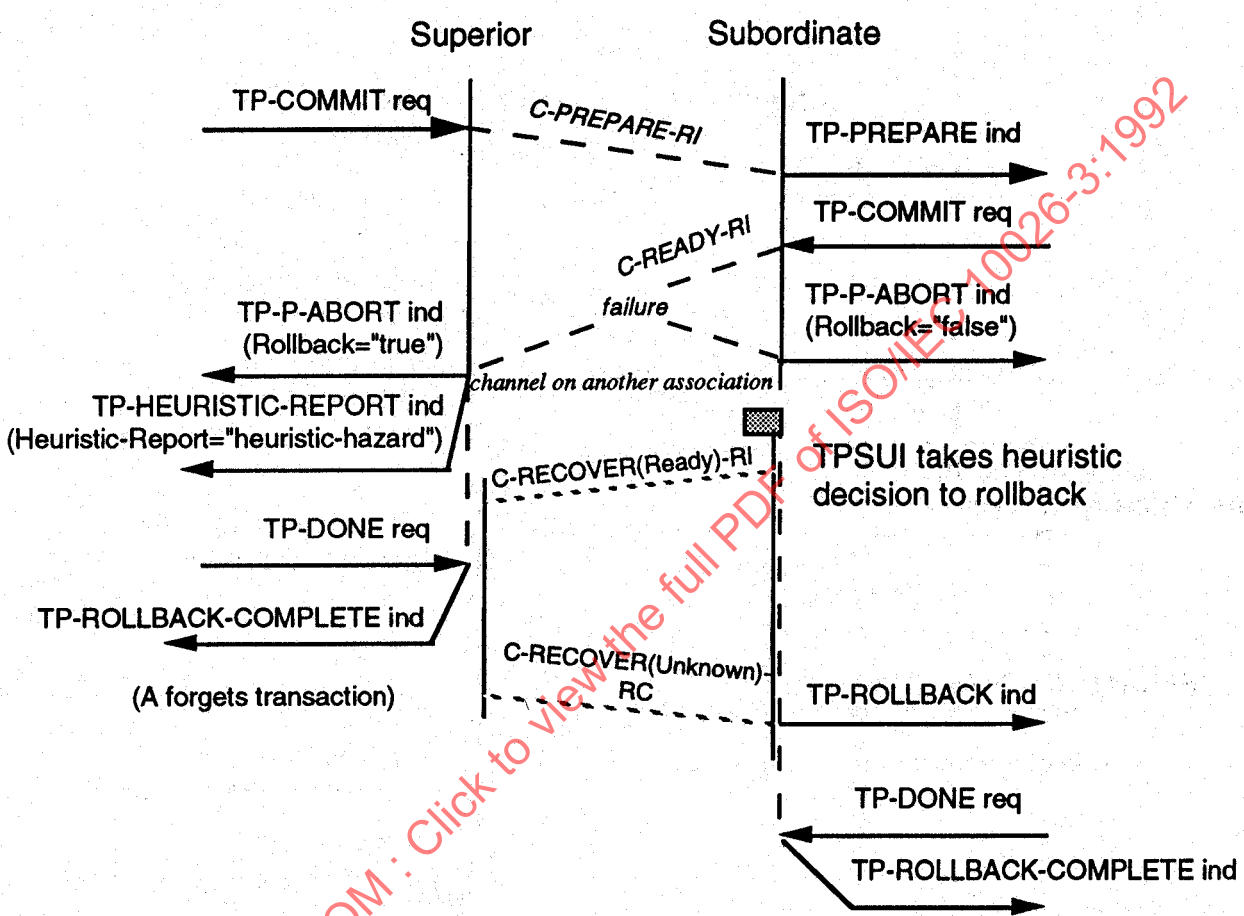


Figure C.89 — Dialogue aborted during commitment; subordinate takes (correct) decision to rollback but a heuristic-damage is reported

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.2.3 Subordinate rolls back, no damage, heuristic-hazard reported

The scenario of figure C.90 shows a case where the subordinate receives a TP-P-ABORT indication with Rollback set to "true" in the active phase. No heuristic decision has been taken but, since the superior has issued TP-COMMIT request, a heuristic-hazard is reported.

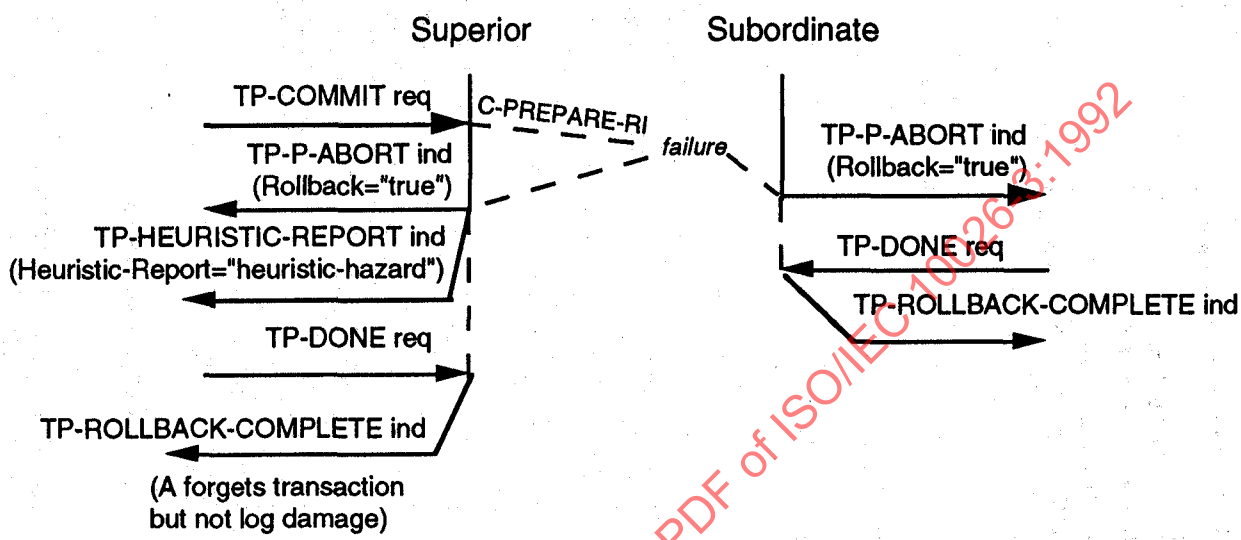


Figure C.90 — Dialogue aborted before subordinate is aware of transaction termination; no heuristic decision taken but a heuristic-damage is reported

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.8.3 Heuristic decisions in trees with multiple dialogues

The following scenarios show three examples of propagation of heuristic report within a tree having three nodes.

The scenario in figure C.91 shows a case where the leaf TPSUI decides to heuristically rollback while the outcome of the transaction is commit. The heuristic-mix report is given to the TPSUIs at B node and at A node.

The scenario in figure C.92 shows a similar case except that the A node is able to compensate and does not issue any heuristic report to its TPSUI.

The scenario in figure C.93 shows a case where the leaf TPSUI decides to heuristically commit while the outcome of the transaction is rollback. The heuristic-mix report is given to the TPSUIs at B node and A node.

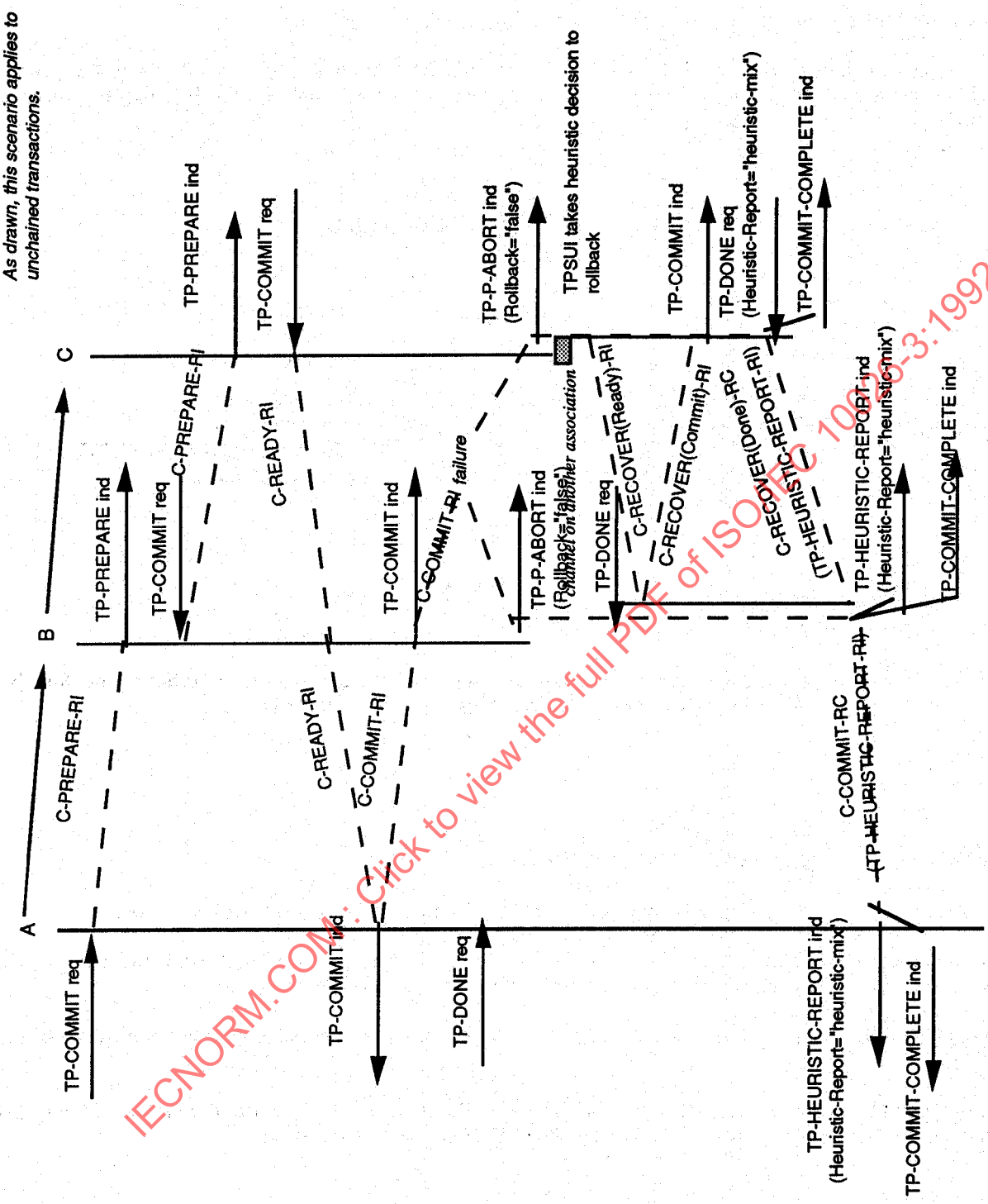


Figure C.91 — Heuristic decisions and reporting in a multi-dialogue tree
(all nodes above failure receive heuristic report)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

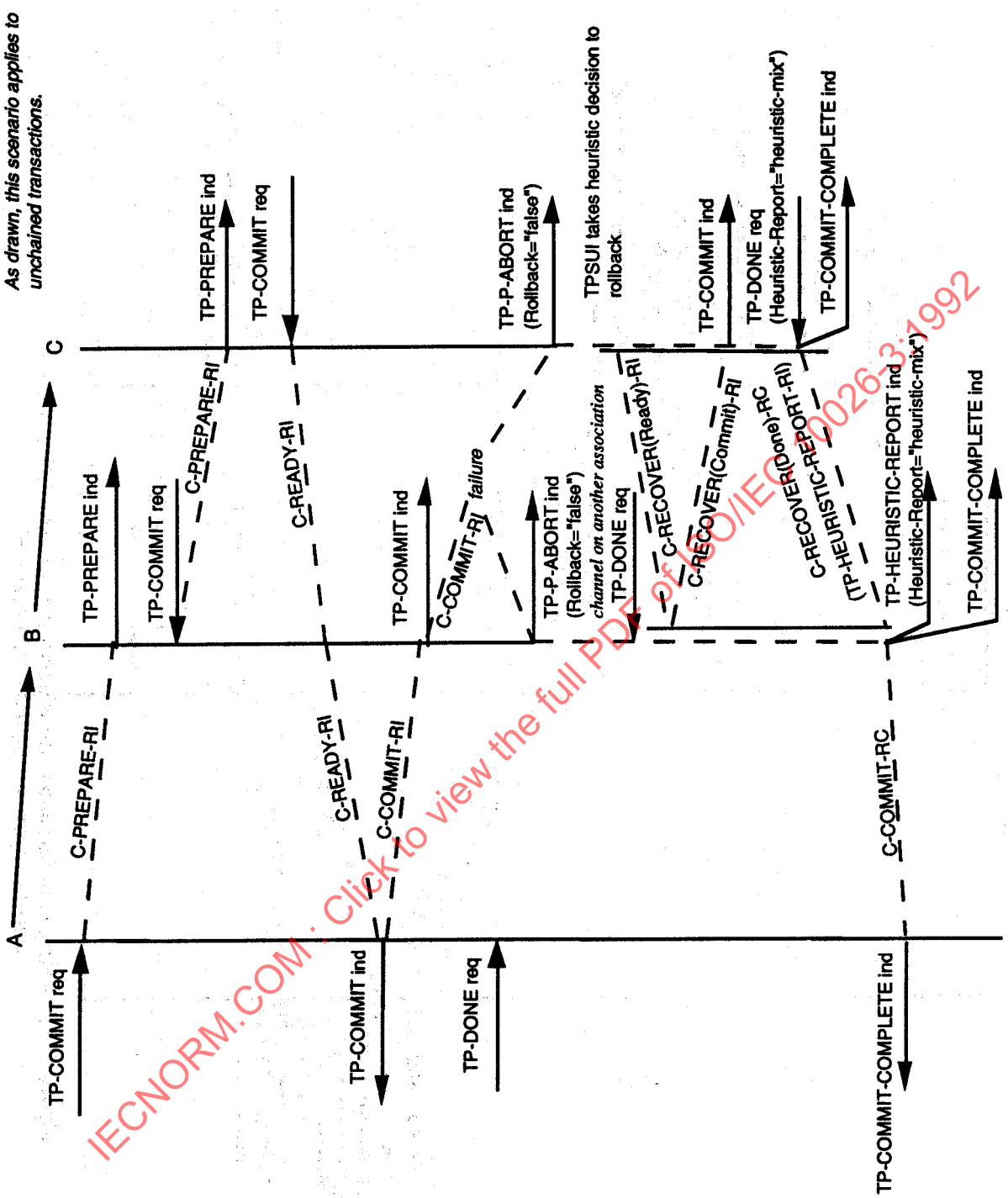


Figure C.92 — Heuristic decisions and reporting in a multi-dialogue tree
(node B compensates before reporting)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

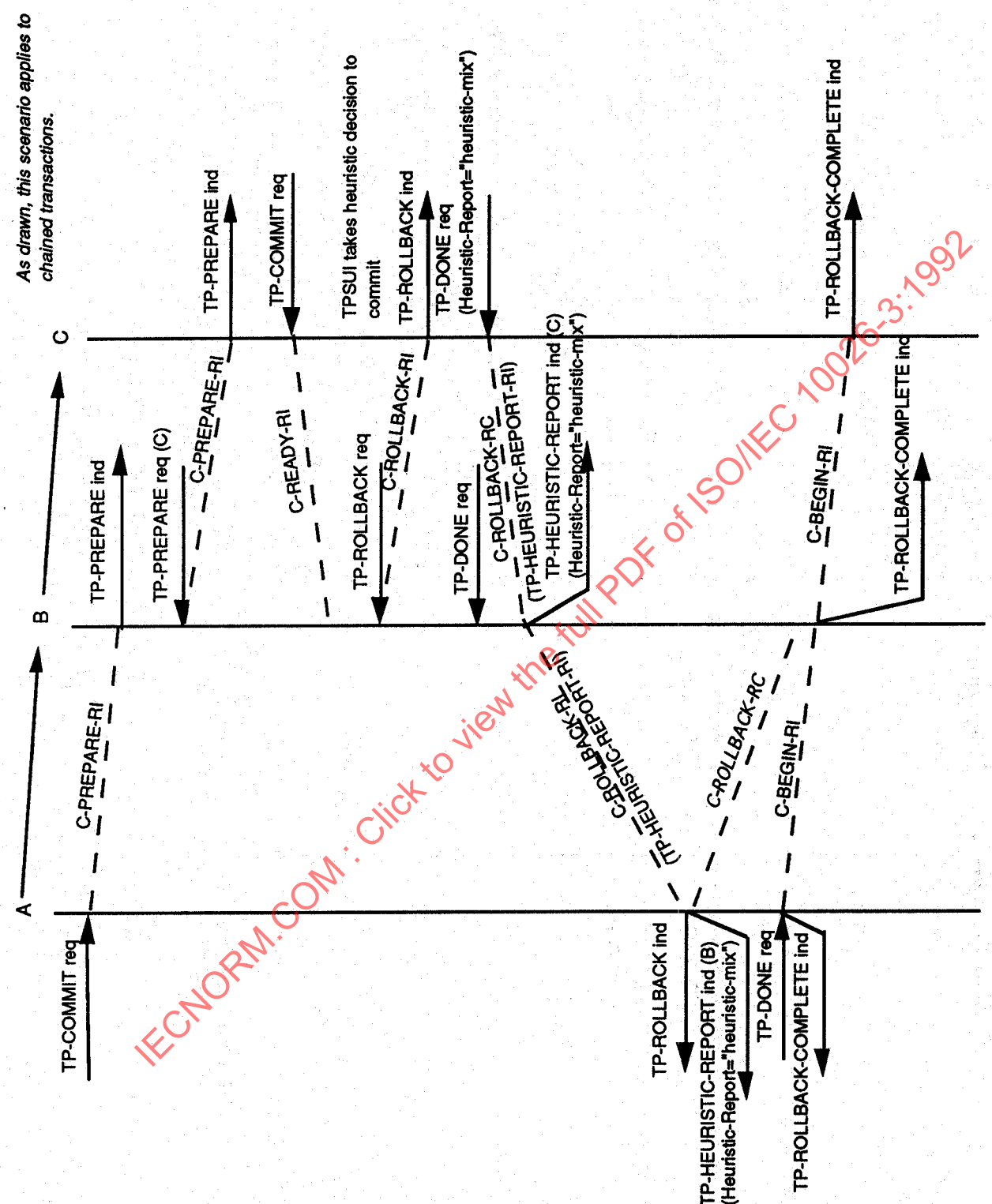


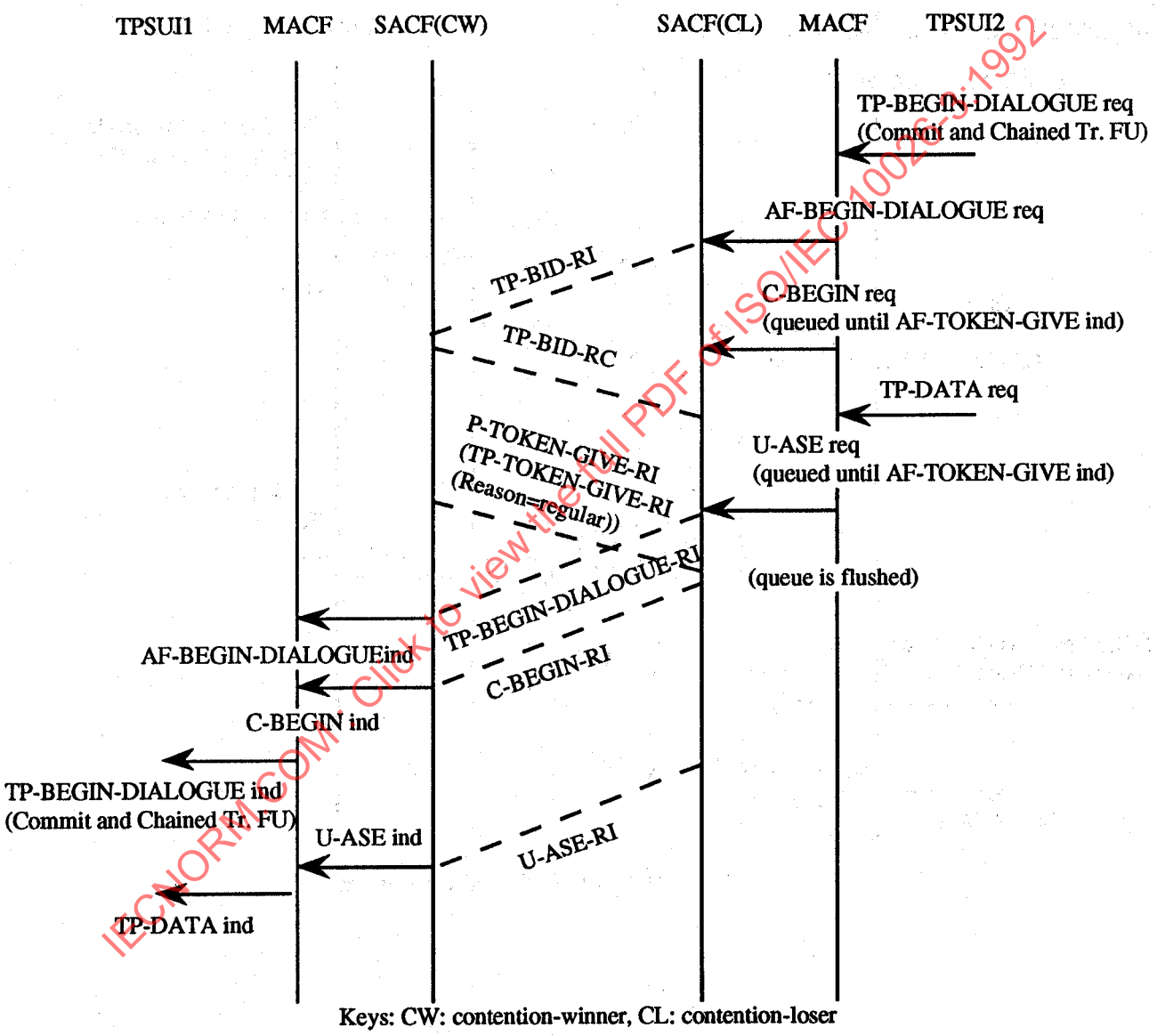
Figure C.93 — Heuristic decisions and reporting in a multi-dialogue tree
(heuristic report during rollback procedure)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.9 Scenarios for SACF

The scenario of figure C.94 shows a case where a contention-loser uses BID to establish a dialogue with provider-supported transaction.

The scenario of figure C.95 shows a case where a contention-loser does not use BID to establish a dialogue with provider-supported transaction. Figure C.95 is a complicated case where stray APDUs occurs.



Keys: CW: contention-winner, CL: contention-loser

Figure C.94 — BID used in the commit

IECNORM.COM Click to view the full PDF of ISO/IEC 10026-3:1992

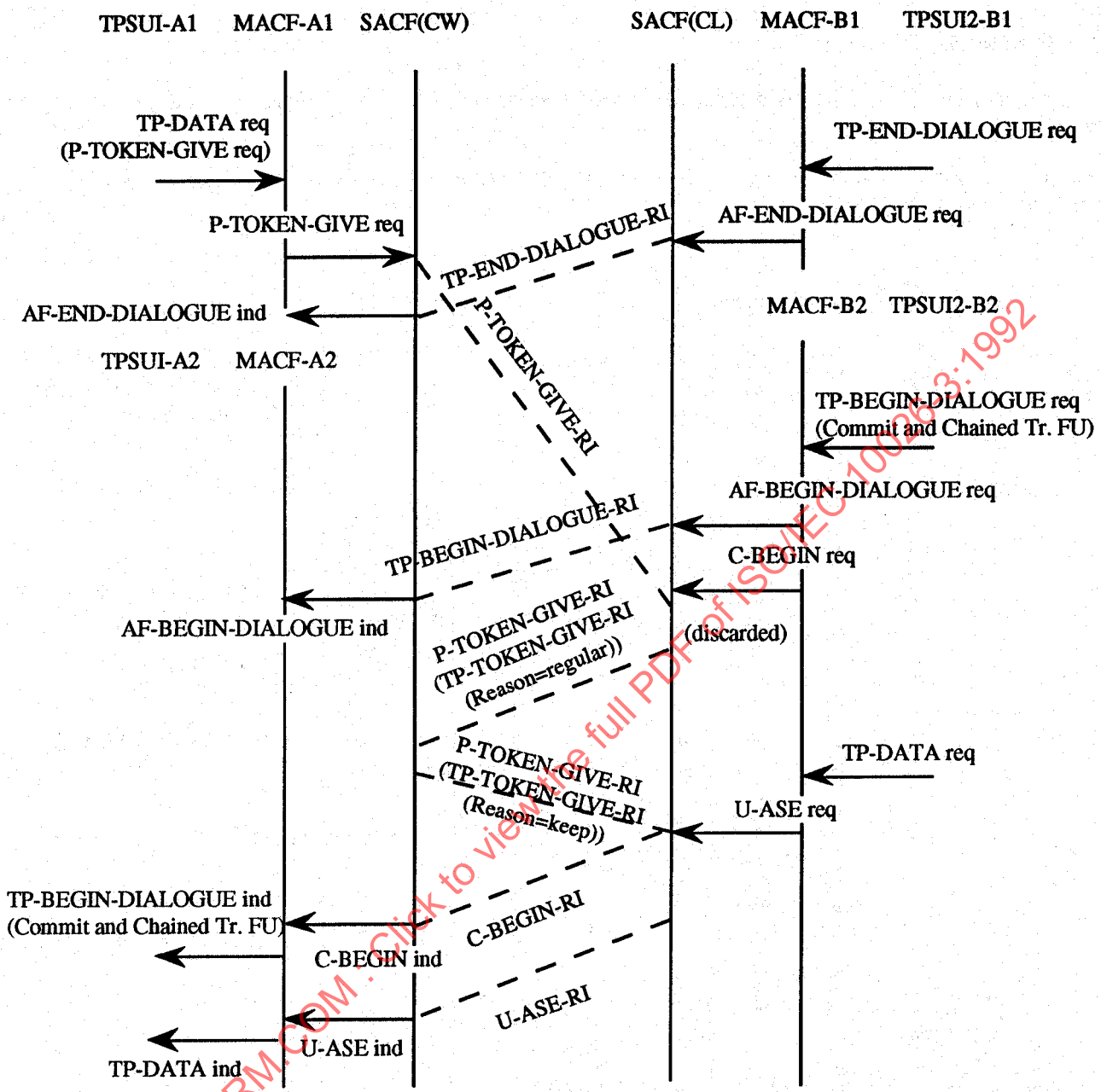


Figure C.95 — BID not used in the commit

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.10 Scenarios for CPM

C.10.1 Scenarios for channel establishment

The scenario of figure C.96 shows a simple sequence for channel establishment from a contention-winner.
The scenario of figure C.97 shows a simple sequence for channel establishment from a contention-loser.
The scenario of figure C.98 shows a complex sequence for channel establishment from a contention-loser.
In this figure, due to stray APDUs, the token is delayed.

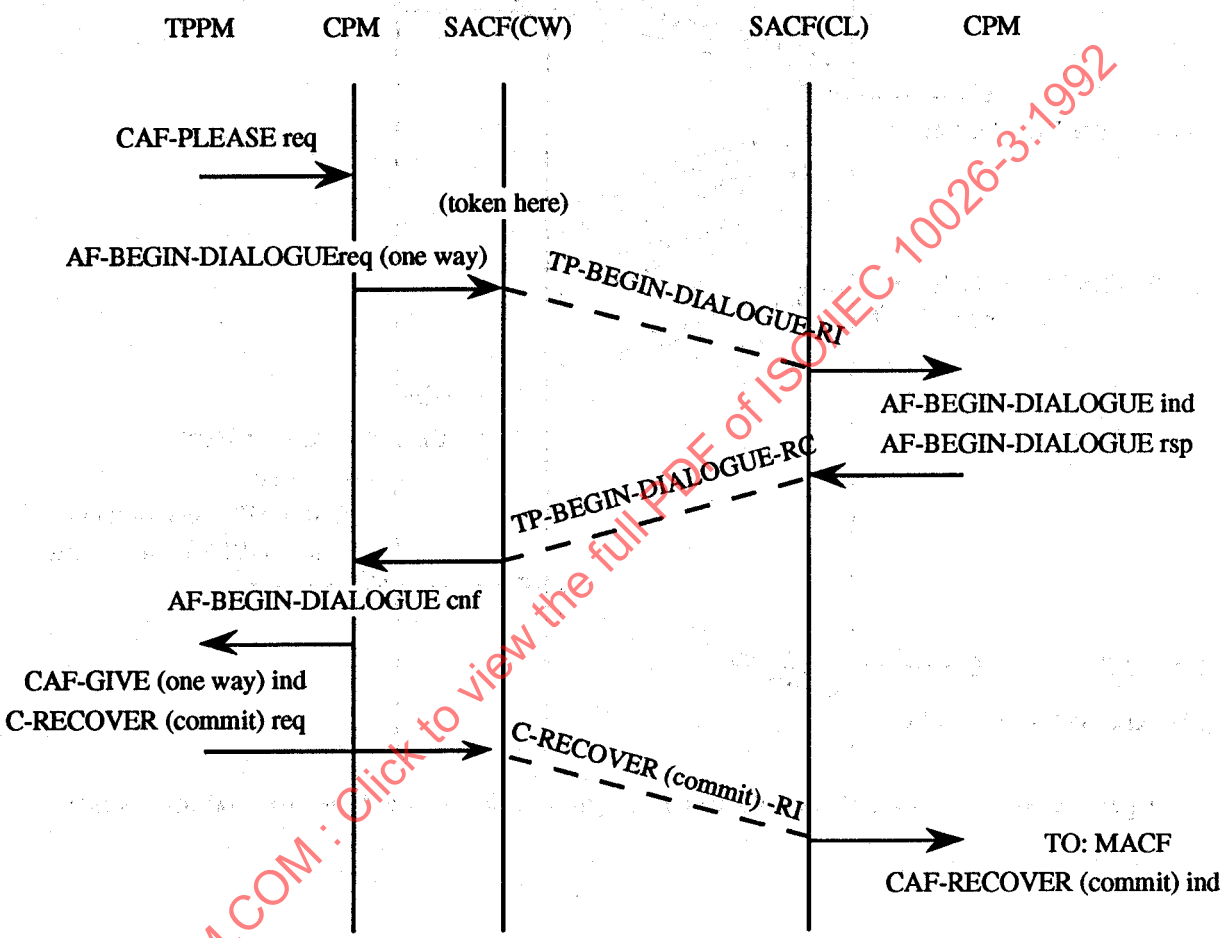


Figure C.96 — Channel establishment requested by contention-winner (simple case)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

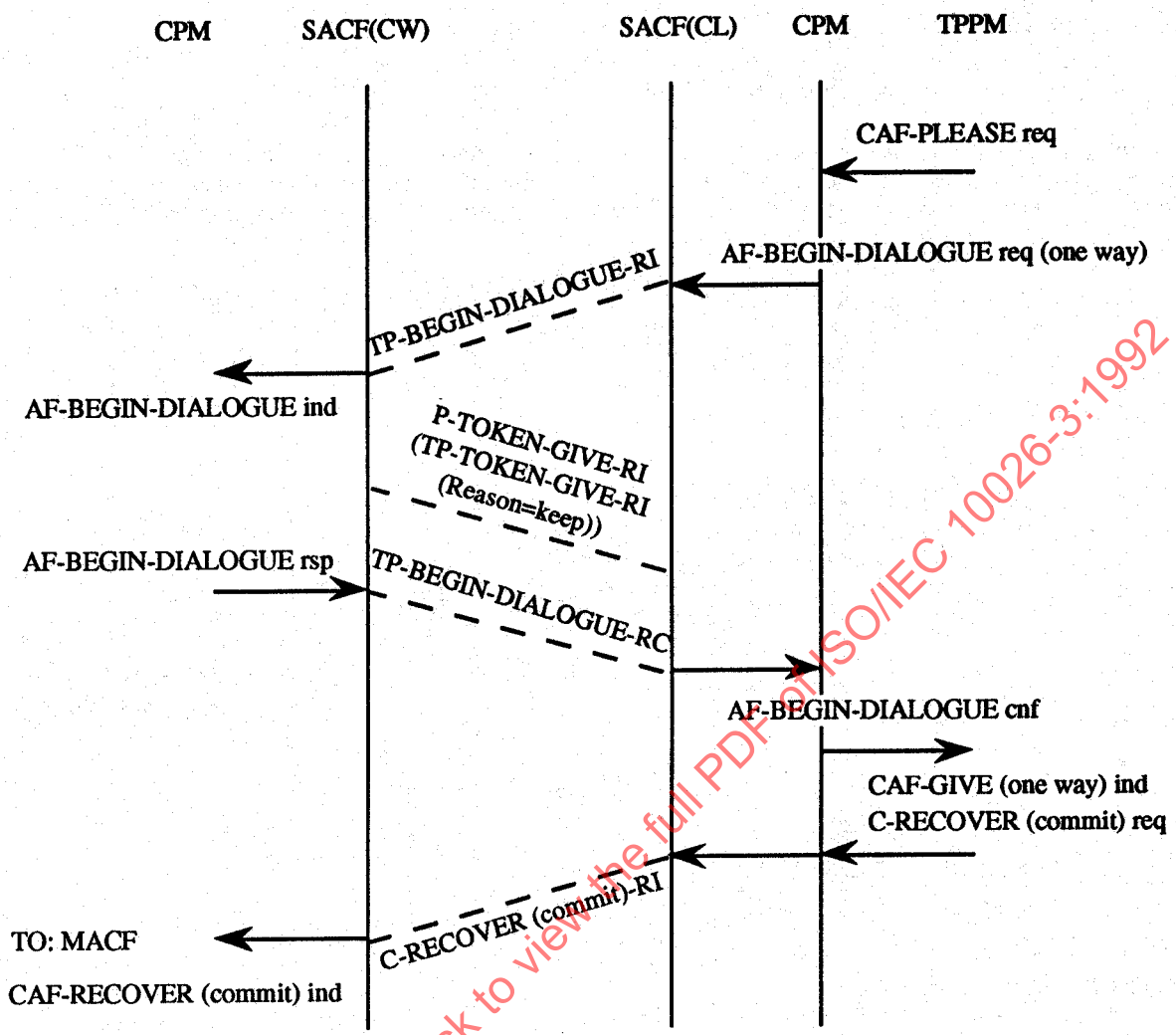
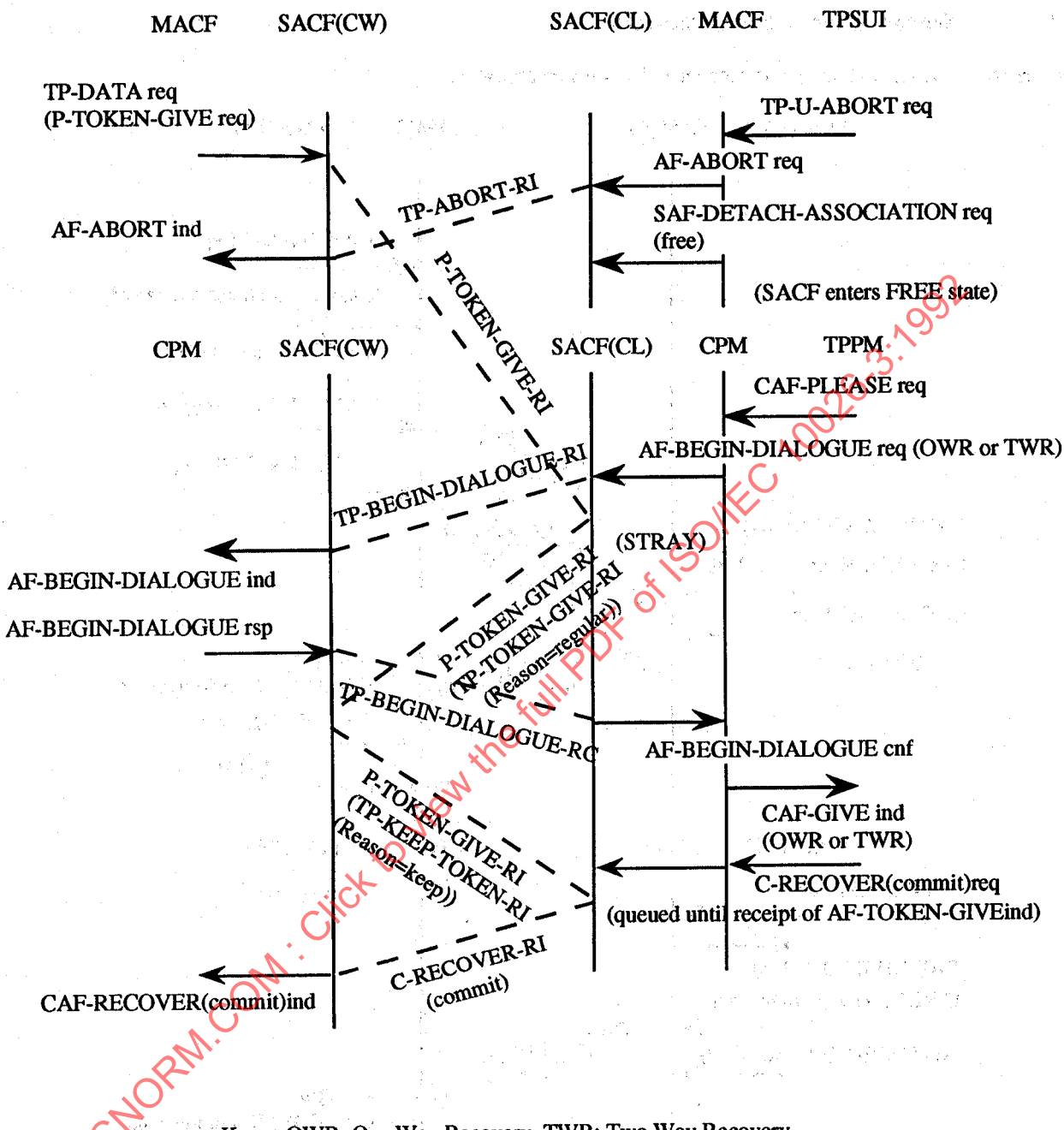


Figure C.97 — Channel establishment requested by contention-loser (simple case)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



Keys : OWR: One Way Recovery, TWR: Two Way Recovery

Figure C.98 — Channel establishment requested by contention-loser (complex case)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

C.10.2 Scenarios for two way recovery

The scenario of figure C.99 shows some sequences for two way recovery.

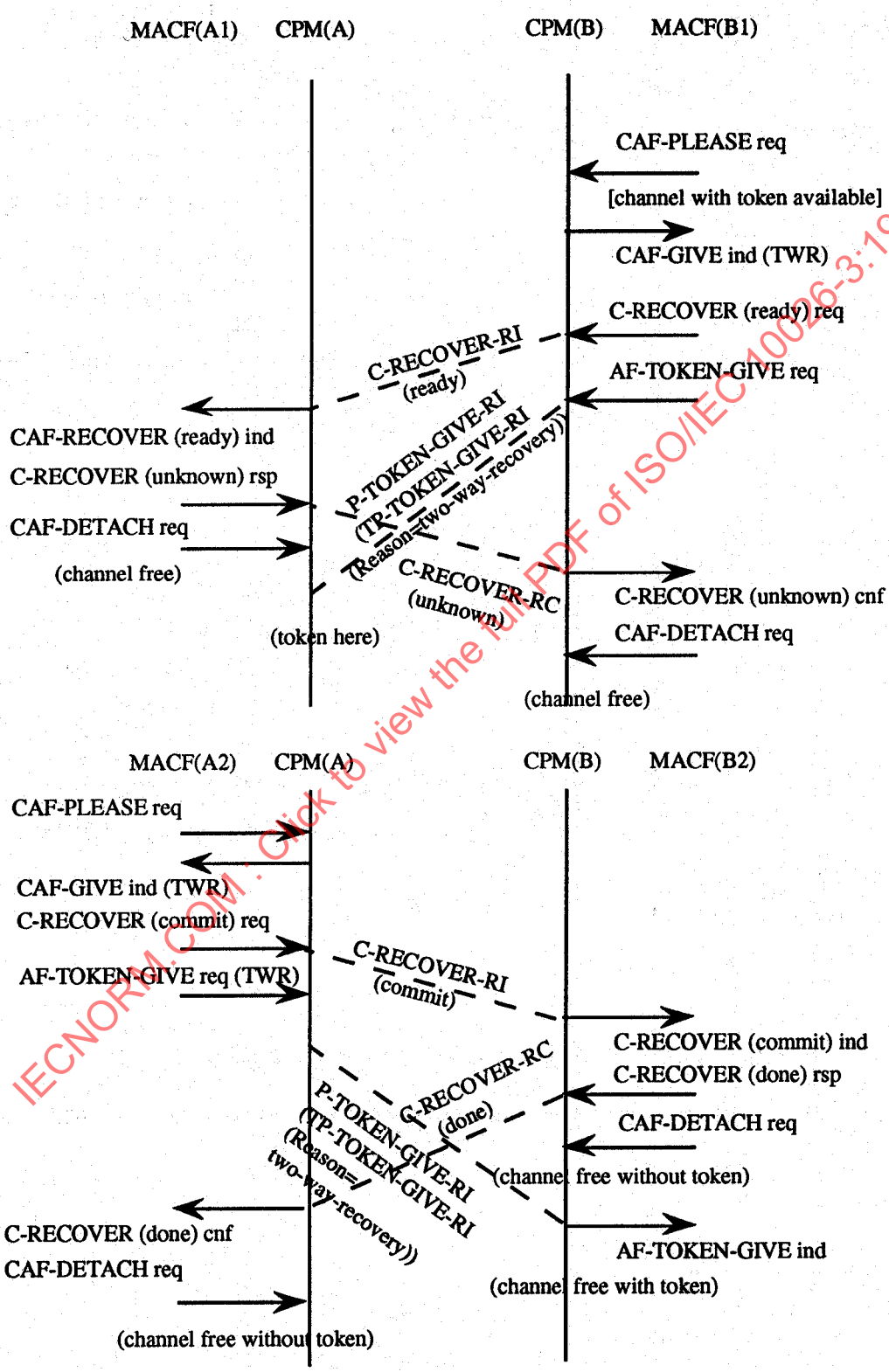
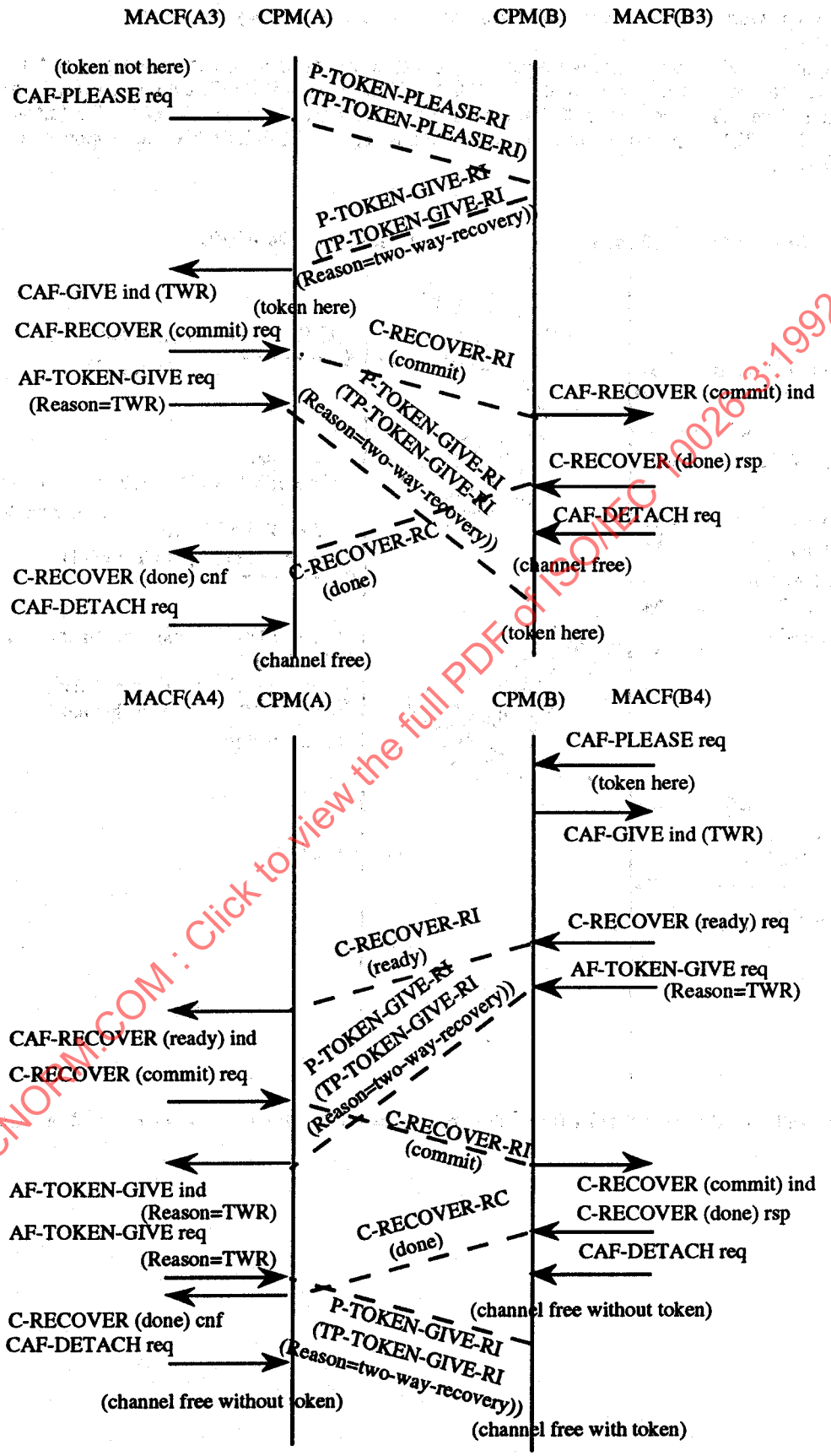


Figure C.99 — Two way recovery

continued on next page

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure C.99 — Two way recovery

C.10.3 Scenarios for recovery collision on separate one way channels

The scenario of figure C.100 shows a case when recovery collison occurs because both a superior and a subordiante initiate recovery. This case can occur when, for example, C-READY-RI or C-COMMIT-RI cannot reach the partner MACF due to the communication failure. When recovery collision occurs, the superior always issues C-RECOVER (retry-later) response as a response to CAF-RECOVER(ready)ind and the subordinate always issues C-RECOVER(done)response as a response to CAF-RECOVER(commit)ind.

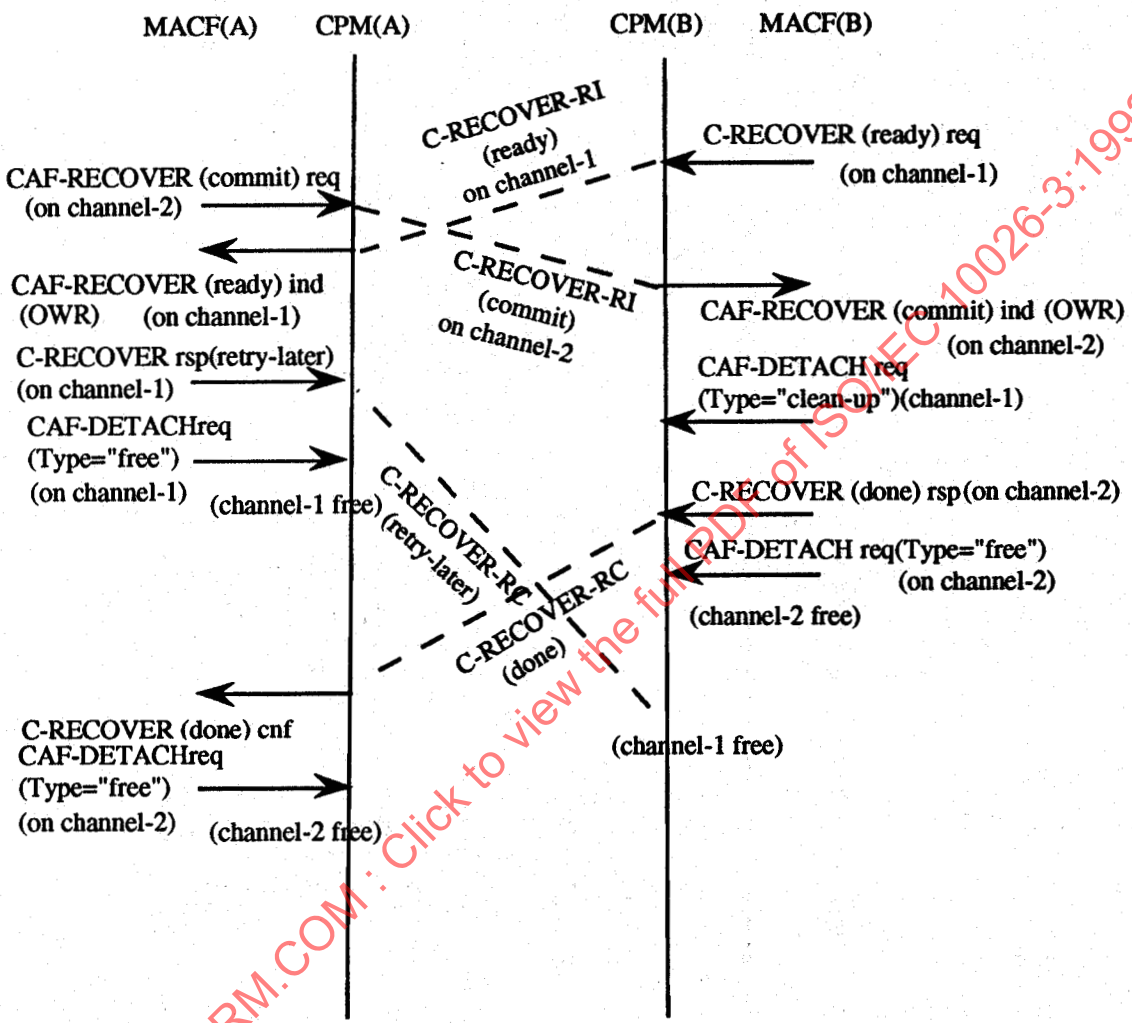


Figure C.100 — Recovery collision due to requests by both a superior and a subordinate

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex D
(informative)

Summary of assigned object identifier values

This annex summarizes the OBJECT IDENTIFIER values assigned in Recommendation X.862 and this part of ISO/IEC 10026.

```
tpASE1 APPLICATION-SERVICE-ELEMENT ::=
      {joint-iso-ccitt transaction-processing(10) ase-id(0) tp-ase(1) version1(0)}

      -- may be used to identify the transaction processing ASE defined in X.862 and
      -- ISO/IEC 10026-3.

      {joint-iso-ccitt transaction-processing(10) modules (1)
       apdus-abstract-syntax(1) version1(0)}

      -- used to reference the ASN.1 module defined in 12.1 of X.862 and
      -- ISO/IEC 10026-3.

id-as-tpase OBJECT IDENTIFIER ::=
      {joint-iso-ccitt transaction-processing(10) abstract-syntax(2) tp-apdus(1)}

      -- may be used to reference the abstract syntax of the
      -- transaction processing ASE APDUs
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex E
(informative)

Recovery from destruction of atomic action data

E.1 Introduction

The correct operation of OSI TP provider supported transactions is dependent on the availability of a reliable recovery log as described in 10026-1. The loss of some or all of the recovery log would mean that a system can not respond to requests to recover any or selected transactions; such failures could include total destruction of a real system with all its data, or partial failure such as loss of part or all of a recovery log.

If such a failure occurs during the later phases of a distributed transaction, then recovery of the transaction in the other nodes in the transaction tree would be unable to proceed; transactions will remain in the READY or DECIDED (COMMIT) state (depending on the role of the individual nodes) and recovery actions will either be rejected if a TP channel can not be established or receive a "retry-later" response.

Intermediate or leaf nodes which are affected because a superior node has suffered such a failure, may take heuristic decisions but that does not change the requirement to complete recovery and enter the "transaction forgotten" state. Such a node will have a log-ready record (and a log-heuristic record if a heuristic decision was taken) but using the mechanisms of OSI TP will not be able to determine the final outcome of the transaction (and whether any heuristic decision was correct or not).

Loss of atomic action data could occur as a result of several failures:

— partial loss of recovery log (for example due to media errors);

— complete loss of recovery log (for example due to a disc crash);

— destruction of system (for example by fire or flood);

— human operator error.

Because of the dependence on the recovery log for the correct operation of distributed transaction processing, many real systems will maintain multiple copies of the recovery log to minimize the chances of loss of all copies and may take other precautions against destruction of a system for example by dispersing the multiple copies across several geographical locations.

This annex is concerned with the case that in spite of measures to protect the recovery log contents, some loss has occurred.

E.2 Recovery actions

The following viewpoints are considered:

— a system which has been destroyed;

— a system which has lost part of its recovery log;

— a system which can not complete recovery because of lack of response from a further damaged system.

E.2.1 Actions after a system has been destroyed

This covers the case where a system can not run at all or that it is sufficiently damaged that the TP Protocol machine can not make any response to requests to establish a TP Channel for recovery. Such a system can not take any useful action and is not considered further.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

E.2.2 Actions of a node which has lost recovery records

Such a node must distinguish between recovery requests for transactions that it can be sure it has no record of and therefore must have completed as defined in the standard, and recovery requests for transactions where the inaccessible recovery records might relate to the transaction in question. A node must not respond "unknown" or "done" for a transaction for which the recovery log record has been or may have been lost as this would imply that the transaction was terminated normally by either commit or rollback.

For example, a node may be certain that any recovery record related to a transaction is not affected by the loss because of the use of a recovery-context-handle which points to a set of recovery records which are unaffected.

Typically an alert would be raised for system management attention when loss of recovery records is detected; a system might also record requests for recovery to which it can not safely respond to assist in the determination of the management actions that are required. Actions to determine the desired outcome of the transaction and to repair damage to applications and their bound data will be application dependent. Eventually the affected node must be repaired such that it can again function normally.

E.2.3 Actions of a node which can not complete recovery

This concerns the case where a system can not complete recovery of a transaction because of lack of response from a further damaged system which could either be a superior or subordinate node. It may be the case that a direct subordinate or superior is undamaged but that recovery is delayed because of a lack of response from a subordinate or superior node further down/up the transaction tree.

Such a node can not directly determine whether recovery is delayed but will be possible eventually, or that recovery will never be possible. The node will observe either that a TP channel to the affected system can not be established or that requests for recovery of the transaction are responded to by 'retry-later'.

In either case, typically after the expiration of a timer, the node may raise an alert for system management attention; this could prompt determination of the reason for lack of response and whether recovery will become possible or will remain impossible because of the conditions highlighted in this annex.

In the latter case, actions to determine the desired outcome of the transaction and to repair damage to applications and their bound data will be system or application dependent. When the desired outcome is determined, then an event must be caused to the TP implementation to remove recovery records related to the transaction such that it can finally be forgotten by the OSI TP Service Provider. In the terms of the main text and state tables for the OSI TP Protocol, this is equivalent to an internal event which instructs the TPPM to locally terminate the transaction. Such an action would be required in all affected nodes in the same transaction as there would be no means to propagate such an outcome throughout the transaction tree.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex F
(informative)

TPPM transaction states

F.1 TPPM transaction states

At any time, a TPPM is either not part of a transaction tree or is in one of the following states:

— ACTIVE state;

— READY state;

— DECIDED (commit) state;

— DECIDED (rollback) state.

The root TPPM of a transaction tree will never be in the READY state. In the following subclauses, these states are defined by listing all legal state transitions (with reference numbers) and the conditions under which they occur. Figure 1 gives a complete state transition diagram, and labels the arcs with the applicable state transition reference numbers.

The transaction state may change during the course of an action sequence according to the following rules. Should a node crash occur, the transaction state is derived from the existence of a log record.

F.1.1 State transitions of a root TPPM

(1) A TPPM which is not part of a transaction tree becomes the root TPPM and enters the ACTIVE state when the TPPM receives

— a TP-BEGIN-DIALOGUE request with the Chained Transactions functional unit selected; or

— a TP-BEGIN-DIALOGUE request with the Unchained Transactions functional unit selected and the Begin-Transaction parameter set to "true"; or

— a TP-BEGIN-TRANSACTION request.

This occurs in the "Initiating a transaction branch" procedure (see 11.5.4).

(2) A root TPPM makes the transition from the ACTIVE state to the DECIDED (commit) state when the *last ready has been received* and the log-commit record is written.

This occurs in the "Making commitment decision" procedure (see 11.5.7).

(3) A root TPPM makes the transition from the ACTIVE state to the DECIDED (rollback) state when the TPPM issues or receives a rollback-initiating service primitive.

This occurs in the "Initiating rollback at TPPM" procedure (see 11.5.5).

(4) A root TPPM in the ACTIVE state with only one subordinate ceases to be part of the transaction tree when the TPPM receives an AF-BEGIN-DIALOGUE (rejected) confirm or AF-BID (rejected) confirm on the dialogue with that subordinate, or receives an AF-ABORT (user/provider, dataRI) from its sole subordinate before receiving a TP-COMMIT request.

This occurs in the AF-BEGIN-DIALOGUE (rejected, dataRI) confirm (see 11.3.5), SAF-ASSOCIATION-LOST indication" procedure (see 11.3.9), AF-BEGIN-DIALOGUE (rejected(user), rollbackRI) confirm (see 11.3.6), and AF-ABORT (provider, dataRI) indication (see 11.3.20) procedures.

(5) A root TPPM makes the transition from the DECIDED (commit) state (or DECIDED (rollback) state) to the ACTIVE

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

state when the TPPM issues a TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication, respectively) provided at least one subordinate *dialogue is chaining*.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) for the case of rollback.

(6) A root TPPM that is in the DECIDED (commit) state (or DECIDED (rollback) state) ceases to be part of the transaction tree when the TPPM issues a TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication) provided that no subordinate *dialogue is chaining*.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and for the "Initiating transaction after rollback" procedure (see 11.5.6) in the case of rollback.

F.1.2 State transitions of an intermediate TPPM

(7) An intermediate TPPM makes the transition from the ACTIVE state to the READY state when the *last ready has been received* and the TPPM has written the log-ready record.

This occurs in the "Entering READY state" procedure (see 11.5.2).

(8) An intermediate TPPM makes the transition from the ACTIVE state to the DECIDED (rollback) state when the TPPM issues or receives a rollback-initiating service primitive.

This occurs in the "Initiating rollback at TPPM" procedure (see 11.5.5).

(9) An intermediate TPPM makes the transition from the READY state to the DECIDED (commit) state when the TPPM receives *a commit indication* from the superior.

This occurs in the "Receiving commit order" procedure (see 11.5.8).

(10) An intermediate TPPM makes the transition from the READY state to the DECIDED (rollback) state when the TPPM receives *a rollback indication* from the superior.

This occurs in the "Initiating rollback at TPPM" procedure (see 11.5.5).

(11) An intermediate TPPM makes the transition from the DECIDED (commit) state (or DECIDED (rollback) state) to the ACTIVE state when the TPPM issues TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication, respectively) provided that the superior and at least one subordinate *dialogue is chaining*.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) for the case of rollback.

(12) An intermediate TPPM that is in the DECIDED (commit) state (or DECIDED (rollback) state, respectively) ceases to be part of the transaction tree when the TPPM issues TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication, respectively) provided that no *dialogue is chaining*.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) in the case of rollback.

(13) An intermediate TPPM in the DECIDED (commit) state (or DECIDED (rollback) state, respectively) becomes a leaf TPPM in the ACTIVE state when the TPPM issues TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication, respectively) provided that the superior *dialogue is chaining* and no subordinate *dialogue is chaining*.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) in the case of rollback.

(14) An intermediate TPPM in the DECIDED (commit) state (or DECIDED (rollback) state, respectively) becomes a root TPPM in the ACTIVE state when the TPPM issues TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication, respectively) provided that the superior *dialogue is not chaining* and at least one subordinate

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

dialogue is chaining.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) in the case of rollback.

(15) An intermediate TPPM makes the transition from the DECIDED (commit) state to the DECIDED (rollback) state when the TPPM issues a TP-COMMIT-COMPLETE indication followed by a TP-ROLLBACK indication.

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) in the case of rollback.

F.1.3 State transitions of a leaf TPPM

(16) A TPPM which is not part of a transaction tree becomes a leaf TPPM and enters the ACTIVE state when the TPPM receives a C-BEGIN indication.

This occurs in the "C-BEGIN indication" procedure (see 11.3.37).

(17) A leaf TPPM makes the transition from the ACTIVE state to the READY state when the *last ready has been received* and the TPPM has written the log-ready record.

This occurs in the "Entering READY state" procedure (see 11.5.2).

(18) A leaf TPPM which is in the ACTIVE state ceases to be part of the transaction tree when the TPPM receives a TP-BEGIN-DIALOGUE (rejected) response, or when there is a dialogue establishment indication outstanding and an AF-ABORT (user/provider, rollbackRI) indication is received.

This occurs in the TP-BEGIN-DIALOGUE response" procedure (see 11.3.3) and in the AF-ABORT (user/provider, rollbackRI) indication or AF-ABORT-AND-HEURISTIC-REPORT (rollbackRI) indication procedure (see 11.3.55).

(19) A leaf TPPM makes the transition from the ACTIVE state to the DECIDED (rollback) state when the TPPM issues or receives a rollback-initiating service primitive.

This occurs in the "Initiating rollback at TPPM" procedure (see 11.5.5).

(20) A leaf TPPM makes the transition from the READY state to the DECIDED (commit) state when the TPPM receives a *commit indication* from the superior.

This occurs in the "Receiving commit order" procedure (see 11.5.8).

(21) A leaf TPPM makes the transition from the READY state to the DECIDED (rollback) state when the TPPM receives a *rollback indication* from its superior.

This occurs in the "Initiating rollback at TPPM" procedure (see 11.5.5).

(22) A leaf TPPM makes the transition from the DECIDED (commit) state (or DECIDED (rollback) state, respectively) to the ACTIVE state when the TPPM issues TP-COMMIT-COMPLETE ind (or TP-ROLLBACK-COMPLETE indication, respectively) provided that the superior *dialogue is chaining*

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) for the case of rollback.

(23) A leaf TPPM that is in the DECIDED (commit) state (or DECIDED (rollback) state, respectively) ceases to be part of the transaction tree when the TPPM issues a TP-COMMIT-COMPLETE indication (or TP-ROLLBACK-COMPLETE indication, respectively) provided that the superior *dialogue is not chaining available for the next transaction.*

This occurs in the "Completing commitment" procedure (see 11.5.1) for the case of commitment and in the "Initiating transaction after rollback" procedure (see 11.5.6) for the case of rollback.

A leaf TPPM that is in the DECIDED (rollback) state also ceases to be part of the transaction when the TPPM receives a

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

TP-BEGIN-DIALOGUE (rejected) response.

This occurs in the "TP-BEGIN-DIALOGUE response" procedure (see 11.3.3)."

(24) A leaf TPPM which is in the ACTIVE state becomes an intermediate TPPM in the ACTIVE state when the TPPM

— issues an AF-BEGIN-DIALOGUE request with selection of the Chained Transactions functional unit; or

— issues an AF-BEGIN-DIALOGUE request with selection of the Unchained Transactions functional unit and the Begin-Transaction parameter set to "commitment" or

— receives a TP-BEGIN-TRANSACTION request.

This occurs in the "Initiating a transaction branch" procedure (see 11.5.4).

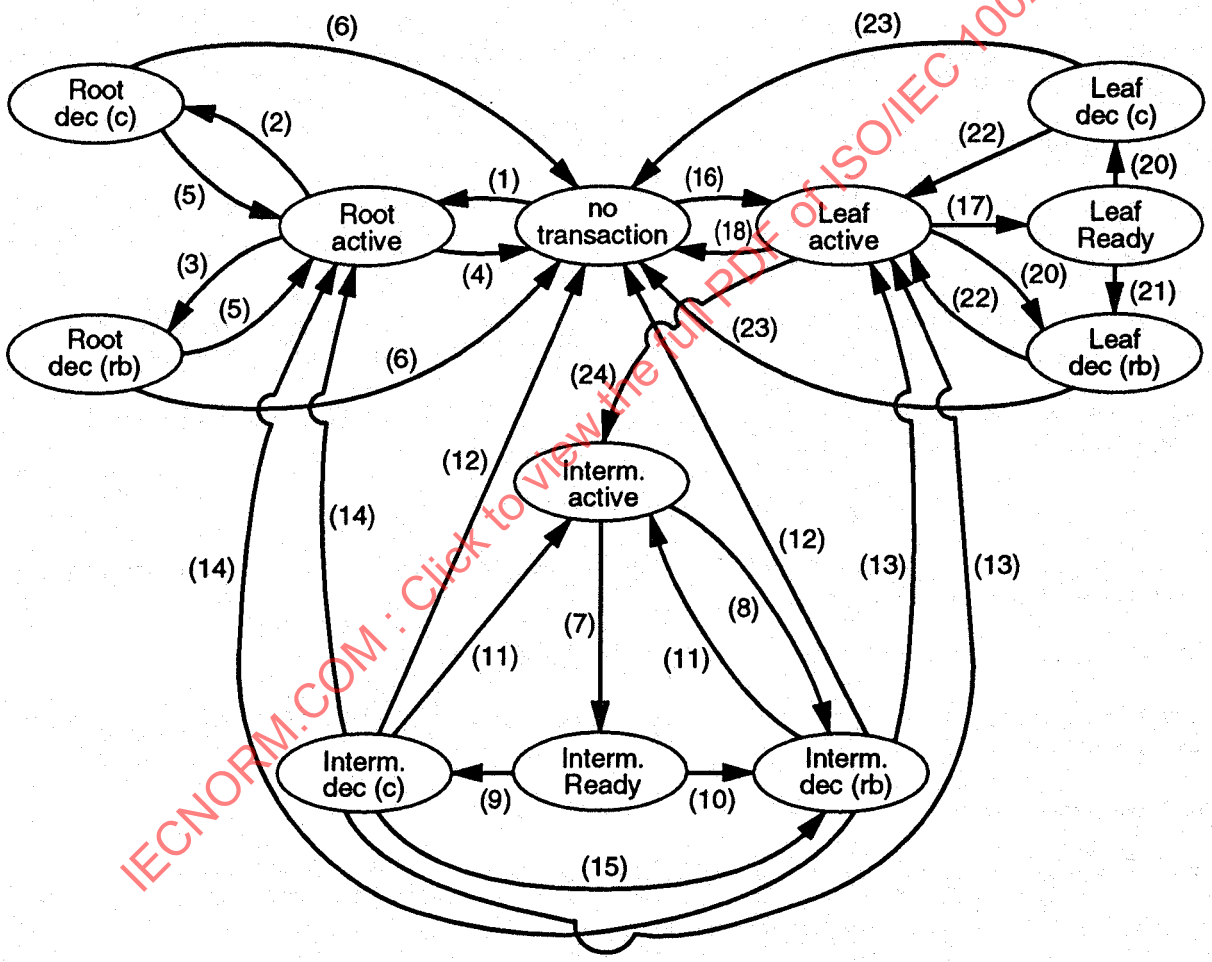


Figure F.1 — State Transition Diagram

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex G
(informative)

Estelle description of the TP protocol

G.1 Introduction

This annex contains an Estelle specification of the OSI TP protocol. It defines the behavior of OSI TP implementations in reaction to OSI TP service primitives, to OSI TP pdus, and to CCR, ACSE, and Presentation service primitives.

The overall structure of the Estelle specification is very close to the structure of the protocol state tables descibed in Annex A of this Standard. The states, variables, and events where to a very large extent derived from Annex A. The only important differences are related to the existence of a MACF state machine in the Estelle specification and the specification of a site channel manager Estelle module, which are both described in other terms in Annex A. Where differences exist with mechanisms used in Annex A, they are noted in relevant sections of the remainder of this introduction.

Figure G.1 shows the structure of the Estelle specification. Every site running OSI TP contains one Channel Protocol Machine (CPM) and an equal number of TP Service User Invocations (TPSUIs) and TP Protocol Machines (TPPMs). In addition, an OSI TP site maintains system logs for the purpose of heuristic reporting and to support recovery in the case of communication failure or node crash.

A TPPM module is created either as a result of a superior dialogue establishment indication or following the creation of a new TPSUI. In the case of a node crash, TPPMs that were engaged in transactions (as detrmined by the system logs) are restarted. The CPM module is created upon initialization of the site. All TPPMs are connected to the site CPM, to prepare for potential recovery.

G.2 Structure of the TPPM

The TPPM consists of multiple dialogue modules nested in an outer module implementing the MACF. Dialogue modules specify the behavior of a TPPM on a specific dialogue or transaction branch, while the outer part of the MACF module specifies the behavior of the TPPM as a node, i.e., coordinate the actions of all dialogues. Two types of dialogue modules are considered: superior dialogues and subordinate dialogues.

In general, when individual dialogue modules receive PDUs and service indications and confirms, they take all actions related to that particular dialogue or transaction branch and, if node level actions are needed as a result, set some exported variables seen by the MACF. The MACF then carries node level actions and may send synchronization messages as Estelle interactions destined to relevant dialogue modules.

Conversely, when the TPSUI issues service primitives, the MACF takes node level actions, if any, and forwards the primitive to the corresponding dialogue or dialogues. The MACF module checks node level service sequencing rules, while individual dialogues check dialogue level service and protocol rules.

Estelle parent/child priority guarantees that nothing is going to happen on any dialogue as long as node level actions are pending. Synchronization messages sent by the MACF to individual transaction branches may be queued behind other messages and will only be processed by these modules once all previous primitives and synchronization messages have been processed. These two properties lead to the same external behavior as action sequences as implemented in Annex A.

During recovery, the MACF acts as an interface between transaction branch modules and the CPM. The MACF module is responsible for locating the transaction branch corresponding to recovery primitives and for dealing with situations where recovery is initiated by the superior and subordinate sides of a same dialogue. Individual modules are therfore guaranteed to have at most one channel attached.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

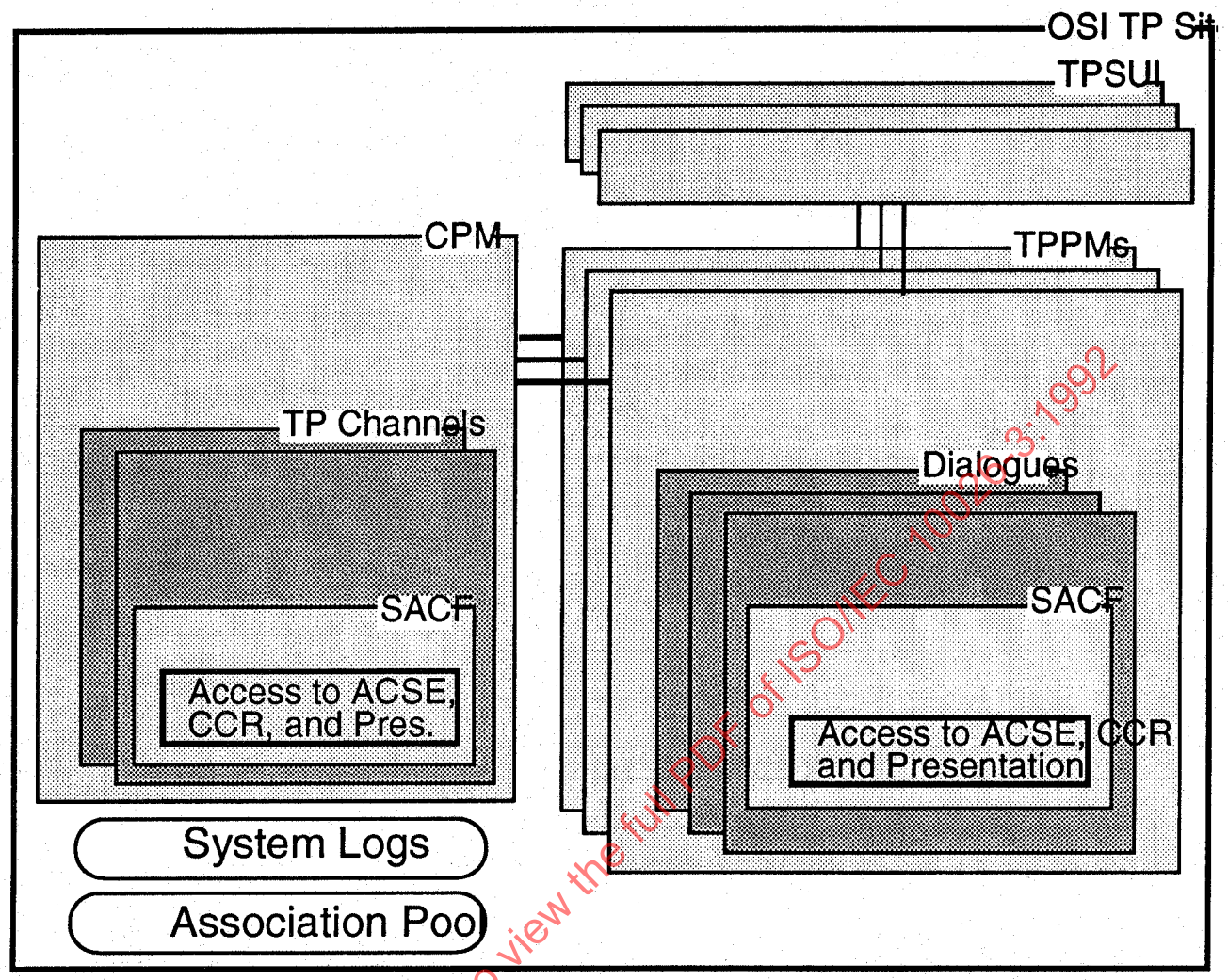


Figure G.1 — Architecture of an OSI TP site

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

G.3 Structure of the CPM

The CPM machine consists of a collection of channel modules nested in an outer module. Individual channel machines are responsible for actions related to specific TP channels, while the outer module acts as a channel manager responsible for establishing and terminating channels and for allocating channels to individual transaction branches.

Inner channel modules have access to ACSE, CCR, and Presentation in the same way as dialogue modules; i.e., through a nested SACF module. The role of channel modules is essentially to establish a TP channel with a superior or subordinate node and position the tokens correctly. When they are associated with a transaction branch, they simply forward recovery primitives to and from the transaction branch. The initial task of locating a TPPM when recovery is initiated by a distant TPPM is carried out by the outer CPM module, using system logs.

G.4 SACF Modules

The SACF module implements the AF Service and the TP-ASE function as defined in this standard. A SACF module is used to support a TP dialogue or channel and manages a single association.

The associations used to support TP dialogues and TP channels are maintained in a pool managed by every site. The implementation of the pool by individual sites is a local issue, not specified in this standard. The Estelle specification included in this annex makes use of Estelle primitive functions to model the basic functionnality expected from such a pool. The

association related to a SACF is obtained from the association pool following either the receipt of a TP channel or dialogue indication or the request for a new dialogue or channel by a local TPSUI or TPPM.

The interface between the SACF and ACSE, CCR, and Presentation is modelled in this annex by an Estelle module with an unspecified module body. The module is assumed to provide SACF modules with means to access a specific association, for the purpose of sending or receiving service primitives. A particularly important assumption made on this module is that it does not violate Presentation service rules with respect to Session purging mechanisms.

G.5 File Structure of the OSI-TP Estelle Specification

The Estelle specification of the OSI TP protocol which follows this introduction is organized as follows:

a) specification shell and site module;

b) tppm macf module;

c) dialogue module interface with the macf;

d) subordinate dialogue module body;

e) superior dialogue module body;

f) common dialogue behavior;

g) action to delimit superior dialogues;

h) cpm module;

i) TP channel module;

j) SACF module;

k) type definitions;

l) Estelle primitive functions;

m) Estelle channels.

These files are connected by Estelle *include directives* to reproduce the structure shown in Figure 1.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
specification OSI_10026_3;

  default individual queue;
  timescale second;

  module site systemactivity;
  end;

  body site_b for site;

#include 'types.e'
#include 'channels.e'
#include 'comfunc.e'


{*****************************************************************************}
{*                                                                           *}
{*  The site module models an open system running ISO 10026.3.               *}
{*                                                                           *}
{*  A site is composed of one CPM module (channel protocol machine) and an   *}
{*  equal number of TPSUIs (TP service user invocations) and TPPMs (TP       *}
{*  protocol machines).  In addition, a pool of associations is managed by   *}
{*  the site and is accessible to the CPM and all the TPPM machines running  *}
{*  on the site.                                                             *}
{*                                                                           *}
{*  The CPM machine is responsible for handling channel establishment and    *}
{*  behavior in support of TP recovery.                                      *}
{*                                                                           *}
{*  Each TPPM machine implements the TP protocol to offer the TP service to  *}
{*  a unique TPSUI.  TPSUIs are not fully described in this specification    *}
{*  since they are outside the scope of this standard.                       *}
{*                                                                           *}
{*  The pool of associations maintained on a site is available to the CPM    *}
{*  and to the TPPM machines.  Only the interface to the pool is described   *}
{*  since the specifics of its implementation are a local issue.             *}
{*                                                                           *}
{*  In addition to managing the pool of associations, the site module is     *}
{*  responsible for the creation/destruction of the CPM, TPPMs, and TPSUIs   *}
{*                                                                           *}
{*****************************************************************************}

    module tpsu_m activity;
      ip
        tppm: TP_Service_aug( user) individual queue;
      export
        tppmIdx: integer;
    end;

    body tpsui_b for tpsu_m; external;

#include 'tppm_m.e'
#include 'cpm_m.e'

   modvar
     tppm: macf_m;
     cpm: cpm_m;
     tpsui: tpsu_m;

   var
     dummyAaid, aaid: atomicActionIdType;
     assocId: assocIdType;
     sub: boolean;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{*                                                                       *}
{*  Initialization of Site.                                              *}
{*                                                                       *}
{*  Create CPM and association pool and restart (TPM,TPSUI) pairs for    *}
{*  every incomplete transaction found in the log (after node crash)     *}
{*                                                                       *}
{*************************************************************************}
  initialize
    var idx: integer;
    begin
      init cpm with cpm_b;
      initAssociationPool;
      for idx := 1 to transNodeRecordLogCount do
        begin
          aaid := transNodeRecordAtIdx( idx);
          init tppm with macf_b( false, nullassocId, true, aaid);
          allocateNewTPPMId( tppm.tppmIdx);
          init tpsui with tpsui_b;
          tpsui.tppmIdx := tppm.tppmIdx;
          connect tppm.tpsu to tpsui.tppm;
          connect tppm.cpm to cpm.tppm[tppm.tppmIdx]
        end;
    end;

{*************************************************************************}
{*                                                                       *}
{*  Management of TPPMs/TPSUIs                                           *}
{*                                                                       *}
{*  A new (TPPM, TPSUI) pair is created when a new dialogue tree is to be *}
{*  be created.  If the new dialogue is initiated locally, then the TPPM  *}
{*  will get an association from the pool or establish a new one; if not, *}
{*  the TPPM is given the id of the related association.                  *}
{*                                                                       *}
{*************************************************************************}
  trans
    provided pendingTPBeginDialogueReq or pendingTPBeginDialogueOrBidInd
      begin
        sub := pendingTPBeginDialogueReq;
        if (sub) then assocId := nullassocId
        else getAssocIdForPendingTPDialOrBidInd( assocId);
        init tppm with macf_b( not sub, assocId,false,dummyAaid);
        allocateNewTPPMId( tppm.tppmIdx);
        init tpsui with tpsui_b;
        tpsui.tppmIdx := tppm.tppmIdx;
        connect tppm.tpsu to tpsui.tppm;
        connect tppm.cpm to cpm.tppm[tppm.tppmIdx]
      end;

    provided exist tppm: macf_m suchthat tppm.dead
      begin
        all tppm: macf_m do
          if tppm.dead then
                begin
                 deallocateTPPMid( tppm.tppmIdx);
                 forone tpsui: tpsu_m suchthat (tpsui.tppmIdx=tppm.tppmIdx)
                   do terminate tpsui;
                 terminate tppm
                end;
      end;
 end;
end.
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{* tppm_m.e file ************************************************************}
{* macf  module.                                                             *}

{*****************************************************************************}
{*                                                                           *}
{*  MACF Module                                                              *}
{*                                                                           *}
{*                                                                           *}
{* The module is created on a site when a dialogue establishment is          *}
{* pending.  If the dialogue is initiated by a distant TPSU (i.e., this      *}
{* is a subordinate node), the identifier of the related association         *}
{* is passed as argument to this module, which will in turn pass it to       *}
{* the dialogue module (see dialogue and sao modules).                       *}
{*                                                                           *}
{*                                                                           *}
{* The module has an interaction point for acess to the TPSU.  The macf      *}
{* receives service primitives from the TPSU, checks TPSP constraints        *}
{* related to more than one dialogue (i.e., those related to the node        *}
{* context rather than to specific dialogue contexts) and forwards these     *}
{* primitives to the appropriate dialogue submodule.  Conversely, it         *}
{* forwards service indications and confirms to the tpsui.                   *}
{* The macf also generates confirms and indications to the TPSUI when        *}
{* they are the result of a condition involving more that one dialogue       *}
{* (e.g., TP_COMMIT_COMPLETE_ind).                                           *}
{*                                                                           *}
{* In addition, the macf module is responsible for all synchronization       *}
{* aspects during commitment and rollback, and for interfacing dialogues     *}
{* and the CPM.                                                              *}
{*                                                                           *}
{* The module may also be restarted after a node crash.  In this case,       *}
{* it is reponsible for creating the appropriate branch modules, based       *}
{* on the node log record for the specified transaction.                     *}
{*                                                                           *}
{*****************************************************************************}

 module macf_m activity(
      superiorDialogue: boolean;  { subordinate node }
      assocId: assocIdType;       { if subordinate, id of association }
      nodeCrash: boolean;         { module created after a node crash ?}
      atomicActId: atomicActionIdType { if module is created after crash }
       );

    ip
      tpsu: TP_Service_aug( provider); { access to the TPSUI }
      cpm: CPM_TPPM_Service( tppm);    { access to the CPM }

    export
      dead: boolean;   { Node ceases to exist when this is set }
      tppmIdx: integer; { local identifier of tppm - shared with tpsui }

  end;

 body macf_b for macf_m;

#include 'dial_m.e'

   modvar
     dialogues: array [DIAL_IDX] of dial_m;
     supDial, dial: dial_m;

   ip
     dIp: array [DIAL_IDX] of TP_Service( user); { access to dialogues }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    dSynchro: array[DIAL_IDX] of synchroMess( macf); {synchro of branches}
    dChanIp: array[DIAL_IDX] of CPM_Service( provider);{dial/cpm interface }

var
  Ncr: boolean;    { dialogue establishment with superior outstanding }
  Nfa: boolean;    { TP-U-ABORT reqs ok during transaction termination }
  Nhd: boolean;    { log damage record exists }
  Nhe: boolean;    { log heuristic record exists }
  Np: boolean;     { prepare indication received from superior }
  Nrn: boolean;    { reject of superior dialogue not allowed }
  Nrpend: boolean;{ rollback next transaction is pending }
  Nt: boolean;     { termination phase of transaction - tree cannot grow }
  Nzp: branchIdType; { identifier of branch to the superior }
  Nzpn: branchIdType; { id of branch to superior for next transaction }
  Nzt: atomicActionIdType; { current transaction identifier }
  Nztn: atomicActionIdType; { transaction identifier of next transaction }
  Nd: boolean; { done received }

  hasSup: boolean;  { subordinate node in a transaction tree ? }
  boundFinal: boolean; { data is in final state }
  boundReady: boolean; { data is in ready state }
  boundInitial: boolean; { data is in initial state }

  noPars: emptyParameterList;
  nullRecoveryCntxtHndl : recoveryContextHandleType;
  nullBranchId: branchIdType;
  supDialIdx: DIAL_IDX;


 state
  NOT_IN_TRANSACTION,                {* node not involved in transaction *}
  ACTIVE,                                  {* node in transaction tree *}
  WAIT_FOR_RDY,           {* commitment phase 1 in progress in subtree *}
  READY,                             {* commitment phase 1 completed *}
  DECIDED_COMMIT,          { commitment phase 2 in progress in subtree *}
  DECIDED_ROLLBACK,       {* rollbak in progress in subordinate subtree *}
  ROLLBACK_REPORTED,       {* waiting for superior,if any, to confirm *}
                                  { rollback and/or send begin ind *}
  COMMIT_COMPLETE,               {* commitment complete in the subtree *}
  ROLLBACK_COMPLETE,      {* rollback complete -  ready for next trans *}
  RESTART;                          {* initial state after node crash *}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{*                                                                       *}
{*  MACF initialization                                                  *}
{*                                                                       *}
{*************************************************************************}

initialize
  provided not nodeCrash
    to NOT_IN_TRANSACTION
      begin
       dead := false;
       Ncr := false;
       Nd := false;
       Nfa := false;
       Nhd := false;
       Nhe := false;
       hasSup := false;
       Np := false;
       Nrn := false;
       Nrpend := false;
       Nt := false;
       if (superiorDialogue) then
        begin
          supDialIdx := allocateDialIdx;
          init dialogues[supDialIdx] with dial_sup_b(assocId,false,Nzt,Nzp);
          supDial := dialogues[supDialIdx];
          supDial.dialIdx := supDialIdx;
          connect supDial.tpsu to dIp[supDialIdx];
          connect supDial.cpm to dChanIp[supDialIdx];
          connect supDial.synchro to dSynchro[supDialIdx];
        end
     end;

initialize { initialization after a node crash }
  provided nodeCrash
    to RESTART
      var brId: branchIdType;
          subs: integer;
          dial: DIAL_IDX;
          sup,sub : boolean;
      begin
       hasSup := false;
       sub := false;
       {+ REBUILDTPPM +}
       Nzt := atomicActId;
       Nzp := branchToSupInTransNodeLog( Nzt);
       Nt := true;
       Nhd := existLogDamage(Nzt);
       Nhe := existLogHeuristic(Nzt);
       {+ create superior dialogue +}
       if (notNullBranchId(Nzp)) then
         begin
          sup := true;
          supDialIdx := allocateDialIdx;
          init dialogues[supDialIdx]
                          with dial_sup_b(nullAssocId,true,Nzt,Nzp);
          supDial := dialogues[supDialIdx];
          supDial.dialIdx := supDialIdx;
          connect supDial.tpsu to dIp[supDialIdx];
          connect supDial.cpm to dChanIp[supDialIdx];
          connect supDial.synchro to dSynchro[supDialIdx];
         end
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        else
          supDialIdx := 0;
        {+ create subordinate dialogues +}
        for subs := 1 to logSubBranchCount( Nzt) do
          begin
           sub := true;
           dial := allocateDialIdx;
           brId := logSubBranchIdAtIdx(Nzt,subs);
           init dialogues[dial] with dial_sub_b(nullAssocId,true,Nzt,brId);
           dialogues[dial].dialIdx := dial;
           connect dIp[dial] to dialogues[dial].tpsu;
           connect dChanIp[dial] to dialogues[dial].cpm;
           connect dSynchro[dial] to dialogues[dial].synchro;
           dialogues[dial].Dzb := brId;
           dialogues[dial].Nzt := Nzt;
         end;
      end;

 trans
   from RESTART
     provided (recordedTransNodeState(Nzt)=logInitial)
       to READY
         begin
           all d: dial_m do
             output dSynchro[d.dialIdx].restartInReady;
         end;

     provided (recordedTransNodeState(Nzt)=logFinal)
       to DECIDED_COMMIT
         begin
           output tpsu.TP_COMMIT_ind(noPars);
           all d: dial_m do
              output dSynchro[d.dialIdx].restartInCommit;
         end;

{*************************************************************************}
{*                                                                       *}
{* Grafting a transaction/dialogue tree                                  *}
{*                                                                       *}
{*************************************************************************}

{******* DIALOGUE ESTABLISHMENT * SUPERIOR SIDE ************************}

 trans {* subordinate dialogue *}
   when tpsu.TP_BEGIN_DIALOGUE_req( param, dial)
     provided ( not Ncr and not beginTransactionReq( param))
       from ACTIVE, NOT_IN_TRANSACTION
         to SAME
           var dial: DIAL_IDX;
           begin
            {+ INITDIASB +}
            dial := allocateDialIdx;
            init dialogues[dial] with dial_sub_b(nullAssocId,false,Nzt,Nzp);
            dialogues[dial].dialIdx := dial;
            connect dIp[dial] to dialogues[dial].tpsu;
            connect dChanIp[dial] to dialogues[dial].cpm;
            connect dSynchro[dial] to dialogues[dial].synchro;
            Nrn := true;
            {+ ABDrq +}
            output dIp[dial].TP_BEGIN_DIALOGUE_req( param);
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    provided ( not Ncr and beginTransactionReq( param) and not Nt)
      from ACTIVE, NOT_IN_TRANSACTION
        to ACTIVE
          var dial: DIAL_IDX;
          begin
            {+ INITDIASB +}
            dial := allocateDialIdx;
            init dialogues[dial] with dial_sub_b(nullAssocId,false,Nzt,Nzp);
            dialogues[dial].dialIdx := dial;
            connect dIp[dial] to dialogues[dial].tpsu;
            connect dChanIp[dial] to dialogues[dial].cpm;
            connect dSynchro[dial] to dialogues[dial].synchro;
            Nrn := true;
            {+ ADDBRSB +}
            if (not hasSup) then
              begin
                Nzt := newAtomicActionId;
                Nzp := nullBranchId;
                createTransNodeRecord(Nztn,Nzpn,false,nullRecoveryCntxtHndl);
              end;
            if (supDialIdx<>0) then supDial.inTrans := true;
            dialogues[dial].Dzb := newBranchId( Nzt);
            dialogues[dial].Nzt := Nzt;
            output dIp[dial].TP_BEGIN_DIALOGUE_req( param);
          end;

{******* DIALOGUE ESTABLISHMENT * SUBORDINATE ****************************}

trans
  provided ((supDialIdx<>0) and not supDial.D1)
    when dIp[supDialIdx].TP_BEGIN_DIALOGUE_ind( param)
      begin
        {+ INITDIASP +}
        Ncr := supDial.Dcr;
        {+ TBDi +}
        output tpsu.TP_BEGIN_DIALOGUE_ind( param, supDialIdx);
      end;

trans
  provided ((supDialIdx<>0) and supDial.D1)
    when dIp[supDialIdx].TP_BEGIN_DIALOGUE_ind( param)
      to ACTIVE
        begin
          {+ INITDIASP +}
          Ncr := supDial.Dcr;
          {+ TBDi +}
          output tpsu.TP_BEGIN_DIALOGUE_ind( param, supDialIdx);
          {+ ADDBRSP +}
          supDial.inTrans := true;
          hasSup := true;
          Nzt := supDial.Nzt;
          Nzp := supDial.Dzb;
          createTransNodeRecord(Nzt,Nzp,supDial.Dhb,supDial.Dzhb);
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{****** UNCHAINED FUNCTIONAL UNIT - RAISING COMITMENT LEVEL  ************}

trans
 from ACTIVE, NOT_IN_TRANSACTION
   to ACTIVE
     provided (supDialIdx<>0)
       when dIp[supDialIdx].TP_BEGIN_TRANSACTION_ind( param)
         begin
           output tpsu.TP_BEGIN_TRANSACTION_ind( param, supDialIdx);
           {+ ADDBRSP +}
           hasSup := true;
           supDial.inTrans := true;
           Nzt := supDial.Nzt;
           Nzp := supDial.Dzb;
           createTransNodeRecord(Nzt,Nzp,supDial.Dhb,supDial.Dzhb);
         end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    to ACTIVE
      when tpsu.TP_BEGIN_TRANSACTION_req( param, dial)
        provided not Nt
          begin
            {+ ADDBRSB +}
            if (not hasSup) then
              begin
                Nzt := newAtomicActionId;
                Nzp := nullBranchId;
                createTransNodeRecord(Nzt,Nzp,supDial.Dhb,supDial.Dzhb);
              end;
            if (supDialIdx<>0) then supDial.inTrans := true;
            dialogues[dial].Dzb := newBranchId( Nzt);
            dialogues[dial].Nzt := Nzt;
            output dIp[dial].TP_BEGIN_TRANSACTION_req( param);
          end;

{*************************************************************************}
{*                                                                       *}
{* Dialogue Specific TP Services                                         *}
{*                                                                       *}
{* MACF simply checks node service squencing rules and forwards to proper *}
{* dialogue or to TPSU.                                                  *}
{*                                                                       *}
{*************************************************************************}

trans
  when tpsu.TP_BEGIN_DIALOGUE_rsp( param, dial)
    provided ((param.result=accepted) or not Nrn)
      begin
        Ncr := false;
        Nrn := true;
        output dIp[ dial].TP_BEGIN_DIALOGUE_rsp( param);
      end;

trans
  any dial: DIAL_IDX do
    when dIp[dial].TP_BEGIN_DIALOGUE_cnf( param)
      begin
        output tpsu.TP_BEGIN_DIALOGUE_cnf( param, dial);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when tpsu.TP_END_DIALOGUE_req( param, dial)
     begin
       Nrn := true;
       Ncr := false;
       output dIp[dial].TP_END_DIALOGUE_req( param)
     end;

trans
  any dial: DIAL_IDX do
    when dIp[dial].TP_END_DIALOGUE_ind( param)
       begin
         output tpsu.TP_END_DIALOGUE_ind( param, dial);
       end;

trans
  when tpsu.TP_END_DIALOGUE_rsp( param, dial)
    begin
      Nrn := true;
      Ncr := false;
      output dIp[dial].TP_END_DIALOGUE_rsp( param);
    end;

trans
  any dial: DIAL_IDX do
    when dIp[dial].TP_END_DIALOGUE_cnf( param)
       begin
         output tpsu.TP_END_DIALOGUE_cnf( param, dial);
       end;

trans
  when tpsu.TP_DEFERRED_END_DIALOGUE_req( param, dial)
    from ACTIVE, NOT_IN_TRANSACTION
       begin
         Nrn := true;
         Ncr := false;
         output dIp[ dial].TP_DEFERRED_END_DIALOGUE_req( param);
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_DEFERRED_END_DIALOGUE_ind( param)
       begin
         output tpsu.TP_DEFERRED_END_DIALOGUE_ind( param, idx);
       end;

trans
  when tpsu.TP_DEFERRED_GC_req( param, dial)
    from ACTIVE, NOT_IN_TRANSACTION
      begin
        Nrn := true;
        Ncr := false;
        output dIp[ dial].TP_DEFERRED_GC_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_DEFERRED_GC_ind( param)
      begin
        output tpsu.TP_DEFERRED_GC_ind( param, idx);
      end;

trans
  when tpsu.TP_U_ABORT_req( param, dial)
    provided (not Nt or Nfa)
      begin
        Nrn := true;
        Ncr := false;
        output dIp[ dial].TP_U_ABORT_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_U_ABORT_ind( param)
      begin
         output tpsu.TP_U_ABORT_ind( param, idx);
         Nfa := true;
         Nd := false;
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_P_ABORT_ind( param)
      begin
       output tpsu.TP_P_ABORT_ind( param, idx);
        Nfa := true;
        Nd := false;
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_U_ERROR_req( param, dial)
      begin
        Nrn := true;
        Ncr := false;
        if (dialogues[dial].Ds) then Nrn := true;
        output dIp[ dial].TP_U_ERROR_req( param);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_U_ERROR_ind( param)
      begin
        output tpsu.TP_U_ERROR_ind( param, idx);
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_DATA_req( param, dial)
      begin
        Nrn := true;
        Ncr := false;
        output dIp[ dial].TP_DATA_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_DATA_ind( param)
      begin
        output tpsu.TP_DATA_ind( param, idx);
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_HANDSHAKE_req( param, dial)
      begin
        Nrn := true;
        Ncr := false;
        output dIp[ dial].TP_HANDSHAKE_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_HANDSHAKE_ind( param)
      begin
        output tpsu.TP_HANDSHAKE_ind( param, idx);
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_HANDSHAKE_rsp( param, dial)
      begin
        Nrn := true;
        Ncr := false;
        if (dialogues[dial].Ds) then Nrn := true;
        output dIp[ dial].TP_HANDSHAKE_rsp( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_HANDSHAKE_cnf( param)
      begin
        output tpsu.TP_HANDSHAKE_cnf( param, idx);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_HANDSHAKE_GC_req( param, dial)
      begin
        if (dialogues[dial].Ds) then Nrn := true;
        output dIp[ dial].TP_HANDSHAKE_GC_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_HANDSHAKE_GC_ind( param)
      begin
        output tpsu.TP_HANDSHAKE_GC_ind( param, idx);
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_HANDSHAKE_GC_rsp( param, dial)
      begin
        if (dialogues[dial].Ds) then Nrn := true;
        output dIp[ dial].TP_HANDSHAKE_GC_rsp( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_HANDSHAKE_GC_cnf( param)
      begin
        output tpsu.TP_HANDSHAKE_GC_cnf( param, idx);
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_GRANT_CONTROL_req( param, dial)
      begin
        if (dialogues[dial].Ds) then Nrn := true;
        output dIp[ dial].TP_GRANT_CONTROL_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_GRANT_CONTROL_ind( param)
      begin
        output tpsu.TP_GRANT_CONTROL_ind( param, idx);
      end;

trans
  from ACTIVE, NOT_IN_TRANSACTION
    when tpsu.TP_REQUEST_CONTROL_req( param, dial)
      begin
        if (dialogues[dial].Ds) then Nrn := true;
        output dIp[ dial].TP_REQUEST_CONTROL_req( param);
      end;

trans
  any idx: DIAL_IDX do
    when dIp[idx].TP_REQUEST_CONTROL_ind( param)
      begin
        output tpsu.TP_REQUEST_CONTROL_ind( param, idx);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{*                                                                       *}
{* Prunning a transaction/dialogue tree                                  *}
{*                                                                       *}
{* When a dialogue ceases to exist and after it has done the necessary   *}
{* clean up operations (such as release association to the pool), it sets *}
{* its "dead" flag and is released by the transitions below.             *}
{*                                                                       *}
{* A dialogue with coordination level "commitment" can be removed in the *}
{* ACTIVE state only if it is a superior dialogue and there are no other *}
{* dialogues with coordination level "commitment".  Otherwise, the       *}
{* transaction will have to go through the termination phase.            *}
{*                                                                       *}
{*************************************************************************}

trans
  from ACTIVE, NOT_IN_TRANSACTION
    provided exist d: dial_m suchthat d.dead
      begin
        all d: dial_m do
          if (d.dead) then
            begin
              terminate d;
              if (d.dialIdx = supDialIdx) then
                begin
                  supDialIdx := 0;
                  hasSup := false;
                end;
            end;
      end;

  from ACTIVE
    to NOT_IN_TRANSACTION
      provided (not exist d:dial_m suchthat d.D1)
        begin
        end;

  from NOT_IN_TRANSACTION
    provided (not exist d: dial_m suchthat true)
      begin
        dead := true;
      end;


{*************************************************************************}
{*                                                                       *}
{*  HEURISTICS                                                           *}
{*                                                                       *}
{*************************************************************************}

trans {* heuristic report from subordinate subtree *}
  any idx: DIAL_IDX do
    when dIp[idx].TP_HEURISTIC_REPORT_ind( param)
      begin
        output tpsu.TP_HEURISTIC_REPORT_ind( param, idx);
        if (not Nhd or (param.heuristicReport=heuristicMix)) then
           logDamage(Nzt,param.heuristicReport);
        Nhd := true;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {* local heuristic decision *}
  from READY
    provided heuristicDecision
      begin
        if (not Nhe) then
          logHeuristic(Nzt,heuristicReport)
        else if (recordedHeuristic(Nzt)<>heuristicReport) then
          logHeuristic(Nzt,heuristicMix);
        Nhe := true;
      end;

trans {* compensating for heuristic damage *}
  from WAIT_FOR_RDY, READY, DECIDED_COMMIT, DECIDED_ROLLBACK
    provided (Nhd and heuristicDecisionComp)
      begin
        deleteLogDamage(Nzt);
        if (Nhe) then deleteLogHeuristic(Nzt);
        Nhd := false;
        Nhe := false;
      end;

{*************************************************************************}
{*                                                                       *}
{*  COMMITMENT - PHASE 1                                                 *}
{*                                                                       *}
{*************************************************************************}

trans
  from ACTIVE
    when dIp[supDialIdx].TP_PREPARE_ind( param)
      provided (not hasSup)
        begin
          Np := true;
          output tpsu.TP_PREPARE_ind( param, supDialIdx);
        end;

    when tpsu.TP_PREPARE_req( param, dial)
      provided (not hasSup or Np)
        begin
          output dIp[dial].TP_PREPARE_req( param);
        end;

    when tpsu.TP_COMMIT_req
      provided (not hasSup or Np)
        to WAIT_FOR_RDY
          begin
            all d: dial_m do
              if (d.Dl) then
                output dIp[d.dialIdx].TP_COMMIT_req( noPars);
            Nt := true;
          end;

trans
  from WAIT_FOR_RDY
    to READY
      provided (boundReady and
        not exist d: dial_m suchthat (d.Dl and not d.Ds and not d.DrdyRcvd))
       begin
         all d: dial_m do
           if (d.Dl) then output dSynchro[d.dialIdx].sendReady;
         recordLogReady(Nzt);
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  from READY
    provided (not hasSup or supDial.DcmtIndRcvd)
      to DECIDED_COMMIT
         begin
           Nd := false;
           setDataFinalState;
           recordLogCommit(Nzt);
           output tpsu.TP_COMMIT_ind( noPars);
           if (hasSup and supDial.Dbr) then
             begin { Nztn and Dzbn received from sup dialogue }
               Nzpn := supDial.Dzbn;
               Nztn := supDial.Nztn;
               createTransNodeRecord(Nztn,Nzpn,supDial.Dhb,supDial.Dzhb);
             end
           else
             begin
               Nztn := newAtomicActionId;
               Nzpn := nullBranchId;
               createTransNodeRecord(Nztn,Nzpn,false,nullRecoveryCntxtHndl);
             end;
           if (Nhe and (recordedHeuristic(Nzt)<>heuristicFinal)) then
             begin
               logDamage(Nzt,heuristicMix);
               Nhe := false;
             end;
           all d: dial_m do
             if (not d.Ds and d.Dl) then
               begin
                 d.Nztn := Nztn;
                 d.Dzbn := newBranchId( Nztn);
                 output dSynchro[d.dialIdx].sendCommitOrder;
               end;
         end;

{*************************************************************************}
{*                                                                       *}
{* COMMITMENT - PHASE 2                                                  *}
{*                                                                       *}
{*************************************************************************}
 trans
   when tpsu.TP_DONE_req( param)
     from DECIDED_COMMIT
       provided (not Nd)
         begin
           if (not Nhd or (param.heuristicReport=heuristicMix)) then
             begin
               logDamage(Nzt,param.heuristicReport);
               Nhd := true;
             end;
           Nd := true;
           Nfa := false;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {* propagating rollback-next during commitment *}
  from DECIDED_COMMIT
    provided (not Nrpend and exist d: dial_m suchthat (d.Dl and d.DrollbackNext))
     begin
       all d: dial_m do
         if (d.Dl and not d.Db and d.Dbr) then
            begin
              output dSynchro[d.dialIdx].rPend;
              d.DrollbackNext := false;
            end;
       Nrpend := true;
     end;

trans  {* commitment complete *}
  from DECIDED_COMMIT
    to COMMIT_COMPLETE
      provided (boundFinal and Nd and not
         exist d:dial_m suchthat (d.Dl and not d.Ds and not d.DcmtCnfRcvd))
       var sup, sub: boolean;
       begin
         output tpsu.TP_COMMIT_COMPLETE_ind( noPars);
         deleteTransNodeRecord(Nzt);
         Nfa := false;
         Np := false;
         sup := supDial.Dbr;
         sub := exist d: dial_m suchthat (not d.Ds and d.Dbr);
         if (supDialIdx<>0) then supDial.Nhd := Nhd;
         Nzt := Nztn;
         Nzp := Nzpn;
         Nt := false;
         all d: dial_m do
           if (d.Dl) then
             begin
               d.Nzt := Nzt;
               output dSynchro[d.dialIdx].commitComplete;
               d.DcommitDone := false;
             end;
       end;

trans {* next trans after commit *}
 from COMMIT_COMPLETE
   provided (not Nrpend and (exist d: dial_m suchthat d.dl) and
       (not exist d: dial_m suchthat (d.dl and not d.DcommitDone)))
     to ACTIVE
       begin
       end;

   provided (not Nrpend and not exist d: dial_m suchthat d.dl)
     to NOT_IN_TRANSACTION
       begin
         deleteTransNodeRecord(Nzt); { was created with Nztn and Nzpn }
       end;

   provided (Nrpend and
       not exist d: dial_m suchthat (d.dl and not d.DcommitDone))
     to DECIDED_ROLLBACK
       begin
         Nfa := true;
         Nt := true;
         Nrpend := false;
         output tpsu.TP_ROLLBACK_ind( noPars);
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{*                                                                       *}
{* ROLLBACK                                                              *}
{*                                                                       *}
{*************************************************************************}

{***** TIGERRING-PROPAGATING ROLLBACK ************************************}

 trans  {* rollback trigerred by TPSU *}
   from ACTIVE
     to DECIDED_ROLLBACK
       when tpsu.TP_ROLLBACK_req
         begin
           setDataInitialState;
           Nt := true;
           all d: dial_m do
              if (d.Dl) then
                 output dIp[d.dialIdx].TP_ROLLBACK_req( param);
           Nd := false;
         end;

 trans
   any idx: DIAL_IDX do
     when dIp[idx].TP_ROLLBACK_ind( param)
       begin
         output tpsu.TP_ROLLBACK_ind( param);
       end;

 trans  { propagation of rollback trigerred by sub/sup dialogue or  TPPM }
  from ACTIVE, WAIT_FOR_RDY, READY
    to DECIDED_ROLLBACK
      provided (rollbackByTPPM
            or exist d: dial_m suchthat (d.Dl and d.DrolledBack))
        begin
          if (not exist d:dial_m suchthat (d.Dl and d.DrolledBack)) then
            output tpsu.TP_ROLLBACK_ind( noPars);
          setDataInitialState;
          if (Nhe) then
            if (recordedHeuristic(Nzt)<>heuristicInitial) then
              begin
                logDamage(Nzt,heuristicMix);
                Nhd := true;
                Nhe := false;
              end;
          Nt := true;
          Nd := false;
          Nfa := true;
          all d: dial_m do
            if (not d.Drolledback) then
              output dSynchro[d.dialIdx].rollbackAll
            else
              d.Drolledback := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{***** ROLLBACK COMPLETION and BEGINNING NEXT TRANS *********************}

 trans
  from DECIDED_ROLLBACK
    when tpsu.TP_DONE_req
      provided (not Nd)
       begin
         Nd := true;
         Nfa := false;
       end;

 trans
  from DECIDED_ROLLBACK
    to ROLLBACK_REPORTED
      provided (boundInitial and not
        exist d: dial_m suchthat (d.Dl and not d.Ds and d.DwrollbackCnf))
      begin
        if (hasSup) then
          begin
            output dSynchro[supDial.dialIdx].reportRollback;
            supDial.Nhd := Nhd;
          end;
      end;

 trans
  from ROLLBACK_REPORTED
    to ROLLBACK_COMPLETE
      provided (Nd and
           (not hasSup
            or (not supDial.DwrollbackCnf and not supDial.DwBeginInd)))
        var sup, sub: boolean;
        begin
          output tpsu.TP_ROLLBACK_COMPLETE_ind( noPars);
          deleteTransNodeRecord(Nzt);
          Nfa := false;
          Np := false;
          Nt := false;
          Nrpend := false;
          Nhe := false;
          Nhd := false;
          if (supDial.Dl) then
            begin
              Nzt := supDial.Nzt; { obtained from c_begin_ind }
              Nzp := supDial.Dzb;
              createTransNodeRecord(Nztn,Nzpn,supDial.Dhb,supDial.Dzhb);
            end
          else
            begin
              Nzt := newAtomicActionId;
              Nzp := nullBranchId;
              createTransNodeRecord(Nztn,Nzpn,false,nullRecoveryCntxtHndl);
            end;
          all d: dial_m do
            if (d.Dl) then
              begin
                if (d.Dbr) then d.Nzt := Nzt;
                output dSynchro[d.dialIdx].rollbackComplete;
                d.DrollbackDone := false;
              end;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { nest transaction after rollback }
  from ROLLBACK_COMPLETE
    provided (not exist d: dial_m suchthat d.D1)
      to NOT_IN_TRANSACTION
         begin
           if (supDialIdx<>0) then supDial.inTrans := false;
           hasSup := false;
         end;

    provided ((exist d:dial_m suchthat d.D1) and not
        exist d: dial_m suchthat (d.D1 and not d.DrollbackDone))
      to ACTIVE
        begin
           if (supDialIdx<>0) then
             begin
               supDial.inTrans := true;
               hasSup := supDial.D1;
             end
           else hasSup := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{*                                                                       *}
{*   RECOVERY.                                                           *}
{*                                                                       *}
{* The role of the MACF (TPPM) module during recovery is essentially to  *}
{* locate transaction branches to which recovery primitives coming from a *}
{* channel are addressed and to insure that a maximum of one channel is  *}
{* attached to any given transaction branch.                             *}
{*                                                                       *}
{* - if the branch corresponding to the recovery order is not already    *}
{*   attached, the primitive is forwarded to the corresponding module.   *}
{* - if the branch corresponding to the recovery order is already attached*}
{*   and the module receives a CAF_GIVE_ind, the TPPM detaches the new    *}
{*   channel;                                                            *}
{* - if the branch corresponding to a RECOVER_ind is already attached,   *}
{*   the following procedure is performed:                               *}
{*      -- if this is a recoverCommit and the node is in the COMMIT state,*}
{*         or if this is a recoverREady and the node is in the ready     *}
{*         state, the primitive is not transferred to the branch, and one *}
{*         or both channels are detached after a recover(retryLater)-rsp; *}
{*      -- if this is a recoverReady in the commit state, the new channel *}
{*         is detached after a recover(retryLater)-rsp; the primitive is  *}
{*         not forwarded to the branch;                                  *}
{*      -- if this is a recoverCommit in the ready state, the old channel *}
{*         is detached and the primitive is forwarded to the branch which *}
{*         will go to the decided(commit) state as a result.             *}
{*                                                                       *}
{* In the reverse direction, the TPPM forwards primitives to the CPM.    *}
{*                                                                       *}
{*************************************************************************}

{*** CPM -> Transaction Branch. *****************************************}

 trans
   when cpm.CPM_TO_TPPM_ind
     provided (param.order=cafGiveInd)
       var cdet: cafDetachType;
           recReq: recoveryReqType;
       begin
         forone d: dial_m
            suchthat (sameBranchId(d.Dzb,param) and not d.Dchat) do
               begin
                  output dChanIp[d.dialIdx].CAF_GIVE_ind(param.cafGivePars);
                  d.Dchat := true;
                  d.chanIdx := param.chanIdx;
               end
            otherwise
               begin
                 recReq.order := cafDetachReq;
                 recReq.cafDetachPars.status := not_used;
                 output cpm.TPPM_TO_CPM_req( recReq);
               end
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { recovery initiated by peer in READY state }
  when cpm.CPM_TO_TPPM_ind
    provided (param.order=cafRecoverInd)
      from READY
        var recReq: recoveryReqType;
            detPars: cafDetachType;
            recState: recoveryStateType;
        begin
          forone d: dial_m suchthat (sameBranchId( d.Dzb, param)) do
            dial := d;
          if (not dial.Dchat) then
            begin
              dial.Dchat := true;
              dial.chanIdx := param.chanIdx;
              output dChanIp[dial.dialIdx].
                                  C_RECOVER_ind(param.cafRecoverIndPars)
            end
          else
           begin { retry one of the two channels }
            recReq.order := cRecoverRsp;
            recReq.cRecoverRspPars.recoveryState := retryLater;
            recReq.cRecoverRspPars.atomicActionId := Nzt;
            recReq.cRecoverRspPars.branchId := dial.Dzb;
            recState := param.cafRecoverIndPars.recoveryState;
            if (recState=readyState) then
              begin  { local decision as to which one should be kept }
                if (wantToDetachFirst) then
                  begin
                    recReq.chanIdx := dial.chanIdx;
                    dial.chanIdx := param.chanIdx;
                    detPars.status := free_channel;
                  end
                else
                  begin
                    recReq.chanIdx := param.chanIdx;
                    detPars.status := free_channel;
                  end
              end
            else
              begin { detach current channel-forward primitive to branch }
                recReq.chanIdx := dial.chanIdx;
                dial.chanIdx := param.chanIdx;
                detPars.status := cleanup;
                output dChanIp[dial.dialIdx].
                                  C_RECOVER_ind( param.cafRecoverIndPars);
              end;
              {* retry the selected one, detach it *)
            output cpm.TPPM_TO_CPM_req( recReq);
            recReq.order := cafDetachReq;
            recReq.cafDetachPars := detPars;
            output cpm.TPPM_TO_CPM_req( recReq);
           end;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { reoovery initiated by peer in COMMIT state }
  when cpm.CPM_TO_TPPM_ind
    provided (param.order=cafRecoverInd)
      from DECIDED_COMMIT
        var recReq: recoveryReqType;
            detPars: cafDetachType;
            recState: recoveryStateType;
        begin
          forone d: dial_m suchthat (sameBranchId( d.Dzb, param)) do
            dial := d;
          if (not dial.Dchat) then
            begin
              dial.Dchat := true;
              dial.chanIdx := param.chanIdx;
              output dChanIp[dial.dialIdx].
                                    C_RECOVER_ind(param.cafRecoverIndPars)
            end
          else
           begin { retry one of the two channels }
             recReq.order := cRecoverRsp;
             recReq.cRecoverRspPars.recoveryState := retryLater;
             recReq.cRecoverRspPars.atomicActionId := Nzt;
             recReq.cRecoverRspPars.branchId := dial.Dzb;
             recState := param.cafRecoverIndPars.recoveryState;
             if (recState=commitState) then
               begin  { local decision as to which one should be kept }
                 if (wantToDetachFirst) then
                   begin
                     recReq.chanIdx := dial.chanIdx;
                     dial.chanIdx := param.chanIdx;
                     detPars.status := free_channel;
                   end
                 else
                   begin
                     recReq.chanIdx := param.chanIdx;
                     detPars.status := free_channel;
                   end
               end
             else
               begin { detach new one }
                 recReq.chanIdx := param.chanIdx;
                 detPars.status := free_channel;
               end;
             {* retry the selected one, detach it *}
             output cpm.TPPM_TO_CPM_req( recReq);
             recReq.order := cafDetachReq;
             recReq.cafDetachPars := detPars;
             output cpm.TPPM_TO_CPM_req( recReq);
           end;
        end;

trans
  when cpm.CPM_TO_TPPM_ind
    provided (param.order=cRecoverInd)
      begin
       forone d: dial_m suchthat (sameBranchId( d.Dzb, param)) do
         output dChanIp[d.dialIdx].C_RECOVER_ind( param.cRecoverIndPars);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when cpm.CPM_TO_TPPM_ind
    provided (param.order=cRecoverCnf)
      begin
       forone d: dial_m suchthat (sameBranchId( d.Dzb, param)) do
         output dChanIp[d.dialIdx].C_RECOVER_cnf( param.cRecoverCnfPars);
      end;

  when cpm.CPM_TO_TPPM_ind
    provided (param.order=afHeurRepInd)
      begin
       forone d: dial_m suchthat (sameBranchId( d.Dzb, param)) do
         output dChanIp[d.dialIdx].AF_HEURISTIC_ind(param.afHeurRepIndPars);
     end;

  when cpm.CPM_TO_TPPM_ind
     provided (param.order=afAbortInd)
       begin
         forone d: dial_m suchthat (sameBranchId( d.Dzb, param)) do
            output dChanIp[d.dialIdx].AF_ABORT_ind( param.afAbortIndPars);
       end;

{** RELAY from TPPM to CPM ***********************************************}

 trans
   any dial: DIAL_IDX do
     when dChanIp[dial].CAF_PLEASE_req
       var cplPars: recoveryReqType;
       begin
         cplPars.branchId := dialogues[dial].Dzb;
         cplPars.atomicActionId := Nzt;
         cplPars.order := cafPleaseReq;
         cplPars.cafPleasePars := param;
         output cpm.TPPM_TO_CPM_req( cplPars);
       end;

     when dChanIp[dial].CAF_DETACH_req
       var recPars: recoveryReqType;
       begin
         recPars.order := cafDetachReq;
         recPars.chanIdx := dialogues[dial].chanIdx;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;

     when dChanIp[dial].C_RECOVER_req
       var recPars: recoveryReqType;
       begin
         recPars.order := cRecoverReq;
         recPars.chanIdx := dialogues[dial].chanIdx;
         recPars.cRecoverReqPars := param;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;

     when dChanIp[dial].C_RECOVER_rsp
       var recPars: recoveryReqType;
       begin
         recPars.order := cRecoverRsp;
         recPars.chanIdx := dialogues[dial].chanIdx;
         recPars.cRecoverRspPars := param;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
   any dial: DIAL_IDX do
      when dChanIp[dial].AF_RECOVER_req
       var recPars: recoveryReqType;
       begin
         recPars.order := afRecoverReq;
         recPars.chanIdx := dialogues[dial].chanIdx;
         recPars.afRecoverReqPars := param;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;

     when dChanIp[dial].AF_HEURISTIC_req
       var recPars: recoveryReqType;
       begin
         recPars.order := afHeurRepReq;
         recPars.chanIdx := dialogues[dial].chanIdx;
         recPars.afHeurRepReqPars := param;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;

     when dChanIp[dial].AF_ABORT_req
       var recPars: recoveryReqType;
       begin
         recPars.order := afAbortReq;
         recPars.chanIdx := dialogues[dial].chanIdx;
         recPars.afAbortReqPars := param;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;

     when dChanIp[dial].AF_TOKEN_GIVE_req
       var recPars: recoveryReqType;
       begin
         recPars.order := afTokenGiveReq;
         recPars.chanIdx := dialogues[dial].chanIdx;
         recPars.afTokenGiveReqPars := param;
         output cpm.TPPM_TO_CPM_req( recPars);
       end;
 end;

{* end of MACF module *************************************************}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*****  dial_m.e: module header for dialogues and transaction branches    **}

{*************************************************************************}
{*                                                                       *}
{*  Dialogue Module                                                      *}
{*                                                                       *}
{*  Dialogue modules keep track of the context of the dialogues maintained *}
{*  by the local TPSUI with other TPSUIs.  Service primitives are received *}
{*  and sent through the interaction point "tpsu".  Dialogue modules have  *}
{*  a nested SACF module providing the AF service and access to ACSE,      *}
{*  presentation, and CCR.  The SACF module itself has a nested interface  *}
{*  module modeling the association supporting the dialogue.               *}
{*                                                                       *}
{*  The module header below represents the external view of the dialogue  *}
{*  by the macf.  The macf will use these variables to synchronize the    *}
{*  behavior of a dialogue with other dialogues when needed.              *}
{*                                                                       *}
{*  There are two types of dialogue module bodies: one for superior dialo- *}
{*  gues, and one for subordinate dialogues.  These module bodies model    *}
{*  the two related behaviors.                                           *}
{*                                                                       *}
{*  During normal operation (i.e., excpet after node crashes), superior    *}
{*  dialogues are assigned an association when they are created.           *}
{*  This association is the association on which the TP site received      *}
{*  the indication trigerring the creation of the TPPM and TPSU.  The      *}
{*  identifier of this association is passed on to the SACF module so that *}
{*  it can attach to it (see SACF modules for details).                    *}
{*                                                                       *}
{*  Except after node crashes, subordinate dialogues are created when the  *}
{*  TPSU requests a new dialogue.                                          *}
{*  The association necessary to support the dialogue may be created by    *}
{*  the SACF module, or it may be extracted from the pool of associations  *}
{*  maintained by the site.  In the latter case, the attributes of the     *}
{*  association are obtained using Estelle primitives.  The exact          *}
{*  implementation of the pool management on a site is a local matter.     *}
{*                                                                       *}
{*  During recovery, dialogue modules interact with the sao through their  *}
{*  "cpm" interaction points.  Primitive sent/received thru this ip are    *}
{*  also processed by the macf module.  When a dialogue is attached to a   *}
{*  channel module, it only receives thru the cpm interaction point        *}
{*  primitives coming through the channel module; other attempts at        *}
{*  recovery by the dialogue peer are rejected by the macf module.         *}
{*                                                                       *}
{*  After a node crash, dialogue branches are created with a branch id and *}
{*  an atomic action Id.  The new dialogues will wait for the restart      *}
{*  signal from the MACF before proceeding.                              *}
{*                                                                       *}
{*************************************************************************}

 module dial_m activity(
     assocId: assocIdType; {association id passed to superior dialogues}
     nodeCrash: boolean;  {is this a restart after a node crash ?}
     actionId: atomicActionIdType; {used only when restarting}
     branchId: branchIdType {used only when restarting}
     );
  ip
     tpsu: TP_service( provider);  {access to tpsu through macf}
     cpm: CPM_service( user);      {access to channel through macf and CPM}
     synchro: synchroMess( dial); {synchronization with macf }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
   export
      Da: boolean;     {dialogue accepted by user - confirm/response sent/rcvd}
      Dabp: boolean;   {dialogue abort rcvd/sent from/to peer}
      Dah: boolean;    {dialogue establishment accepted and held until TP-DONE}
      Danyb: boolean;  {any abort received - true if Dabp, Db, or Dtb}
      Db: boolean;     {dialogue aborted - association not available}
      Dbcr: boolean;   {C-begin confirm received}
      Dbr: boolean;    {subordinate dialogue available for next transaction}
      Dc: boolean;     {had control at the beginning of transaction -}
                       {if rollback occurs, control returns here     }
      Dchat: boolean;  {attached to a channel}
      Dcr: boolean;    {confirmation requested and not yet received/sent}
      Ddef: boolean;   {defer to be sent on prepare }
      Ddp: boolean;    {data ind may be rcvd after prepare req was sent}
      De: boolean;     {deferred end ind/req recvd/sent}
      Denb: integer;   {number of outstanding TP_U_ERROR requests}
      Denbb: integer;  { number of outstanding U_ERROR requests before transact}
      Depnb: integer;  {number of TP-U-ERROR resp to send after begin dial rsp}
      Dg: boolean;     {AF-DEFER(grant-control) sent/rcvd.}
      Dh: boolean;     {handshake functional unit selected}
      Dhb: boolean;    {subordinate must specify recovery-context handle }
                       {when initiating recovery}
      Dhp: boolean;    {superior must specify recovery-context-handle when}
                       {initiating recovery}
      Dl: boolean;     {coordination level is "commitment"}
      Dco: boolean;    {commit functional unit selected}
      Drvyp: boolean;  {C-recover(retry-later) confirm rcvd; TPPM has recovery}
                       {responsability, but branch not under recovery process }
      Dps: boolean;    {prepare req sent}
      Ds: boolean;     {dialogue to the superior}
      Dsh: boolean;    {shared control FU selected}
      Dtb: boolean;    {tp_u_error_req pending}
      Du: boolean;     {unchained transactions FU selected}
      Dzb: branchIdType;  {current branch identifier}
      Dzbn: branchIdType; {branch identifier for next transaction}
      Dzhb: recoveryContextHandleType; {recovery context handle for sub.}
      Dzhp: recoveryContextHandleType; {recovery context handle for superior}
      Nzt: atomicActionIdType;  {atomic action id for current transaction}
      Nztn: atomicActionIdType; {atomic action id for next transaction}
      Nhd: boolean; {log damage}
      Nhe: boolean; {log heuristic}
      inTrans: boolean; {is this node already in a transaction tree}

      dialIdx: integer; {for macf use - index of dialogue in dial. array}
      chanIdx: integer; {for macf use - identifier of associated channel}
      dead: boolean; {dialogue ceased to exist and nested sao is free}
      Drolledback: boolean; {branch rolled back}
      DwrollbackCnf: boolean; {branch waiting for rollback confirm}
      DwBeginInd: boolean; {waiting for begin ind after rollback}
      DcmtIndRcvd: boolean; {commit ind received from superior}
      DcmtCnfRcvd: boolean; {commit confirm received from subordinate}
      DrollbackNext: boolean; {need to rollback next trans}
      DrdyRcvd: boolean; {c_ready received from subordinate}
      DcommitDone: boolean; {commitment complete in node and reported to sup}
      DrollbackDone: boolean; {complete done in node - ready for next trans}
   end;

#include 'dialsup.e'    {module body for superior dialogues}

#include 'dialsub.e'    {module body for subordinate dialogues}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*****  dialsub.e: behavior specific to subordinate dialogues **}

 body dial_sub_b for dial_m;  { module body for subordinate dialogues }

 state
  STATE_1,    {* new dialogue - initial state *}
  STATE_2,    {* Data transfer - shared or polarized control - control *}
  STATE_3,    {* Data transfer - polarized control without control. *}
  STATE_4,    {* AF_U_ERROR_req sent *}
  STATE_5,    {* AF_U_ERROR_in received. *}
  STATE_11,   {* AF_END_DIALOGUE(confirmed)_req sent *}
  STATE_12,   {* AF_END_DIALOGUE(confirmed)_ind received *}
  STATE_6,    {* AF_HANDSHAKE_req sent *}
  STATE_7,    {* AF_HANDSHAKE_ind received *}
  STATE_8,    {* AF_HANDSHAKE_ind/req collision *}
  STATE_9,    {* AF_END_DIALOGUE_ind received - handshake req is pending *}
  STATE_10,   {* AF_HANSHAKE_ind received - end dialogue req is pending *}
  STATE_13,   {* AF_HANDSHAKE_&_GRANT_CONTROL_req sent *}
  STATE_14,   {* AF_HANDSHAKE_AND_GRANT_CONTROL_ind received *}
  STATE_15,   {* TP_PREPARE_req sent - C_READY_ind expected. *}
  STATE_20_1,{* TP_COMMIT_req rcvd - wait for C_READY_ind  *}
  STATE_17,   {* C_READY_ind received - TP_COMMIT_req awaited *}
  STATE_20_2,{* waiting for C_READY_ind on all subordinate dialogues *}
  STATE_20_3,{* READY state - ready reported to sup commit ind expected. *}
  STATE_21_1,{* commitment confirm expected. *}
  STATE_21_2,{* commit confirm awaited - need to rollback next transaction *}
  STATE_21_3,{* commit confirm received *}
  STATE_21_4,{* commit confirm received - rollback next trans initiated *}
  STATE_25,   {* branch destroyed without trigerring rollback *}
  STATE_99,   {* waiting for channel establishment *}
  STATE_23_1,{* rollback request sent - waiting for confirm *}
  STATE_23_2,{* rollback ind/confirm rcvd - rollback complete in subtree *}
  WAIT_FOR_RESTART; {* initial state following node crash *}

  var
   DtbDiag: userDataType;
   Dx: boolean;
   peerCntrl: boolean;
   noPars: emptyParameterList;
   beginPars: cBeginRspType;
   beginDialPars: afBeginDialRspType;
   errPars: afUErrorRspType;
   retryRecovery: boolean;
   detPars: safDetachAssocReqType;
   savedBDInd: afBeginDialIndType;

#include 'sao.e'

 modvar
   sao: sao_m;

 ip
   af: AF_Service( user) common queue;
   saf: SAF_Service( user) common queue;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{ Subordinate Dialogue Initialization                                     }
{                                                                         }
{*************************************************************************}

initialize { normal creation }
  provided not nodeCrash
    to STATE_1
       begin
         Dx := false;
         Ds := false;
         Da := false;
         Dabp := false;
         Dah := false;
         Danyb := false;
         Db := false;
         Dbcr := false;
         Dc := false;
         Dchat := false;
         Dcr := false;
         De := false;
         Denb := 0;
         Depnb := 0;
         Dg := false;
         Dps := false;
         Drvyp := false;
         Dtb := false;
         dead := false;
         Drolledback := false;
         DwrollbackCnf := false;
         DrollbackNext := false;
         DcommitDone := false;
       end;

initialize { creation after node crash }
  provided nodeCrash
    to WAIT_FOR_RESTART
      begin
        Dx := false;
        Dl := true;
        Ds := false;
        Db := true;
        Danyb := true;
        Dchat := false;
        Dzb := branchId;
        Nzt := actionId;
        recoveryContextHndlInTransLog(Nzt,Dhp,Dzhp);
        Dtb := false;
        Dabp := false;
        Drolledback := false;
        DwrollbackCnf := false;
        DrollbackNext := false;
        DcommitDone := false;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  from WAIT_FOR_RESTART
    when synchro.restartInReady
      to STATE_20_3
        begin
          DRdyRcvd := true;
        end;

    when synchro.restartInCommit
      to STATE_99
        var abortPars: tpPAbortIndType;
            cplReq: cafPleaseType;
        begin
          DCmtCnfRcvd := false;
          {+ TPABiPF +}
          abortPars.rollback := false;
          abortPars.diagnostic := ab_permanentFailure;
          output tpsu.TP_P_ABORT_ind( abortPars);
          {+ CPLrq +}
          output cpm.CAF_PLEASE_req( cplreq);
        end;

{*************************************************************************}
{                                                                         }
{  Dialogue Establishment                                                 }
{                                                                         }
{*************************************************************************}

  trans { local reject }
    from STATE_1
      when tpsu.TP_BEGIN_DIALOGUE_req
        provided (rejectDialReq( param))
          to STATE_1
            var tpBegDialPars: tpBeginDialogueCnfType;
            begin
              {+ TBDcRPru or TBDcRP +}
              tpBegDialPars.rollback := beginTransactionReq( param);
              if (unknownRecipient( param)) then
                tpBegDialPars.diagnostic := dr_recipientUnknown
              else
                tpBegDialPars.diagnostic := dr_noReason;
              tpBegDialPars.result := rejected_p;
              output tpsu.TP_BEGIN_DIALOGUE_cnf( tpBegDialPars);
              Dl := false;
              dead := true;
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
   from STATE_1
     when tpsu.TP_BEGIN_DIALOGUE_req
       provided (not rejectDialReq(param))
         to STATE_2
           var cBeginPars: cBeginReqType;
               afBegDialPars: afBeginDialReqType;
           begin
             {+ INITDIASB +}
             Dh := handshakeFuncUnitBDReq(param);
             Dco := commitFuncUnitBDReq(param);
             Du := unchainedFuncUnitBDReq( param);
             Dsh := sharedFuncUnitBDReq( param);
             Ds := false;
             Dcr := param.confirmation=always;
             Dl := beginTransactionReq(param);
             Dbr := Dl and not Du;
             peerCntrl := Dsh;
             {+ ABDRq +}
             mapBeginDialreqTPtoAF( param, afBegDialPars);
             init sao with sao_b( false, nullAssocId, afBegDialPars, true);
             connect sao.af to af;
             if (Dl) then
               begin
                 {+ ADDBRSB +}
                 Dc := true;
                 if (sao.Ahr) then
                   begin
                     Dhp := true;
                     Dzhp := sao.Aahr
                   end;
                 addSubordinateBranchRecord(Nzt,Dzb,Dhp,Dzhp);
                 {+ CBErq +}
                 cBeginPars.atomicActionId := Nzt;
                 cBeginPars.atomicBranchId := Dzb;
                 output af.C_BEGIN_req( cBeginPars);
               end
           end;

trans
  when af.AF_BEGIN_DIALOGUE_cnf
    from STATE_2, STATE_3, STATE_4, STATE_6, STATE_11, STATE_12, STATE_13,
         STATE_15, STATE_20_1, STATE_23_1
      provided ((param.result=accepted) and (param.mapping=dataRI))
        var beginDialPars: tpBeginDialogueCnfType;
        begin
          if (Dcr) then
            begin
              {+ TBDcX +}
              mapBeginDialCnfAFtoTP( param, beginDialPars);
              beginDialPars.rollback := false;
              output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars)
            end;
          Da := true;
          Dcr := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { dialogue reject by peer }
   when af.AF_BEGIN_DIALOGUE_cnf
     from STATE_2, STATE_3, STATE_4, STATE_6, STATE_11, STATE_13
       provided (not Dl and (param.result<>accepted) and (param.mapping=dataRI))
         to STATE_1
           var beginDialPars: tpBeginDialogueCnfType;
           begin
             {+ TBDcX +}
             mapBeginDialCnfAFtoTP( param, beginDialPars);
             beginDialPars.rollback := false;
             output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
             {+ SDETrqF +}
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
           end;

     from STATE_2, STATE_3, STATE_4, STATE_6, STATE_13, STATE_15
       provided (Dl and (param.result<>accepted) and (param.mapping=dataRI)
             and (Du or (param.result=rejected_p)))
         to STATE_25
           var beginDialPars: tpBeginDialogueCnfType;
           begin
             {+ TBDcX +}
             mapBeginDialCnfAFtoTP( param, beginDialPars);
             beginDialPars.rollback := false;
             output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
             {+ SDETrqRB +}
             detPars.status := rollback_ind_expected;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
             {+ ABDET +}
             Db := true;
             Danyb := true;
             Dtb := false;
             Dabp := false;
           end;

     from STATE_2, STATE_3, STATE_4, STATE_6, STATE_13, STATE_15
       provided (Dl and (param.result<>accepted) and (param.mapping=dataRI)
             and (param.result=rejected_u))
         to STATE_25
           var beginDialPars: tpBeginDialogueCnfType;
               crbRsp: cRollbackRspType;
           begin
             {+ TBDcX +}
             mapBeginDialCnfAFtoTP( param, beginDialPars);
             beginDialPars.rollback := false;
             output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
             {+ CRBrs +}
             output af.C_ROLLBACK_rsp( crbRsp);
             {+ SDETrqF +}
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
             {+ ABDET +}
             Db := true;
             Danyb := true;
             Dtb := false;
             Dabp := false;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when saf.SAF_ASSOCIATION_LOST_ind
    from STATE_2, STATE_3, STATE_4, STATE_6, STATE_13, STATE_11
      provided (not D1)
       to STATE_1
          var beginDialPars: tpBeginDialogueCnfType;
          begin
            {+ TBDcRP +}
            beginDialPars.result := rejected_p;
            beginDialPars.rollback := false;
            output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
          end;

    from STATE_2, STATE_3, STATE_4, STATE_6, STATE_13, STATE_11
      provided (D1)
       to STATE_25
          var beginDialPars: tpBeginDialogueCnfType;
          begin
            {+ TBDcRP +}
            beginDialPars.result := rejected_p;
            beginDialPars.rollback := false;
            output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
            {+ ABDET +}
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
            {+ DELBRANCH +}
            deleteSubordinateBranchRecord(Nzt,Dzb);
          end;

trans
  when af.AF_END_DIALOGUE_ind
    provided (D1 and Dsh and Dx and (Denbb=0))
      from STATE_2, STATE_4, STATE_6, STATE_15
       to STATE_25
         var abortPars: tpPAbortIndType;
         begin
           {+ TPABiBTED +}
           abortPars.diagnostic := ab_beginTransactionEndDialCollision;
           abortPars.rollback := true;
           output tpsu.TP_P_ABORT_ind( abortPars);
           {+ SDETrqRB +}
           detPars.status := rollback_ind_expected;
           output saf.SAF_DETACH_ASSOCIATION_req( detPars);
           disconnect sao;
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
           {+ DELBRANCH +}
           deleteSubordinateBranchRecord(Nzt,Dzb);
         end;

    provided (D1 and Dsh and Dx and (Denbb>0))
      from STATE_2, STATE_4, STATE_6, STATE_15, STATE_20_1
       begin
        {+ DECDENB +}
        Denb := Denb-1;
        if (Denbb>0) then Denbb := Denbb-1;
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{ Deferred services                                                       }
{*************************************************************************}

 trans
   when tpsu.TP_DEFERRED_END_DIALOGUE_req
     from STATE_2, STATE_4
       provided (Dl and not De and (Dsh or not peerCntrl))
         var deferPars: afDeferReqType;
         begin
           De := true;
           if (wantToDelayDeferredEnd) then Ddef := true
           else
             begin
               {+ ADErq +}
               Ddef := false;
               deferPars.deferType := endDialogue;
               output af.AF_DEFER_req( deferPars);
             end;
         end;

   when tpsu.TP_DEFERRED_GC_req
     from STATE_2, STATE_4
       provided (Dl and not Dsh and not De and not Dg)
         var deferPars: afDeferReqType;
         begin
           Dg := true;
           if (wantToDelayDeferredGC) then Ddef := true
           else
             begin
               {+ ADGrq +}
               Ddef := false;
               deferPars.deferType := grantControl;
               output af.AF_DEFER_req( deferPars);
             end;
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{   Raising Dialogue level (unchained functional unit)                    }
{*************************************************************************}

 trans
   when tpsu.TP_BEGIN_TRANSACTION_req
     from STATE_2, STATE_4
       provided (Du and not Dl and (not Da or hasToken(sao.assocId))
                 and (Dsh or not peerCntrl))
         var cBeginPars: cBeginReqType;
         begin
           {+ ADDBRSB +}
           Dl := true;
           Dbr := false;
           Dc := true;
           if (sao.Ahr) then
             begin
               Dhp := true;
               Dzhp := sao.Aahr;
             end;
           addSubordinateBranchRecord(Nzt,Dzb,Dhp,Dzhp);
           {+ BEGTRANS +}
           Denbb := Denb;
           Dx := true;
           {+ CBErq +}
           cBeginPars.atomicActionId := Nzt;
           cBeginPars.atomicBranchId := Dzb;
           output af.C_BEGIN_req( cBeginPars);
         end;

 trans
  when af.C_BEGIN_cnf
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15, STATE_20_1,
         STATE_6, STATE_7, STATE_8, STATE_13, STATE_23_1
      provided (Dl)
        begin
          Dbcr := true;
          Dx := false;
        end;

 trans { abort while waiting for Dbcr - collison case }
   when af.AF_ABORT_ind
     provided ((param.mapping=dataRI) and Dl and Du and not Dbcr and not Dcr)
       from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15, STATE_6,
            STATE_7, STATE_13, STATE_14
         to STATE_25
           var uAbortPars: tpUAbortIndType;
           begin
             uAbortPars.rollback := false;
             output tpsu.TP_U_ABORT_ind( uAbortPars);
             {+ SDETrqRB +}
             detPars.status := rollback_ind_expected;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
             {+ ABDET +}
             Db := true;
             Danyb := true;
             Dtb := false;
             Dabp := false;
             {+ DELBRANCH +}
             deleteSubordinateBranchRecord(Nzt,Dzb);
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{ COMMITMENT - preparing for commitment of subordinate subtree            }
{                                                                         }
{*************************************************************************}

 trans
  when tpsu.TP_PREPARE_req
    from STATE_2, STATE_4
      provided (D1 and not Dcr and (Dsh or not peerCntrl))
        to STATE_15
          var prepPars: afPrepareReqType;
              deferPars: afDeferReqType;
          begin
            DrdyRcvd := false;
            Dps := true;
            {+ maybe SENDEF +}
            if (Ddef) then
              begin
               Ddef := false;
               if (De) then
                 begin
                   deferPars.deferType := endDialogue;
                   output af.AF_DEFER_req( deferPars);
                 end
               else if (Dg) then
                 begin
                  deferPars.deferType := grantControl;
                  output af.AF_DEFER_req( deferPars);
                 end;
              end;
            {+ VDdp(T/F) +}
            Ddp := param.dataPermitted or Dsh;
            {+ APRrq +}
            prepPars.dataPermitted := Ddp;
            output af.AF_PREPARE_req( prepPars);
          end;

 trans {** waiting for subordinate subtree to be ready **}
  when af.C_READY_ind
    from STATE_15
      to STATE_17
        begin
          {+ TRYi +}
          output tpsu.TP_READY_ind( param);
          {+ COUNTRDY +}
          DrdyRcvd := true;
        end;

    from STATE_20_1
      to STATE_20_2
        begin
          {+ COUNTRDY +}
          DrdyRcvd := true;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
 when tpsu.TP_COMMIT_req
   from STATE_17
     to STATE_20_2
       begin
         DRdyRcvd := true;
       end;

   from STATE_25
     to SAME
       begin
         DrdyRcvd := true;
       end;

   from STATE_2, STATE_4, STATE_15
     provided (Dl and not Dcr)
       to STATE_20_1
         var prepPars: afPrepareReqType;
             deferPars: afDeferReqType;
         begin
           if (not Dps) then
             begin
               {+ APRrq and maybe SENDDEF +}
               Dps := true;
               prepPars.dataPermitted := false;
               output af.AF_PREPARE_req( prepPars);
               DRdyRcvd := false;
               if (Ddef) then
                 begin
                   if (De) then
                     deferPars.deferType := endDialogue
                   else if (Dg) then
                     deferPars.deferType := grantControl;
                   output af.AF_DEFER_req( deferPars);
                   Ddef := false;
                 end;
             end;
          end;

trans
   when af.U_ASE_ind
     from STATE_15
       provided (not Dcr and (Dsh or Ddp))
         begin
           {+ TDTi +}
           output tpsu.TP_DATA_ind( param)
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{* SUBORDINATE SUBTREE IN RDY STATE - WAITING FOR WHOLE NODE TO BE READY *}

 trans
   when af.AF_ABORT_ind
     provided (param.mapping=abortRI)
       from STATE_20_3
         to STATE_20_3
            var abortPars: tpPAbortIndType;
            begin
              {+ TPABiPF if not Danyb +}
              if (not Danyb) then
                begin
                 abortPars.rollback := false;
                 abortPars.diagnostic :=  param.diagnostic;
                 output tpsu.TP_P_ABORT_ind( abortPars);
                end;
              {+ VDchatF +}
              Dchat := false;
              {+ ABDET +}
              Db := true;
              Danyb := true;
              Dabp := false;
              Dtb := false;
            end;

 trans
  from STATE_20_1
    when synchro.sendReady
      to STATE_20_3
        begin
        end;

{*************************************************************************}
{                                                                         }
{ Trigerring ROLLBACK of subordinate subtree                              }
{                                                                         }
{*************************************************************************}

 trans {* Rollback initiated by local TPSU *}
   when tpsu.TP_ROLLBACK_req
     from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15,
          STATE_6, STATE_7, STATE_8, STATE_13, STATE_14, STATE_17
       provided (Dl)
         to STATE_23_1
           var abortPars: afAbortReqType;
               crbReq: cRollbackReqType;
           begin
             {+ RBREQ +}
             if (not Dtb) then
               output af.C_ROLLBACK_req( crbReq)
             else
               begin
                 Dtb := false;
                 Dabp := true;
                 abortPars.aType := userAbort;
                 abortPars.mapping := rollbackRI;
                 abortPars.userData := DtbDiag;
                 output af.AF_ABORT_req( abortPars)
               end;
             {+ INITRB - see MACF +}
             DwrollbackCnf := true
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when tpsu.TP_ROLLBACK_req
    from STATE_25
      to STATE_23_2
        begin
          DwrollbackCnf := false
        end;

  when tpsu.TP_U_ABORT_req
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15,
         STATE_6, STATE_7, STATE_8, STATE_13, STATE_14,
         STATE_17
      provided (Dl)
        to STATE_23_1
          var abortPars: afAbortReqType;
          begin
            {+ ABTPSUI +}
            Danyb := true;
            Dtb := true;
            {+ NOTAVNXT +}
            Dbr := false;
            {+ INITRB  and rollback all +}
            abortPars.aType := userAbort;
            abortPars.mapping := rollbackRI;
            output af.AF_ABORT_req( abortPars);
            Drolledback := true;
            DwRollbackCnf := true;
          end;

trans  {* Rollback originating in subordinate subtree *}
  when af.C_ROLLBACK_ind
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15, STATE_20_1,
         STATE_6, STATE_7, STATE_8, STATE_13, STATE_14
      provided (Dl and not Dcr)
        to STATE_23_2
          var crbRsp: cRollbackRspType;
          begin
            {+ TRBi +}
            output tpsu.TP_ROLLBACK_ind( noPars);
            {+ CRBrs +}
            output af.C_ROLLBACK_rsp( crbRsp);
            {+ CPSAP if unchained +}
            if (Du) then sao.locked := true;
            {+ INITRB - COUNTRB +}
            Drolledback := true;
            DwrollbackCnf := false;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.AF_HEURISTIC_ind
  from STATE_15, STATE_20_1
    provided ((param.mapping=rollbackRI) and not Dcr)
      to STATE_23_2
        var crbRsp: cRollbackRspType;
            heurRep: tpHeuristicReportIndType;
        begin
          {+ TRBi +}
          output tpsu.TP_ROLLBACK_ind( noPars);
          {+ THRi +}
          heurRep.heuristicReport := param.heuristicReport;
          output tpsu.TP_HEURISTIC_REPORT_ind( heurRep);
          {+ CRBrs +}
          output af.C_ROLLBACK_rsp( crbRsp);
          {+ INITRB - COUNTRB +}
          Drolledback := true;
          DwrollbackCnf := false;
          {+ CPSAP +}
          if (Du) then
            sao.locked := true;
        end;

when af.AF_ABORT_HEURISTIC_ind
  from STATE_15, STATE_20_1
    provided ((param.mapping=rollbackRI) and not Dcr)
      to STATE_23_2
        var abortPars: tpUAbortIndType;
            crbRsp: cRollbackRspType;
            heurPars: tpHeuristicReportIndType;
        begin
          {+ TUABiR +}
          abortPars.rollback := true;
          output tpsu.TP_U_ABORT_ind( abortPars);
          {+ THRi +}
          heurPars.heuristicReport := param.heuristicReport;
          output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
          {+ CRBrs +}
          output af.C_ROLLBACK_rsp( crbRsp);
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ INITRB and COUNTRB +}
          Drolledback := true;
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.AF_ABORT_ind
  provided (Dl and (param.mapping=abortRI))
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_6, STATE_7, STATE_8,
         STATE_13, STATE_14, STATE_15, STATE_20_1, STATE_17, STATE_20_2
      to STATE_23_2
         var pAbortPars: tpPAbortIndType;
             crbPars : cRollbackRspType;
             heurPars: tpHeuristicReportIndType;
         begin
           {+ TPABiPFr +}
           pAbortPars.rollback := true;
           pAbortPars.diagnostic :=  param.diagnostic;
           output tpsu.TP_P_ABORT_ind( pAbortPars);
           if (Dps) then
             begin
               {+ THRiH +}
               heurPars.heuristicReport := heuristicHazard;
               output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
             end;
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
           {+ NOTAVNXT +}
           Dbr := false;
           {+ INITRB - COUNTRB +}
           Drolledback := true;
           DwRollbackCnf := false;
         end;

when af.AF_BEGIN_DIALOGUE_cnf
  from STATE_20_1
    provided (((param.result=rejected_u) and Du)
           or (param.result=rejected_p))
      to STATE_23_2
         var beginDialPars: tpBeginDialogueCnfType;
         begin
           {+ TBDcXr +}
           mapBeginDialCnfAFtoTP( param, beginDialPars);
           beginDialPars.rollback := true;
           output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
           {+ SDETrqRB +}
           detPars.status := rollback_ind_expected;
           output saf.SAF_DETACH_ASSOCIATION_req( detPars);
           disconnect sao;
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
           {+ NOTAVNXT +}
           Dbr := false;
           {+ INITRB +}
           Drolledback := true;
           DwrollbackCnf := false;
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when saf.SAF_ASSOCIATION_LOST_ind
  from STATE_20_1
    to STATE_23_2
       var beginDialPars: tpBeginDialogueCnfType;
       begin
         {+ TBDcRPr +}
         beginDialPars.result := rejected_p;
         beginDialPars.rollback := true;
         output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
         {+ ABDET +}
         Db := true;
         Danyb := true;
         Dtb := false;
         Dabp := false;
         {+ NOTAVNXT +}
         Dbr := false;
         {+ INITRB +}
         Drolledback := true;
         DwrollbackCnf := false;
       end;

when af.AF_END_DIALOGUE_ind
  provided (Dl and Dsh and Dx and (Denbb=0))
    from STATE_20_1
      to STATE_23_2
        var abortPars: tpPAbortIndType;
        begin
          {+ TPABiBTEDr +}
          abortPars.diagnostic := ab_beginTransactionEndDialCollision;
          abortPars.rollback := true;
          output tpsu.TP_P_ABORT_ind( abortPars);
          {+ SDETrqRB +}
          detPars.status := rollback_ind_expected;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ INITRB +}
          Drolledback := true;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.AF_ABORT_ind
  provided ((param.mapping=dataRI) and not Dbcr)
    from STATE_20_1
      to STATE_23_2
         var uAbortPars: tpUAbortIndType;
         begin
          {+ TUABiR +}
          uAbortPars.rollback := true;
          output tpsu.TP_U_ABORT_ind( uAbortPars);
          {+ SDETrqRB +}
          detPars.status := rollback_ind_expected;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ INITRB - COUNTRB +}
          Drolledback := true;
          DwrollbackCnf := false;
        end;

  provided ((param.mapping=rollbackRI)
     and(((param.aType=providerAbort) and Du and not Dbcr)
         or ((param.aType=userAbort) and not Dcr)))
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15, STATE_20_1,
         STATE_6, STATE_7, STATE_8, STATE_13, STATE_14
      to STATE_23_2
         var uAbortPars: tpUAbortIndType;
             pAbortPars: tpPAbortIndType;
             crspPars: cRollbackRspType;
         begin
           if (param.aType=userAbort) then
             begin {+ TUABiR +}
               uAbortPars.rollback := true;
               output tpsu.TP_U_ABORT_ind( uAbortPars)
             end
           else
             begin {+ TPABiPFr +}
               pAbortPars.diagnostic := param.diagnostic;
               pAbortPars.rollback := true;
               output tpsu.TP_P_ABORT_ind( pAbortPars)
             end;
          {+ CRBrs +}
          output af.C_ROLLBACK_rsp( crspPars);
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ INITRB - COUNTRB +}
          Drolledback := true;
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.U_ASE_ind
   from STATE_20_1
    provided (not Dcr and not Dsh and not Ddp)
      to STATE_23_1
        var crbReq: cRollbackReqType;
        begin                output tpsu.TP_ROLLBACK_ind( noPars);
          {+ INITRB - rollback-all - CRBrq +}
          DrolledBack := true;
          DwrollbackCnf := true;
          output af.C_ROLLBACK_req( crbReq);
        end;

when af.AF_U_ERROR_ind
  from STATE_15, STATE_20_1
    to STATE_23_1
      var crbReq: cRollbackReqType;
      begin
        {+ TRBi +}
        output tpsu.TP_ROLLBACK_ind( noPars);
        {+ INITRB - rollback-all +}
        output af.C_ROLLBACK_req( crbReq);
        DrolledBack := true;
        DwrollbackCnf := true;
      end;

when af.AF_HANDSHAKE_ind
  from STATE_15, STATE_20_1
    to STATE_23_1
      var crbReq: cRollbackReqType;
      begin
        {+ TRBi +}
        output tpsu.TP_ROLLBACK_ind( noPars);
        {+ INITRB - rollback-all +}
        output af.C_ROLLBACK_req( crbReq);
        DrolledBack := true;
        DwrollbackCnf := true;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {* rollback resulting from rollback of other dialogue *}
  when synchro.rollbackAll
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_15, STATE_20_1,
         STATE_6, STATE_7, STATE_8, STATE_13, STATE_14, STATE_17,
         STATE_20_2
      to STATE_23_1
         var crbReq: cRollbackReqType;
             abortPars: afAbortReqType;
         begin
           {+ RBREQ +}
           if (Dtb) then
             begin
               abortPars.mapping := rollbackRI;
               abortPars.aType := userAbort;
               abortPars.userData := DtbDiag;
               output af.AF_ABORT_req( abortPars);
               DwrollbackCnf := false;
               Dabp := true;
               Dtb := false;
             end
           else
             begin
               output af.C_ROLLBACK_req( crbReq);
               DwrollbackCnf := true;
             end;
         end;

 when synchro.rollbackAll
   provided (not Db)
     from STATE_20_3  { READY STATE }
       to STATE_23_1
          var crbReq: cRollbackReqType;
              abortPars: afAbortReqType;
       begin
         {+ RBREQ +}
         if (Dtb) then
           begin
             abortPars.mapping := rollbackRI;
             abortPars.aType := userAbort;
             abortPars.userData := DtbDiag;
             output af.AF_ABORT_req( abortPars);
             DwrollbackCnf := false;
             Dabp := true;
             Dtb := false;
           end
         else
           begin
             output af.C_ROLLBACK_req( crbReq);
             DwrollbackCnf := true;
           end;
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when synchro.rollbackAll
  provided (Db)
   from STATE_20_3, STATE_25
     to STATE_23_2
        var recPars: cRecoverReqType;
            cdet: cafDetachType;
     begin
       if (Dchat) then
         begin
           {+ CRErsU +}
           recPars.recoveryState := unknownState;
           recPars.atomicActionId := Nzt;
           recPars.branchId := Dzb;
           output cpm.C_RECOVER_req(recPars);
           {+ CDETrq +}
           cdet.status := free_channel;
           output cpm.CAF_DETACH_req( cdet);
           {+ VDchatF +}
           Dchat := true;
         end;
     end;

when synchro.rollbackAll
  from STATE_23_1, STATE_23_2
    begin
    end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{  COMMITMENT - Phase 2 - commitment of subordinate subtree               }
{                                                                         }
{*************************************************************************}

 trans { sending commitment order to subordinate subtree }
   when synchro.sendCommitOrder
     provided (not Db)
       from STATE_20_2, STATE_20_3
         to STATE_21_1
            var cComBegPars: cCommitBeginReqType;
                abortPars: afAbortReqType;
                cComPars: cCommitReqType;
         begin
           {+ COMREQ +}
           if (Dbr) then
             begin
               cComBegPars.atomicActionId := Nztn;
               cComBegPars.atomicBranchId := Dzbn;
               output af.C_COMMIT_C_BEGIN_req( cComBegPars);
               addSubordinateBranchRecord(Nztn,Dzbn,Dhp,Dzhp);
             end
           else if (Dtb) then
             begin
               abortPars.aType :=  userAbort;
               abortPars.userData := DtbDiag;
               abortPars.mapping := commitRI;
               output af.AF_ABORT_req( abortPars);
               Dabp := true;
               Dtb := false;
             end
           else
             output af.C_COMMIT_req( cComPars);
           {+ if De NOTAVNXT +}
           if (De) then Dbr := false;
           DcmtCnfRcvd := false;
         end;

   when synchro.sendCommitOrder
     provided (Db and not Dchat)
       from STATE_20_3
         to STATE_99
           var cplPars: cafPleaseType;
         begin
           {+ CPLrq +}
           DcmtCnfRcvd := false;
           output cpm.CAF_PLEASE_req( cplPars);
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when synchro.sendCommitOrder
    provided (Dchat)
      from STATE_20_3
        to STATE_21_1
          var cRecPars: cRecoverReqType;
              afRecPars: afRecoverReqType;
              tokPars: afTokenGiveReqType;
          begin
            {+ RCVRCOM +}
            if (Dhp) then
              begin
                afRecPars.mapping := recoverComRI;
                afRecPars.recoveryContextHandle := Dzhp;
                afRecPars.atomicActionId := Nzt;
                afRecPars.branchId := Dzb;
                output cpm.AF_RECOVER_req( afRecPars);
              end
            else
              begin
                cRecPars.recoveryState := commitState;
                cRecPars.atomicActionId := Nzt;
                cRecPars.branchId := Dzb;
                output cpm.C_RECOVER_req( cRecPars);
              end;
            output cpm.AF_TOKEN_GIVE_req( tokPars);{CPM checks (Dtwr & PTok)}
            DcmtCnfRcvd := false;
          end;

  when synchro.sendCommitOrder
    from STATE_25
      to STATE_21_3
        begin
          DcmtCnfRcvd := true;
        end;

trans {* rollback next transaction is pending - if conf of commit was *)
      {* received, then initiate rollback of subordinate subtree      *}
  when synchro.rPend
    from STATE_21_1
      provided (not Db and Dbr)
        to STATE_21_2
          begin
          end;

    from STATE_21_3
      provided (not Danyb and Dbr)
        to STATE_21_4
          var crbReq: cRollbackReqType;
          begin {+ CRBrq +}
            output af.C_ROLLBACK_req( crbReq);
          end;

      provided (Dtb and Dbr)
        to STATE_21_3
          var abortPars : afAbortReqType;
          begin
            abortPars.mapping := rollbackRI;
            abortPars.aType := userAbort;
            output af.AF_ABORT_req( abortPars);
            detPars.status := rollback_cnf_expected;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          end;
```

ISONORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    from STATE_21_1, STATE_21_3, STATE_99
      provided (Db or not Dbr)
        begin
        end;

trans { completing commitment of subordinate subtree }
  when af.C_COMMIT_cnf
    from STATE_21_1
      to STATE_21_3
        var abortPars: afAbortReqType;
        begin
          {+ COUNTCOM +}
          DcmtcnfRcvd := true;
          if (not Danyb and not De) then
             sao.locked:= true  {+ CPSAP +}
          else if (De or Dabp or Dtb) then
            begin
              if (Dtb) then
                begin
                  {+ AABrqUd +}
                  abortPars.mapping := dataRI;
                  abortPars.userData := DtbDiag;
                  abortPars.aType := userAbort;
                  output af.AF_ABORT_req( abortPars);
                  Dtb := false;
                end;
              {+ SDETrqF +}
              detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
              {+ ABDET +}
              Danyb := true;
              Db := true;
              Dabp := false;
            end
        end;

  when af.C_COMMIT_cnf
    from STATE_21_2
      provided (not Danyb)
        to STATE_21_4
          var abortPars: afAbortReqType;
              crbReq: cRollbackReqType;
          begin
            {+ COUNTCOM +}
            DcmtcnfRcvd := true;
            {+ CRBrq +}
            output af.C_ROLLBACK_req( crbReq);
            {+ CPSAP +}
            sao.locked:= true;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when af.C_COMMIT_cnf
    from STATE_21_2
      provided (not Dabp and Dtb)
        to STATE_21_3
          var abortPars: afAbortReqType;
          begin
            {+ COUNTCOM +}
            DcmtcnfRcvd := true;
            {+ AABrqUd +}
            abortPars.mapping := rollbackRI;
            abortPars.userData := DtbDiag;
            abortPars.aType := userAbort;
            output af.AF_ABORT_req( abortPars);
            Dtb := false;
            {+ SDETrqF +}
            detPars.status := rollback_cnf_expected;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
            {+ ABDET +}
            Danyb := true;
            Db := true;
            Dabp := false;
          end;

trans { commitment with heuristic report }
  when af.AF_HEURISTIC_ind
    provided (param.mapping=commitRC)
      from STATE_21_1
        to STATE_21_3
          var heurPars: tpheuristicReportIndType;
              abortPars: afAbortReqType;
          begin
            {+ COUNTCOM +}
            DcmtcnfRcvd := true;
            {+ THRi +}
            heurPars.heuristicReport := param.heuristicReport;
            output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
            if (not Danyb and not De) then
               {+ CPSAP +}
               sao.locked := true
            else if (not Danyb and De or Dabp or Dtb) then
               begin
                 {+ SDETrqF and ABDET +}
                 if (Dtb) then
                   begin
                     abortPars.mapping := dataRI;
                     abortPars.userData := DtbDiag;
                     abortPars.aType := userAbort;
                     output af.AF_ABORT_req( abortPars);
                     Dtb := false;
                   end;
                 Danyb := true;
                 Db := true;
                 Dtb := false;
                 Dabp := false;
                 detPars.status := free_assoc;
                 output saf.SAF_DETACH_ASSOCIATION_req( detPars);
                 disconnect sao;
               end;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.AF_HEURISTIC_ind
  provided (param.mapping=commitRC) and not Danyb
    from STATE_21_2
      to STATE_21_4
        var heurPars: tpheuristicReportIndType;
            crbReq: cRollbackReqType;
        begin
          {+ COUNTCOM +}
          DcmtcnfRcvd := true;
          {+ THRi +}
          heurPars.heuristicReport := param.heuristicReport;
          output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
          {+ CRBrq +}
          output af.C_ROLLBACK_req( crbReq);
          {+ CPSAP }
          sao.locked:= true;
        end;

when af.AF_HEURISTIC_ind
  provided (param.mapping=commitRC) and not Dabp and Dtb
    from STATE_21_2
      to STATE_21_3
        var heurPars: tpheuristicReportIndType;
            abortPars: afAbortReqType;
        begin
          {+ COUNTCOM +}
          DcmtcnfRcvd := true;
          {+ THRi +}
          heurPars.heuristicReport := param.heuristicReport;
          output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
          {+ AABrqUr +}
          abortPars.mapping := rollbackRI;
          abortPars.userData := DtbDiag;
          abortPars.aType := userAbort;
          output af.AF_ABORT_req( abortPars);
          Dtb := false;
          {+ SDETrqRBC +}
          detPars.status := rollback_cnf_expected;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ CPSAP +}
          sao.locked:= true;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when af.AF_ABORT_ind
    from STATE_21_1
      provided ((param.aType=userAbort) and
                (param.mapping=commitRC) and (not Danyb or not Dabp))
       to STATE_21_3
         var abortPars: tpUAbortIndType;
         begin
           {+ TUABi  if not Daynb +}
           if (not Danyb) then
             begin
               abortPars.rollback := false;
               output tpsu.TP_U_ABORT_ind( abortPars);
             end;
           {+ RBNEXT if Dbr +}
           if (Dbr) then DrollbackNext := true;
           {+ COUNTCOM +}
           DcmtcnfRcvd := true;
           {+ SDETrqF +}
           detPars.status := free_assoc;
           output saf.SAF_DETACH_ASSOCIATION_req( detPars);
           disconnect sao;
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
         end;

trans
  when af.AF_ABORT_HEURISTIC_ind
    from STATE_21_1
      provided ((param.mapping=commitRC) and (not Danyb or not Dabp))
       to STATE_21_3
         var abortPars: tpUAbortIndType;
             heurPars: tpheuristicReportIndType;
         begin
           {+ TUABi  if not Daynb +}
           if (not Danyb) then
             begin
               abortPars.rollback := false;
               output tpsu.TP_U_ABORT_ind( abortPars);
             end;
           {+ THRi +}
           heurPars.heuristicReport := param.heuristicReport;
           output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
           {+ RBNEXT if Dbr +}
           if (Dbr) then DrollbackNext := true;
           {+ COUNTCOM +}
           DcmtcnfRcvd := true;
           {+ SDETrqF +}
           detPars.status := free_assoc;
           output saf.SAF_DETACH_ASSOCIATION_req( detPars);
           disconnect sao;
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when af.AF_ABORT_ind
    provided (param.mapping=abortRI)
     from STATE_21_1, STATE_21_2
       to STATE_99
         var abortPars: tpPAbortIndType;
             cplPars: cafPleaseType;
         begin
           {+ TPABiPF +}
           if (not Danyb) then
             begin
               abortPars.rollback := false;
               abortPars.diagnostic := param.diagnostic;
               output tpsu.TP_P_ABORT_ind( abortPars);
             end;
           {+ RBNEXT +}
           if (Dbr) then DrollbackNext := true;
           {+ CPLrq +}
           output cpm.CAF_PLEASE_req( cplPars);
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
         end;


{*************************************************************************}
{                                                                         }
{ COMMITMENT - Phase 2 - completion of commitment on whole TPPM subtree   }
{                                                                         }
{*************************************************************************}

 trans  { not part of next transaction tree or no rollback pending }
   from STATE_21_3
     when synchro.commitComplete
       provided (Danyb or De)
         to STATE_1
           begin
             DcommitDone := true;
             {+ CMPCOMSB +}
             sao.locked := false;
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
             {+ DELBR +}
             Dl := false;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_21_3
   when synchro.commitComplete
      provided (not Danyb and not De and (Dsh or not Dg))
       to STATE_2
         begin
           {+ CMPCOMSB +}
           if (Dbr) then Dzb := Dzbn;
           Dc := not Dg;
           sao.locked := false;
           {+ DELBR if not Dbr +}
           Dl := not Dbr;
           {+ RESETD +}
           Dg := false;
           Ddef := false;
           Denb := 0;
           Depnb := 0;
           if (Du) then Dbcr := false;
           peerCntrl := Dsh;
           DcommitDone := true;
         end;

     provided (not Danyb and not De and not Dsh and Dg)
       to STATE_3
         begin
           {+ CMPCOMSB +}
           sao.locked := false;
           if (Dbr) then Dzb := Dzbn;
           Dc := not Dg;
           {+ DELBR +}
           Dl := not Dbr;
           {+ RESETD +}
           Dg := false;
           Ddef := false;
           Denb := 0;
           Depnb := 0;
           if (Du) then Dbcr := false;
           peerCntrl := Dsh;
           DcommitDone := true;
         end;


trans { rolling back next transaction }
  from STATE_21_4
    when synchro.commitComplete
      to STATE_23_1
        begin
          {+ CMPCOMSB +}
          sao.locked := false;
          if (Dbr) then Dzb := Dzbn;
          Dc := not Dg;
          DwrollbackCnf := true;
          DcommitDone := true;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{ COMPLETING ROLLBACK OF SUBORDINATE SUBTREE                              }
{                                                                         }
{*************************************************************************}

{*** rollback trigerred by this side ************************************}

trans
  from STATE_23_1
    to STATE_23_2
      when af.C_ROLLBACK_ind (* collision with subordinate rollback *)
        var abortPars: afAbortReqType;
            crbRsp: cRollbackRspType;
        begin
          if (not Danyb) then
            begin
              {+ CRBrs +}
              output af.C_ROLLBACK_rsp( crbRsp);
              {+ CPSAP +}
              sao.locked := not Dbr;
            end
          else
            begin
              {+ AABrqUrbc +}
              abortPars.userData := DtbDiag;
              abortPars.mapping := rollbackRC;
              abortPars.aType := userAbort;
              {+ SDETrqF +}
              output af.AF_ABORT_req( abortPars);
              detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
              {+ ABDET +}
              Dtb := false;
              Dabp := false;
              Danyb := true;
              Db := true;
            end;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
    when af.C_ROLLBACK_cnf
      var abortPars: afAbortReqType;
      begin
        if (Danyb) then
          begin
            if (Dtb) then
              begin
                {+ AABrqUd +}
                abortPars.userData := DtbDiag;
                abortPars.mapping := dataRI;
                abortPars.aType := userAbort;
                output af.AF_ABORT_req( abortPars);
              end;
            {+ SDETrqF +}
            detPars.status := free_assoc;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
            {+ ABDET +}
            Dtb := false;
            Dabp := false;
            Danyb := true;
            Db := true;
          end;
        {+ CPSAP +}
        if (Dbr) then sao.locked := true;
        {+ COUNTRB +}
        DwrollbackCnf := false;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
    when af.AF_HEURISTIC_ind
      provided ((param.mapping=rollbackRI) or (param.mapping=rollbackRC))
        var abortPars: afAbortReqType;
            crbRsp: cRollbackRspType;
            heurPars: tpHeuristicReportIndType;
        begin
          {+ THRi +}
          heurPars.heuristicReport := param.heuristicReport;
          output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
          if (not Danyb) then
            begin
              {+ CRBrs +}
              output af.C_ROLLBACK_rsp( crbRsp);
              {+ CPSAP +}
              sao.locked := not Dbr;
            end
          else
            begin
              {+ AABrqUrbc or AABrqUd +}
              abortPars.userData := DtbDiag;
              if (param.mapping=rollbackRI) then
                abortPars.mapping := rollbackRC
              else
                abortPars.mapping := dataRI;
              abortPars.aType := userAbort;
              output af.AF_ABORT_req( abortPars);
              {+ SDETrqF +}
              detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
              {+ ABDET +}
              Dtb := false;
              Dabp := false;
              Danyb := true;
              Db := true;
            end;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
    when af.AF_ABORT_HEURISTIC_ind
      provided ((param.mapping=rollbackRI) or (param.mapping=rollbackRC))
        var abortPars: tpUAbortIndType;
            crbRsp: cRollbackRspType;
            heurPars: tpHeuristicReportIndType;
        begin
          {+ TUABi +}
          if (not Danyb) then
            begin
              abortPars.rollback := false;
              output tpsu.TP_U_ABORT_ind( abortPars);
            end;
          {+ THRi +}
          heurPars.heuristicReport := param.heuristicReport;
          output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
          {+ NOTAVNXT +}
          Dbr := false;
          {+ CRBrs +}
          if (param.mapping=rollbackRI) then
            output af.C_ROLLBACK_rsp( crbRsp);
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1, STATE_23_2
  to STATE_23_2
    when af.AF_ABORT_ind
      provided ((param.mapping=dataRI) and (Dx or not DwrollbackCnf))
        var uAbortPars: tpUAbortIndType;
        begin
          if (not Danyb) then
            begin
              {+ TUABi +}
              uAbortPars.rollback := false;
              output tpsu.TP_U_ABORT_ind( uAbortPars);
            end;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ SDET +}
          if (DwrollbackCnf) then
            detPars.status := rollback_cnf_expected
          else
            detPars.status := rollback_cnf_expected;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;


from STATE_23_1, STATE_23_2
  to STATE_23_2
    when af.AF_ABORT_ind
      provided (param.mapping=abortRI)
        var pAbortPars: tpPAbortIndType;
            heurPars: tpHeuristicReportIndType;
        begin
          if (Dps) then
            begin
              {+ THRi +}
              output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
            end;
          if (not Danyb) then
            begin
              pAbortPars.diagnostic := param.diagnostic;
              pAbortPars.rollback := false;
              output tpsu.TP_P_ABORT_ind( pAbortPars);
            end;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ COUNTRB (if necessary)+}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
    when af.AF_ABORT_ind
      provided ((param.mapping=rollbackRI) or (param.mapping=rollbackRC))
        var uAbortPars: tpUAbortIndType;
            pAbortPars: tpPAbortIndType;
            crbRsp: cRollbackRspType;
        begin
          if (not Danyb) then
            if (param.aType=providerAbort) then
               begin
                 pAbortPars.diagnostic := param.diagnostic;
                 pAbortPars.rollback := false;
                 output tpsu.TP_P_ABORT_ind( pAbortPars);
               end
            else
               begin
                 uAbortPars.rollback := false;
                 output tpsu.TP_U_ABORT_ind( uAbortPars);
               end;
          if (param.mapping=rollbackRI) then
             output af.C_ROLLBACK_rsp( crbRsp);
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
   when af.AF_BEGIN_DIALOGUE_cnf
     provided ((param.result=accepted) and (param.mapping=rollbackRC))
       var begPars: tpBeginDialogueCnfType;
           abortPars: afAbortReqType;
       begin
         if (Dtb) then
           begin
             {+ AABrqUd +}
             abortPars.userData := dtbDiag;
             abortPars.aType := userAbort;
             abortPars.mapping := dataRI;
             output af.AF_ABORT_req( abortPars);
             {+ SDETrqF +}
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
             {+ ABDET +}
             Db := true;
             Danyb := true;
             Dabp := false;
             Dtb := false;
           end
         else
           begin
             if (Dcr) then
               begin
                 {+ TBDcX +}
                 begPars.result := accepted;
                 output tpsu.TP_BEGIN_DIALOGUE_cnf( begPars);
                 Dcr := false;
               end;
             Da := true;
             {+ if not Dbr CPSAP +}
             sao.locked := not Dbr;
           end;
         DwRollbackCnf := false;
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
   when af.AF_BEGIN_DIALOGUE_cnf
     provided (((param.result=rejected_u) and Du)
               or (param.result=rejected_p))
        var beginDialPars: tpBeginDialogueCnfType;
            rbrsp: cRollbackRspType;
        begin
          if (not Danyb) then
             begin {+ TBDcX +}
               mapBeginDialCnfAFtoTP( param, beginDialPars);
               beginDialPars.rollback := false;
               output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
             end;
          if (param.mapping=dataRI) then
            begin
              {+ SDETrqRBC +}
              detPars.status := rollback_ind_expected;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
            end
          else
            begin
              if (param.mapping=rollbackRI) then
                begin
                  {+ CRBrs +}
                  output af.C_ROLLBACK_rsp( rbrsp);
                end;
              {+ SDETrqF +}
              detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
            end;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_1
  to STATE_23_2
    when af.AF_END_DIALOGUE_ind
      provided (Dx and (not param.confirmation or (Denbb=0)))
        var abortPars: tpPAbortIndType;
        begin
          if (not Danyb) then
            begin
             {+ TPABiBTED +}
             abortPars.diagnostic := ab_beginTransactionEndDialCollision;
             abortPars.rollback := false;
             output tpsu.TP_P_ABORT_ind( abortPars);
            end;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ SDETrqRBC +}
          detPars.status := rollback_cnf_expected;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ COUNTRB +}
          DwrollbackCnf := false;
        end;

from STATE_23_1
  to STATE_23_1
    when af.AF_END_DIALOGUE_ind
      provided (Dx and param.confirmation and (Denbb>0))
        begin
         {+ DECDENB +}
         Denb := Denb-1;
         if (Denbb>0) then Denbb := Denbb-1;
        end;

from STATE_23_1
  to STATE_23_2
    when saf.SAF_ASSOCIATION_LOST_ind
       var beginDialPars: tpBeginDialogueCnfType;
       begin
         if (not Danyb) then
           begin
             {+ TBDcRP +}
             beginDialPars.result := rejected_p;
             beginDialPars.rollback := false;
             output tpsu.TP_BEGIN_DIALOGUE_cnf( beginDialPars);
           end;
         {+ ABDET +}
         Db := true;
         Danyb := true;
         Dtb := false;
         Dabp := false;
         {+ NOTAVNXT +}
         Dbr := false;
         {+ COUNTRB +}
         DwrollbackCnf := false;
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { completion of rollback for whole TPPM subtree }
  from STATE_23_2
    when synchro.rollbackComplete
      provided (Danyb)
        to STATE_1
          begin
            {+ DELBR +}
            Dl := false;
            DrollbackDone := true;
          end;

  from STATE_23_2
    when synchro.rollbackComplete
      provided (not Danyb and Dc)
        to STATE_2
          var beginPars: cBeginReqType;
         begin
           {+ CMPRBSB +}
           if (Dbr) then
             begin {+ CBErq +}
               Dzb := newBranchId( Nzt);
               beginPars.atomicActionId := Nzt;
               beginPars.atomicBranchId := Dzb;
               output af.C_BEGIN_req( beginPars);
               addSubordinateBranchRecord(Nzt,Dzb,Dhp,Dzhp);
             end
           else
             Dl := false; {+ DELBR +}
           {+ RESETD +}
           De := false;
           Dg := false;
           Ddef := false;
           Denb := 0;
           Depnb := 0;
           if (Du) then Dbcr := false;
           peerCntrl := Dsh;
           DrollbackDone := true;
           {+ OPSAP +}
           sao.locked := false;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  from STATE_23_2
    when synchro.rollbackComplete
      provided (not Danyb and not Dc)
       to STATE_3
         var beginPars: cBeginReqType;
         begin
           {+ OPSAP +}
           sao.locked := false;
           {+ CMPRBSB +}
           if (Dbr) then
             begin {+ CBErq +}
               Dzb := newBranchId( Nzt);
               beginPars.atomicActionId := Nzt;
               beginPars.atomicBranchId := Dzb;
               output af.C_BEGIN_req( beginPars);
               addSubordinateBranchRecord(Nzt,Dzb,Dhp,Dzhp);
             end
           else
             Dl := false; {+ DELBR +}
           {+ RESETD +}
           Dg := false;
           De := false;
           Ddef := false;
           Denb := 0;
           Depnb := 0;
           if (Du) then Dbcr := false;
           peerCntrl := Dsh;
           DrollbackDone := true;
         end;

{*** DISCARDED Indications during rollback ********************************}

 trans
   from STATE_23_1
     when af.AF_U_ERROR_ind
       begin
       end;

     when af.AF_U_ERROR_cnf
       begin
       end;

     when af.AF_GRANT_CONTROL_ind
       begin
       end;

     when af.AF_REQUEST_CONTROL_ind
       begin
       end;

     when af.AF_HANDSHAKE_ind
       begin
       end;

     when af.AF_HANDSHAKE_GC_ind
       begin
       end;

     when af.U_ASE_ind
       begin
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{  USER ABORTS DURING TERMINATION PHASE                                   }
{                                                                         }
{*************************************************************************}

trans
  when tpsu.TP_U_ABORT_req
    from STATE_21_1, STATE_21_2, STATE_21_3, STATE_21_4
      provided (not Danyb)
        var abortPars: afAbortReqType;
        begin
          {+ ABTPSUI +}
          Danyb := true;
          Dtb := true;
          {+ RBNEXT +}
          if (Dbr) then DrollbackNext := true;
        end;

    from STATE_23_1
      provided (not Danyb)
        begin
          {+ ABTPSUI +}
          DtbDiag := param.userData;
          Dtb := true;
          Danyb := true;
          {+ NOTAVNXT +}
          Dbr := false;
        end;

    from STATE_23_2
      provided (not Danyb)
        var abortPars: afAbortReqType;
        begin
          {+ NOTAVNXT +}
          Dbr := false;
          {+ AABrqUd +}
          abortPars.aType := userAbort;
          abortPars.mapping := dataRI;
          output af.AF_ABORT_req( abortPars);
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ OPSAP +}
          sao_locked := false;
          {+ ABDET +}
          Danyb := true;
          Db := true;
          Dtb := false;
          Dabp := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{ Recovery                                                                }
{                                                                         }
{  Note: cases of double chanels are handled in MACF                      }
{                                                                         }
{*************************************************************************}

trans
  from STATE_99
    provided retryRecovery
      delay( recoverDelay )
        var cplPars: cafPleaseType;
        begin
          output cpm.CAF_PLEASE_req( cplPars)
        end;

trans
  from STATE_99
    when cpm.CAF_GIVE_ind
      to STATE_21_1
        var tokPars: afTokenGiveReqType;
            afRecPars: afRecoverReqType;
            cRecPars: cRecoverReqType;
        begin
          Dchat := true;
          {+ RCVRCOM +}
          if (Dhp) then
            begin
              afRecPars.mapping := recoverComRI;
              afRecPars.recoveryContextHandle := Dzhp;
              afRecPars.atomicActionId := Nzt;
              afRecPars.branchId := Dzb;
              output cpm.AF_RECOVER_req( afRecPars);
            end
          else
            begin
              cRecPars.recoveryState := commitState;
              cRecPars.atomicActionId := Nzt;
              cRecPars.branchId := Dzb;
              output cpm.C_RECOVER_req( cRecPars);
            end;
          output cpm.AF_TOKEN_GIVE_req(tokPars); { CPM will check Dtwr&Atok}
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {* rollback resulting from subordinate dialogue failure *}
  when cpm.C_RECOVER_ind
    provided (param.recoveryState=readyState)
      from STATE_15, STATE_20_1, STATE_17, STATE_20_2
        to STATE_23_2
           var tpAbortPars: tpPAbortIndType;
               afAbortPars: afAbortReqType;
               heurPars: tpHeuristicReportIndType;
               recPars: cRecoverReqType;
               cdet: cafDetachType;
           begin
             {+ TPABiFr +}
             tpAbortPars.diagnostic := ab_permanentFailure;
             tpAbortPars.rollback := true;
             output tpsu.TP_P_ABORT_ind( tpAbortPars);
             {+ THRiH +}
             heurPars.heuristicReport := heuristicHazard;
             output tpsu.TP_HEURISTIC_REPORT_ind(heurPars);
             {+ ABDET +}
             Db := true;
             Danyb := true;
             Dtb := false;
             Dabp := false;
             {+ INITRB - COUNTRB +}
             DwrollbackCnf := false;
             Drolledback := true;
             {+ NOTAVNXT +}
             Dbr := false;
             {+ AABrqPa +}
             afAbortPars.aType := providerAbort;
             afAbortPars.mapping := abortRI;
             output af.AF_ABORT_req( afAbortPars);
             {+ CRErsU +}
             recPars.recoveryState := unknownState;
             recPars.atomicActionId := Nzt;
             recPars.branchId := Dzb;
             output cpm.C_RECOVER_req(recPars);
             {+ CDETrq +}
             cdet.status := free_channel;
             output cpm.CAF_DETACH_req( cdet);
             {+ VDchatF +}
             Dchat := false;
             {+ SETTOKX +}
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {* recovery in ready state *}
  when cpm.C_RECOVER_ind
    provided (param.recoveryState=readyState)
      from STATE_20_3
        to STATE_20_3
           var tpAbortPars: tpPAbortIndType;
               afAbortPars: afAbortReqType;
               recPars : cRecoverRspType;
               cdet: cafDetachType;
        begin
          {+ TPABiPF if not Danyb +}
          if (not Danyb) then
            begin
              tpAbortPars.diagnostic := ab_permanentFailure;
              tpAbortPars.rollback := false;
              output tpsu.TP_P_ABORT_ind( tpAbortPars);
            end;
          {+ AABrqPa if not Db +}
          if (not Db) then
            begin
              afAbortPars.aType := providerAbort;
              afAbortPars.mapping := abortRI;
              output af.AF_ABORT_req( afAbortPars);
            end;
          if (wantToWaitForOutcome) then
            begin
              recPars.recoveryState := retryLater;
              recPars.atomicActionId := Nzt;
              recPars.branchId := Dzb;
              output cpm.C_RECOVER_rsp( recPars);
              cdet.status := free_channel;
              output cpm.CAF_DETACH_req( cdet);
            end
          else
            Dchat := true;
          {+ ABDET +}
          Db := true;
          Danyb := true;
          Dtb := false;
          Dabp := false;
          {+ see macf for double channel situation +}
          {+ see cpm for Vatokx settings +}
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans  {* recovery in decided commit state *}
  when cpm.C_RECOVER_ind
    provided (param.recoveryState=readyState)
      from STATE_21_1, STATE_21_2, STATE_99
        to STATE_21_1
          var tpAbortPars: tpPAbortIndType;
              recPars: cRecoverReqType;
              afAbortPars: afAbortReqType;
          begin
            if (not Danyb) then
              begin
                {+ TPABiPF +}
                tpAbortPars.diagnostic := ab_permanentFailure;
                tpAbortPars.rollback := false;
                output tpsu.TP_P_ABORT_ind( tpAbortPars);
              end;
            if (not Db) then
              begin
                {+ AABrqPa +}
                afAbortPars.aType := providerAbort;
                afAbortPars.mapping := abortRI;
                output af.AF_ABORT_req( afAbortPars);
              end;
            {+ ABDET +}
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
            {+ VDchatT +}
            Dchat := true;
            {+ RCVRCOM - see cpm as well +}
            recPars.recoveryState := commitState;
            recPars.atomicActionId := Nzt;
            recPars.branchId := Dzb;
            output cpm.C_RECOVER_req( recPars);
            {+ RBNEXT if Dbr +}
            if (Dbr) then DrollbackNext := true;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when cpm.C_RECOVER_cnf
    provided (param.recoveryState=doneState)
      from STATE_21_1
        to STATE_21_3
          var cdet: cafDetachType;
          begin
            {+ COUNTCOM +}
            DCmtCnfRcvd := true;
            {+ CDETreq +}
            Dchat := false;
            cdet.status := free_channel;
            output cpm.CAF_DETACH_req( cdet);
          end;

    provided (param.recoveryState=retryLater)
      from STATE_21_1
        to STATE_99
          var cdet: cafDetachType;
          begin
            retryRecovery := true;
            {+ CDETrq +}
            Dchat := false;
            cdet.status := free_channel;
            output cpm.CAF_DETACH_req( cdet)
          end;

  when cpm.AF_HEURISTIC_ind
    provided (param.mapping=recoverDoneRC)
      from STATE_21_1
        to STATE_21_3
          var heurPars: tpheuristicReportIndType;
              cdet: cafDetachType;
          begin
            {+ THRi +}
            heurPars.heuristicReport := param.heuristicReport;
            output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
            {+ COUNTCOM +}
            DcmtCnfRcvd := true;
            {+ CDETrq +}
            Dchat := false;
            cdet.status := free_channel;
            output cpm.CAF_DETACH_req( cdet)
          end;

trans  {* recovery in decided rollback state *}
  when cpm.C_RECOVER_ind
    provided (param.recoveryState=readyState)
      from STATE_23_1, STATE_23_2
        to STATE_23_2
          var tpAbortPars: tpPAbortIndType;
              afAbortPars: afAbortReqType;
              recPars: cRecoverReqType;
              cdet: cafDetachType;
              heurPars: tpHeuristicReportIndType;
          begin
            if (not Danyb) then
              begin
                {+ TPABiPF +}
                tpAbortPars.diagnostic := ab_permanentFailure;
                tpAbortPars.rollback := false;
                output tpsu.TP_P_ABORT_ind( tpAbortPars);
              end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
            if (not Db) then
              begin
                {+ AABrqPa +}
                afAbortPars.aType := providerAbort;
                afAbortPars.mapping := abortRI;
                output af.AF_ABORT_req( afAbortPars);
              end;
            {+ RCVRCOM - see cpm as well +}
            recPars.recoveryState := unknownState;
            recPars.atomicActionId := Nzt;
            recPars.branchId := Dzb;
            output cpm.C_RECOVER_req( recPars);
            {+ CDETrq +}
            cdet.status := free_channel;
            output cpm.CAF_DETACH_req( cdet);
            Dchat := false;
            {+ NOTAVNXT +}
            Dbr := false;
            {+ COUNTRB or THRi +}
            if (DwrollbackCnf) then
              begin
                output tpsu.TP_HEURISTIC_REPORT_ind( heurPars);
                DwrollbackCnf := false;
              end;
            {+ ABDET +}
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
          end;

 trans {* aborts on a channel - aborts received on cpm interaction point *}
   when cpm.AF_ABORT_ind
     provided (param.mapping=abortRI)
      from STATE_21_1, STATE_21_2
        to STATE_99
          var cdet: cafDetachType;
              cplPars: cafPleaseType;
          begin
            Dchat := false;
            cdet.status := free_channel;
            output cpm.CAF_DETACH_req(cdet);
            output cpm.CAF_PLEASE_req(cplPars);
          end;

     from STATE_20_3
       to STATE_20_3
         var cdet: cafDetachType;
         begin
           Db := true;
           Danyb := true;
           Dchat := false;
           cdet.status := free_channel;
           output cpm.CAF_DETACH_req(cdet);
         end;

#include 'dialcom.e'

end; { body for subordinate dialogues }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*****  dialsup.e: behavior specific to superior dialogues **}

 body dial_sup_b for dial_m;  { module body for superior  dialogues }

  state
   STATE_1,    {* new dialogue - initial state *}
   STATE_1_1,  {* C_BEGIN_ind expected on dialogue establishment  *}
   STATE_2,    {* Data transfer - shared or polarized cntrl - cntrl here *}
   STATE_3,    {* Data transfer - polarized control without control. *}
   STATE_4,    {* AF_U_ERROR_req sent *}
   STATE_5,    {* AF_U_ERROR_in received. *}
   STATE_11,   {* AF_END_DIALOGUE(confirmed)_req sent *}
   STATE_12,   {* AF_END_DIALOGUE(confirmed)_ind received *}
   STATE_6,    {* AF_HANDSHAKE_req sent *}
   STATE_7,    {* AF_HANDSHAKE_ind received *}
   STATE_8,    {* AF_HANDSHAKE_ind/req collision *}
   STATE_9,    {* AF_END_DIALOGUE_ind received - handshake req is pending *}
   STATE_10,   {* AF_HANSHAKE_ind received - end dialogue req is pending *}
   STATE_13,   {* AF_HANDSHAKE_&_GRANT_CONTROL_req sent *}
   STATE_14,   {* AF_HANDSHAKE_AND_GRANT_CONTROL_ind received *}
   STATE_15,   {* not used by superior dialogue *}
   STATE_20_1,{* not used by superior dialogue *}
   STATE_18,   {* C_PREPARE_ind received from parent *}
   STATE_20_2,{* waiting for C_READY_ind on all subordinate dialogues *}
   STATE_20_3,{* READY state - commit indication expected. *}
   STATE_21_5,{* waiting for commit confirm from all subordinates *}
   STATE_21_6,{* waiting for commit conf from subordinates - rb pending *}
   STATE_99,   {* waiting for channel establishment *}
   STATE_23_3,{* waiting to report rollback to superior *}
   STATE_23_4,{* waiting to confirm rollback with superior *}
   STATE_23_5,{* rollback request sent to sup - confirm expected *}
   STATE_23_6,{* report to superior done - waiting for next trans *}
   STATE_23_7,{* report to superior done - no next trans *}
   STATE_23_8,{* rollback not complete in subtree - superior gone *}
   WAIT_FOR_RESTART; {* initial state after following node crash *}

   var
     DtbDiag: userDataType;
     savedBDInd: afBeginDialIndType;
     peerCntrl: boolean;
     noPars: emptyParameterList;
     beginPars: cBeginRspType;
     beginDialPars: afBeginDialRspType;
     errPars: afUErrorRspType;
     retryRecovery: boolean;
     detPars: safDetachAssocReqType;

#include 'sao.e'

   modvar
     sao: sao_m;

   ip
     af: AF_Service( user) common queue;
     saf: SAF_Service( user) common queue;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{ Superior Dialogue Initialization                                        }
{*************************************************************************}
  initialize { normal case }
    provided not nodeCrash
      to STATE_1
        var nullBeginDialPars: afBeginDialReqType;
        begin
          Ds := true;
          Da := false;
          Dabp := false;
          Dah := false;
          Danyb := false;
          Db := false;
          Dbcr := false;
          Dc := false;
          Dchat := false;
          Dcr := false;
          De := false;
          Denb := 0;
          Depnb := 0;
          Dg := false;
          Dps := false;
          Drvyp := false;
          Dtb := false;
          dead := false;
          Drolledback := false;
          DrollbackNext := false;
          inTrans := false;
          DcommitDone := false;
          init sao with sao_b( true, assocId, nullBeginDialPars, true);
          connect sao.af to af;
        end;

initialize { creation after node crash }
  provided nodeCrash
    to WAIT_FOR_RESTART
      begin
        inTrans := true;
        Dl := true;
        Ds := true;
        Db := true;
        Danyb := true;
        Dchat := false;
        Dzb := branchId;
        Nzt := actionId;
        recoveryContextHndlInTransLog(Nzt,Dhb,Dzhb);
        Dtb := false;
        Dabp := false;
        dead := false;
        Drolledback := false;
        DrollbackNext := false;
        DcommitDone := false;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  from WAIT_FOR_RESTART
    when synchro.restartInReady
      to STATE_99
        var abortPars: tpPAbortIndType;
            cplReq: cafPleaseType;
        begin
          DcmtIndRcvd := false;
          {+ TPABiPF +}
          abortPars.rollback := false;
          abortPars.diagnostic := ab_permanentFailure;
          output tpsu.TP_P_ABORT_ind( abortPars);
          {+ CPLrq +}
          output cpm.CAF_PLEASE_req( cplreq);
        end;

    when synchro.restartInCommit
      to STATE_21_5
        begin
        end;

{*************************************************************************}
{ Superior Dialogue Establishment                                         }
{*************************************************************************}
trans
  when af.AF_BEGIN_DIALOGUE_ind
    from STATE_1
      provided rejectDialInd( param)
        var afBeginDialPars: afBeginDialRspType;
        begin
          {+ ABDrsRPd +}
          afBeginDialPars.result := rejected_p;
          afBeginDialPars.mapping := dataRI;
          output af.AF_BEGIN_DIALOGUE_rsp( afBeginDialPars);
          if (beginTransactionInd(param)) then
            detPars.status := c_begin_expected
          else if (Du)  then
            detPars.status := begin_fear
          else
            detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
        end;

      provided not rejectDialInd(param) and not beginTransactionInd( param)
          and not sharedFuncUnitBDInd( param)
        to STATE_3
          var dialPars: tpBeginDialogueIndType;
          begin
            {+ INITDIASP +}
            savedBDInd := param;
            Dsh := false;
            Dc := false;
            Dh := handshakeFuncUnitBDInd( param);
            Dco := commitFuncUnitBDInd( param);
            Du := unchainedFuncUnitBDInd( param);
            Dl := false;
            Dcr := param.confirmation=always;
            {+ TBDi +}
            mapBeginDialIndAFtoTP( param, dialPars);
            output tpsu.TP_BEGIN_DIALOGUE_ind( dialPars);
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    provided not rejectDialInd(param) and not beginTransactionInd(param)
             and sharedFuncUnitBDInd( param)
      to STATE_2
         var dialPars: tpBeginDialogueIndType;
         begin
           {+ INIDIASP +}
           peerCntrl := true;
           savedBDInd := param;
           Dsh := true;
           Dc := true;
           Dh := handshakeFuncUnitBDInd( param);
           Dco := commitFuncUnitBDInd( param);
           Du := unchainedFuncUnitBDInd( param);
           Dl := false;
           Dcr := param.confirmation=always;
           {+ TBDi +}
           mapBeginDialIndAFtoTP( param, dialPars);
           output tpsu.TP_BEGIN_DIALOGUE_ind( dialPars);
         end;

    provided not rejectDialInd(param) and beginTransactionInd(param)
      to STATE_1_1
         begin
           {+ INITDIASP +}
           savedBDInd := param;
           Ds := true;
           Dsh := sharedFuncUnitBDInd( param);
           Dc := Dsh;
           Dh := handshakeFuncUnitBDInd(param);
           Dco := commitFuncUnitBDInd( param);
           Du := unchainedFuncUnitBDInd(param);
           Dcr := param.confirmation=always;
         end;

when af.C_BEGIN_ind
  from STATE_1_1
    provided Dsh
      to STATE_2
         var dialPars: tpBeginDialogueIndType;
         begin
           {+ ADDBRSP +}
           Dl := true;
           peerCntrl := true;
           Nzt := param.atomicActionId;
           Dzb := param.atomicBranchId;
           Dbr := not Du;
           if (sao.Ahr) then
             begin
               Dhb := true;
               Dzhb := sao.Aahr;
             end;
           {+ TBDi +}
           mapBeginDialIndAFtoTP( savedBDInd, dialPars);
           output tpsu.TP_BEGIN_DIALOGUE_ind( dialPars);
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when af.C_BEGIN_ind
    from STATE_1_1
      provided not Dsh
        to STATE_3
          var dialPars: tpBeginDialogueIndType;
          begin
            {+ ADDBRSP +}
            Dl := true;
            peerCntrl := true;
            Nzt := param.atomicActionId;
            Dzb := param.atomicBranchId;
            Dbr := not Du;
            if (sao.Ahr) then
              begin
                Dhb := true;
                Dzhb := sao.Aahr;
              end;
            {+ TBDi +}
            mapBeginDialIndAFtoTP( savedBDInd, dialPars);
            output tpsu.TP_BEGIN_DIALOGUE_ind( dialPars);
          end;

 trans
   when tpsu.TP_BEGIN_DIALOGUE_rsp
     provided ((param.result=accepted) and Dcr)
       from STATE_2, STATE_3, STATE_5, STATE_7, STATE_14, STATE_18
         begin
#include 'deldlg.e'
         end;

   when tpsu.TP_BEGIN_DIALOGUE_rsp
     provided ((param.result<>accepted) and not Da)
       from STATE_2, STATE_3, STATE_5, STATE_7, STATE_12, STATE_14, STATE_18
         to STATE_1
           var beginDialPars: afBeginDialRspType;
           begin
             Dcr := false;
             if (not Dl) then
               begin
                 if (Du) then
                   detPars.status := begin_fear
                 else
                   detPars.status := free_assoc;
                 beginDialPars.mapping := dataRI
               end
             else
               begin
                 Dl := false;
                 detPars.status := rollback_cnf_expected;
                 beginDialPars.mapping := rollbackRI
               end;
             beginDialPars.result := rejected_u;
             {+ ABDrsRUd  or ABDrsUr +}
             output af.AF_BEGIN_DIALOGUE_rsp( beginDialPars);
             {+ SETFEAR +}
             sao.Abf := not Dl and Du;
             {+ SDETrqF or SDETrqRBC +}
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { completion of dialogue establishement during rollback - }
      { in this state, (^Da & Dcr) can only be true for leaves  }
  from STATE_23_4
    when tpsu.TP_BEGIN_DIALOGUE_rsp
      provided (Dcr and (param.result=accepted))
        begin
          Dcr := false;
          Dah := true;
        end;

      provided (not Da and (param.result<>accepted))
        to STATE_1
          var beginPars : afBeginDialRspType;
          begin
            {+ ABDrsRUrbc +}
            Dcr := false;
            beginPars.result := rejected_u;
            beginPars.mapping := rollbackRC;
            output af.AF_BEGIN_DIALOGUE_rsp( beginPars);
            {+ DELBR - see macf +}
            Dl := false;
            {+ SDETrqF +}
            detPars.status := free_assoc;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
          end;

{*************************************************************************}
{ Deferred services                                                       }
{*************************************************************************}

trans
  when af.AF_DEFER_ind
    from STATE_2, STATE_3, STATE_4, STATE_6
      provided (Dl and (param.deferType=endDialogue) and not De and
                (Dsh or peerCntrl))
        begin
          {+ VdeT +}
          De := true;
          {+ TDEi +}
          output tpsu.TP_DEFERRED_END_DIALOGUE_ind( noPars)
        end;

    from STATE_3, STATE_4
      provided (not De and (param.deferType=grantControl) and not Dg and
                not Dsh)
          begin
            {+ VDgT +}
            Dg := true;
            {+ TDGi +}
            output tpsu.TP_DEFERRED_GC_ind( noPars)
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{ Raising Dialogue level (unchained functional unit)                      }
{*************************************************************************}
 trans
   when af.C_BEGIN_ind
     provided (Du and not inTrans and not Dl)
       from STATE_2, STATE_3, STATE_4, STATE_6
           begin
             {+ ADDBRSP +}
             Dl := true;
             if (Da) then output af.C_BEGIN_rsp( noPars);
             if (sao.Ahr) then
               begin
                 Dhb := true;
                 Dzhb := sao.Aahr;
               end;
             Nzt := param.atomicActionId;
             Dzb := param.atomicBranchId;
             output tpsu.TP_BEGIN_TRANSACTION_ind( noPars);
           end;

     provided (not Dl and Du and not inTrans and (Dsh or peerCntrl))
       from STATE_2, STATE_3, STATE_4, STATE_6
         to STATE_1
           var afAbortPars: afAbortReqType;
               tpAbortPars: tpPAbortIndType;
           begin
             {+ TPABiBTR +}
             tpAbortPars.rollback := true;
             tpAbortPars.diagnostic := ab_beginTransactionReject;
             output tpsu.TP_P_ABORT_ind( tpAbortPars);
             {+ AABrqPrTR +}
             afAbortPars.aType := providerAbort;
             afAbortPars.diagnostic := ab_beginTransactionReject;
             afAbortPars.mapping := rollbackRI;
             output af.AF_ABORT_req( afAbortPars);
             {+ SDETrqRBC +}
             detPars.status := rollback_cnf_expected;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
           end;

     provided (Dsh and Du)
       from STATE_11
         to STATE_1
           var tpAbortPars: tpPAbortIndType;
               abortPars: afAbortReqType;
           begin
             {+ TPABiBTED +}
             tpAbortPars.rollback := true;
             tpAbortPars.diagnostic := ab_beginTransactionEndDialCollision;
             output tpsu.TP_P_ABORT_ind( tpAbortPars);
             {+ CRBrq +}
             abortPars.mapping := rollbackRI;
             abortPars.diagnostic := ab_beginTransactionReject;
             output af.AF_ABORT_req( abortPars);
             {+ SDETrqRBC +}
             detPars.status := rollback_cnf_expected;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
             Dl := false;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{ COMMITMENT - Phase 1 - Node in active/ready state                       }
{*************************************************************************}

{*** preparing for commitment in subtree *********************************}

 trans
   when af.AF_PREPARE_ind
     from STATE_2, STATE_3
       provided (D1 and (Dsh or peerCntrl))
         to STATE_18
           begin
             {+ TPRi +}
             output tpsu.TP_PREPARE_ind( param);
             {+ VNpT see macf +}
             {+ VDdp(T/F) +}
             Ddp := Dsh or param.dataPermitted;
           end;

     from STATE_4, STATE_6
       provided (D1)
         to STATE_23_3
           begin
             {+ TRBi +}
             output tpsu.TP_ROLLBACK_ind( noPars);
             {+ INITRB +}
             Drolledback := true;
             DwrollbackCnf := true;
           end;

 trans
   when tpsu.TP_DATA_req
     from STATE_18
       provided ((Dsh or Ddp) and not Dcr)
         begin
#include 'deldlg.e'
           {+ UASErq +}
           output af.U_ASE_req( param);
         end;

 trans { commitment order from TPSU }
   from STATE_18
     when tpsu.TP_COMMIT_req
       to STATE_20_2
         begin
         end;

 trans { going to ready state - waiting for phase 2 }
   when synchro.sendReady
     from STATE_20_2
       to STATE_20_3
         var tokPars: pTokenGiveReqType;
         begin
           DcmtIndRcvd := false;
#include 'deldlg.e'
           {+ PTOKGrq +}
           if (hasToken(assocId)) then
              output af.P_TOKEN_GIVE_req( tokPars);
           {+ CRYrq +}
           output af.C_READY_req( noPars);
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*** Provider abort in READY state - recovery is needed ***************}

 trans
  when af.AF_ABORT_ind
    from STATE_20_3
      provided (param.mapping=abortRI)
        to STATE_99
          var abortPars: tpPAbortIndType;
              cplPars: cafPleaseType;
          begin
            if (not Danyb) then
              begin
                {+ TPABiPF +}
                abortPars.diagnostic := param.diagnostic;
                abortPars.rollback := false;
                output tpsu.TP_P_ABORT_ind( abortPars);
              end;
            {+ VDchatF +}
            Dchat := false;
            {+ CPLrq +}
            output cpm.CAF_PLEASE_req(cplPars);
            {+ ABDET +}
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
          end;

{*** commitment order from superior TPPM - to phase 2  ********}

 trans
  from STATE_20_3
    when af.C_COMMIT_ind
      provided (not Dbr or De)
        to STATE_21_5
           begin
             {+ RECCOM - see macf as well +}
             DcmtIndRcvd := true;
             Nztn := newAtomicActionId;
             Dzbn := newBranchId( Nztn);
             Dbr := false;
           end;

  from STATE_20_3
    when af.C_COMMIT_C_BEGIN_ind
      provided (Dbr and not De)
        to STATE_21_5
           begin
             {+ RECCOM see macf as well +}
             DcmtIndRcvd := true;
             Nztn := param.atomicActionId;
             Dzbn := param.atomicBranchId;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  from STATE_20_3
   when af.AF_ABORT_ind
     provided ((param.aType=userAbort) and (param.mapping=commitRI)
               and (not Danyb or not Dabp))
       to STATE_21_5
          var abortPars: tpUAbortIndType;
          begin
            {+ RECCOM - see macf as well +}
            DcmtIndRcvd := true;
            if (not Danyb) then
              begin
                {+ TUABi +}
                abortPars.rollback := false;
                output tpsu.TP_U_ABORT_ind( abortPars)
              end;
            Nztn := newAtomicActionId;
            Dzbn := newBranchId( Nztn);
            {+ ABPTNR +}
            Dabp := true;
            Danyb := true;
            Dtb := false;
          end;

{*** starting rollback in subtree ************************************}

 trans { Rollback initiated by local/superior TPSU }
   when tpsu.TP_ROLLBACK_req
     from STATE_2, STATE_3, STATE_4, STATE_5,
        STATE_6, STATE_7, STATE_8, STATE_13, STATE_14, STATE_18
       provided (D1 and not Dcr)
         to STATE_23_3
           begin
             Da := true;
             Drolledback := true;
             DwrollbackCnf := true;
           end;

   when af.C_ROLLBACK_ind
     from STATE_2, STATE_3, STATE_4, STATE_5,
          STATE_6, STATE_7, STATE_8, STATE_13, STATE_14
       provided (D1)
         to STATE_23_4
           begin
             {+ TRBi +}
             output tpsu.TP_ROLLBACK_ind( noPars);
             {+ INITRB +}
             Drolledback := true;
             DwrollbackCnf := false;
           end;

     from STATE_18, STATE_20_2, STATE_20_3
       provided (D1)
         to STATE_23_4
           begin
             {+ TRBi +}
             output tpsu.TP_ROLLBACK_ind( noPars);
             {+ LOGDAMRB see macf +}
             {+ INITRB +}
             Drolledback := true;
             DwrollbackCnf := false;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
 trans { Rollback resulting from rollback on a subordinate dialogue }
   when synchro.rollbackAll
     from STATE_2, STATE_3, STATE_4, STATE_5, STATE_6, STATE_7, STATE_8,
          STATE_13, STATE_14, STATE_18, STATE_20_2, STATE_20_3
       to STATE_23_3
         begin
         end;

     from STATE_23_3, STATE_23_4, STATE_23_8
       begin
       end;

 trans { rollback of subtree trigerred by superior dial abort }
   when tpsu.TP_U_ABORT_req
     from STATE_2, STATE_3, STATE_4, STATE_5,
          STATE_6, STATE_7, STATE_8, STATE_13, STATE_14, STATE_18
       provided (Dl and not Dcr)
         to STATE_23_3
           begin
#include 'deldlg.e'
             {+ ABTPSUI +}
             Danyb := true;
             Dtb := true;
             DtbDiag := param.userData;
             {+ NOTAVNXT +}
             Dbr := false;
             {+ INITRB  and rollback all +}
             Drolledback := true;
             DwRollbackCnf := true;
           end;

   when af.AF_ABORT_ind
     provided (Dl and not Dcr and
               (param.aType=userAbort) and (param.mapping=rollbackRI))
       from STATE_2, STATE_3, STATE_4, STATE_5, STATE_6, STATE_7, STATE_8,
            STATE_13, STATE_14
         to STATE_23_4
            var uAbortPars: tpUAbortIndType;
            begin
              {+ TUABiR +}
              uAbortPars.rollback := true;
              output tpsu.TP_U_ABORT_ind( uAbortPars);
              {+ ABPTNR +}
              Dtb := false;
              Dabp := true;
              Danyb := true;
              {+ NOTAVNXT +}
              Dbr := false;
              {+ INITRB +}
              Drolledback := true;
              DwrollbackCnf := false;
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.AF_ABORT_ind
  provided (param.mapping=rollbackRI)
    from STATE_18, STATE_20_2, STATE_20_3
      to STATE_23_4
         var uAbortPars: tpUAbortIndType;
         begin
           if (not Danyb) then
             begin {+ TUABiR +}
               uAbortPars.rollback := true;
               output tpsu.TP_U_ABORT_ind( uAbortPars)
             end
           else output tpsu.TP_ROLLBACK_ind( noPars);
           {+ LOGDAMRB - see macf +}{+ INITRB +}
           Drolledback := true;
           DwrollbackCnf := false;
           {+ ABPTNR +}
           Dabp := true;
           Danyb := true;
           {+ NOTAVNXT +}
           Dbr := false;
         end;

when af.AF_ABORT_ind
  provided (D1 and (param.mapping=abortRI) and Dcr)
    from STATE_2, STATE_3, STATE_5
      to STATE_1
        var pAbortPars: tpPAbortIndType;
        begin
          pAbortPars.rollback := false;
          pAbortPars.diagnostic := param.diagnostic;
          output tpsu.TP_P_ABORT_ind( pAbortPars);
        end;

  provided (D1 and (param.mapping=abortRI) and not Dcr)
    from STATE_2, STATE_3, STATE_4, STATE_5, STATE_6, STATE_7, STATE_8,
         STATE_13, STATE_14, STATE_18, STATE_20_2
      to STATE_23_8
         var pAbortPars: tpPAbortIndType;
             crbPars : cRollbackRspType;
         begin
           {+ TPABiPPr +}
           pAbortPars.rollback := true;
           pAbortPars.diagnostic := param.diagnostic;
           output tpsu.TP_P_ABORT_ind( pAbortPars);
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
           {+ NOTAVNXT +}
           Dbr := false;
           {+ INITRB +}
           Drolledback := true;
           DwRollbackCnf := false;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { leaf node case }
  when af.AF_ABORT_ind
    provided (D1 and Dcr and
              (param.aType=userAbort) and (param.mapping=rollbackRI))
      from STATE_2, STATE_3, STATE_5, STATE_7, STATE_14
        to STATE_1
           var uAbortPars: tpUAbortIndType;
               crbPars : cRollbackRspType;
           begin
             {+ TUABi +}
             uAbortPars.rollback := false;
             output tpsu.TP_U_ABORT_ind( uAbortPars);
             {+ CRBrs +}
             output af.C_ROLLBACK_rsp( crbPars);
             {+ SDETrqF +}
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             {+ DELBR +}
             D1 := false;
           end;

{*************************************************************************}
{  Commitment Termination - Phase 2 - Commitment of subtree                 }
{*************************************************************************}

trans { when next transaction may have to be rolled back }
  when synchro.rPend
    from STATE_21_5
      provided (not Db and Dbr)
        to STATE_21_6
          begin
          end;

    from STATE_21_5, STATE_99
      provided (Db or not Dbr)
        begin
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans  { no rollback of next trans }
  from STATE_21_5
    when synchro.commitComplete
      provided ((Danyb or De) and not Dchat)
        to STATE_1
          var abortPars: afAbortReqType;
              heurPars: afHeuristicReqType;
              abHeurPars: afAbortAndHeuristicReqType;
              cComPars: cCommitRspType;
        begin
          DcommitDone := true;
          if (not Db) then
            begin
              {+ COMRSP +}
              if (not Nhd) then
                if (Dtb and Du) then
                  begin
                    abortPars.userData := DtbDiag;
                    abortPars.aType := userAbort;
                    abortPars.mapping := commitRC;
                    output af.AF_ABORT_req( abortPars);
                  end
                else output af.C_COMMIT_rsp( cComPars)
              else
                if (Dtb and Du) then
                  begin
                    abHeurPars.heuristicReport := recordedDamage(Nzt);
                    abHeurPars.userData := DtbDiag;
                    abHeurPars.mapping := commitRC;
                    output af.AF_ABORT_HEURISTIC_req( abHeurPars);
                  end
              else
                  begin
                    heurPars.heuristicReport := recordedDamage(Nzt);
                    heurPars.mapping := commitRC;
                    output af.AF_HEURISTIC_req( heurPars);
                  end;
              if (Dtb) then
                begin
                  Dtb := false;
                  Dabp := true;
                end;
              {+ SDETrqF +}
              detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
            end;
          {+ DELBR +}
          Dl := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_21_5
  when synchro.commitComplete
    provided (Dchat)
      to STATE_1
        var heurPars: afHeuristicReqType;
            afRecPars: afRecoverReqType;
            cRecPars: cRecoverReqType;
            cafDetach: cafDetachType;
      begin
        {+ RECVRDONE +}
        if (Nhd) then
          begin
            heurPars.heuristicReport := recordedDamage(Nzt);
            heurPars.mapping := recoverDoneRC;
            heurPars.atomicActionId := Nzt;
            heurPars.branchId := Dzb;
            output cpm.AF_HEURISTIC_req( heurPars);
          end
        else
          begin
            cRecPars.recoveryState := doneState;
            cRecPars.atomicActionId := Nzt;
            cRecPars.branchId := Dzb;
            output cpm.C_RECOVER_rsp( cRecPars);
          end;
        {+ DELBR +}
        Dl := false;
        {+ CDETrq +}
        Dchat := false;
        cafDetach.status := free_channel;
        output cpm.CAF_DETACH_req(cafDetach);
        DcommitDone := true;
      end;

    provided (not Danyb and not De and (Dsh or not Dg) and not Dchat)
      to STATE_2
        var heurPars: afheuristicReqType;
            cComPars: cCommitRspType;
      begin
        {+ COMRSP +}
        if (not Nhd) then
          output af.C_COMMIT_rsp( cComPars)
        else
          begin
            heurPars.heuristicReport := recordedDamage(Nzt);
            heurPars.mapping := commitRC;
            output af.AF_HEURISTIC_req( heurPars);
          end;
        {+ CMPCOMSP +}
        Dc := not Dg;
        if (Dbr) then Dzb := Dzbn
        else Dl := false; {+ DELBR +}
        {+ RESETD +}
        Dg := false;
        Ddef := false;
        Denb := 0;
        Depnb := 0;
        if (Du) then Dbcr := false;
        peerCntrl := Dsh;
        DcommitDone := true;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    provided (Dg and not (Danyb or De or Dsh or Dchat))
      to STATE_3
        var heurPars: afheuristicReqType;
            cComPars: cCommitRspType;
      begin
        {+ COMRSP +}
        if (not Nhd) then
          output af.C_COMMIT_rsp( cComPars)
        else
          begin
            heurPars.heuristicReport := recordedDamage(Nzt);
            heurPars.mapping := commitRC;
            output af.AF_HEURISTIC_req( heurPars);
          end;
        {+ CMPCCOMSP +}
        Dc := not Dg;
        if (Dbr) then Dzb := Dzbn
        else Dl := false; {+ DELBR +}
        {+ RESETD +}
        Dg := false;
        Ddef := false;
        Denb := 0;
        Depnb := 0;
        if (Du) then Dbcr := false;
        peerCntrl := false;
        DcommitDone := true;
      end;


trans  { completion of commitment when next trans is rolled back }
  from STATE_21_6
    when synchro.commitComplete
      to STATE_23_3
        var heurPars: afheuristicReqType;
            cComPars: cCommitRspType;
        begin
          {+ COMRSP +}
          if (not Nhd) then
            output af.C_COMMIT_rsp( cComPars)
          else
            begin
              heurPars.heuristicReport := recordedDamage(Nzt);
              heurPars.mapping := commitRC;
              output af.AF_HEURISTIC_req( heurPars);
            end;
          {+ CMPCOMSP +}
          Dc := not Dg;
          if (Dbr) then Dzb := Dzbn;
          Drolledback := true;
          DwrollbackCnf := true;
          DcommitDone := true;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{  delayed aborts during commitment                                       }
{*************************************************************************}

 trans
   when af.AF_ABORT_ind
     from STATE_21_5, STATE_21_6
       provided (Dl and (param.mapping=abortRI))
         to STATE_21_5
           var abortPars: tpPAbortIndType;
           begin
             {+ TPABiPF if not Danyb +}
             if (not Danyb) then
               begin
                 abortPars.rollback := true;
                 abortPars.diagnostic := param.diagnostic;
                 output tpsu.TP_P_ABORT_ind( abortPars);
               end;
             {+ RBNEXT if Dbr +}
             if (Dbr) then DrollbackNext := true;
             {+ ABDET +}
             Db := true;
             Danyb := true;
             Dtb := false;
             Dabp := false;
           end;

 trans
   when tpsu.TP_U_ABORT_req
     from STATE_20_3, STATE_21_6
       provided (not Danyb)
         begin
           {+ ABTPSUI +}
           DtbDiag := param.userData;
           Dtb := true;
           Danyb := true;
           {+ NOTAVNXT +}
           Dbr := false;
         end;

     from STATE_21_5
       provided (not Danyb)
         begin
           {+ ABTPSUI +}
           DtbDiag := param.userData;
           Dtb := true;
           Danyb := true;
           {+ RBNEXT +}
           DrollbackNext := Dbr;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{ ROLLBACK OF SUBTREE                                                     }
{*************************************************************************}
 trans
   from STATE_23_3
     when synchro.reportRollback
       to STATE_23_5
         var abortPars: afAbortReqType;
             heurAbortPars: afAbortAndHeuristicReqType;
             heurPars: afheuristicReqType;
             crbPars: cRollbackRspType;
         begin   { sending rollback to superior }
           DwRollbackCnf := true;
           {+ RBRQ +}
           if (not Nhd and not Dtb) then
             output af.C_ROLLBACK_req( crbPars)
           else if (Nhd and not Dtb) then
             begin
               heurPars.mapping := rollbackRI;
               heurPars.heuristicReport := recordedDamage(Nzt);
               output af.AF_HEURISTIC_req( heurPars);
             end
           else if (not Nhd and Dtb) then
             begin
               abortPars.aType := userAbort;
               abortPars.userData := DtbDiag;
               abortPars.mapping := rollbackRI;
               output af.AF_ABORT_req( abortPars);
             end
           else if (Nhd and Dtb) then
             begin
               heurAbortPars.userData := DtbDiag;
               heurAbortPars.mapping := rollbackRI;
               heurPars.heuristicReport := recordedDamage(Nzt);
               output af.AF_ABORT_HEURISTIC_req( heurAbortPars);
             end;
           if (Dtb) then
             begin
               Dtb := false;
               Dabp := true;
             end;
            Da := true;
         end;

 trans { waiting for rollback confirm from superior TPPM }
   from STATE_23_5
     when af.C_ROLLBACK_ind   { collision on rollback - taken as confirm }
       provided (not Danyb and Dbr)
         to STATE_23_6
           var heurPars: afheuristicReqType;
               crbPars: cRollbackRspType;
           begin {+ RBRSP +}
             if (not Nhd) then
               output af.C_ROLLBACK_rsp( crbPars)
             else begin
                 heurPars.mapping := rollbackRC;
                 heurPars.heuristicReport := recordedDamage(Nzt);
                 output af.AF_HEURISTIC_req( heurPars);
               end;
             DwRollbackCnf := false;
             DwBeginInd := true;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_5
  when af.C_ROLLBACK_ind   { collision on rollback - taken as confirm }
    provided (not Dbr)
      to STATE_23_7
         var abortPars: afAbortReqType;
             heurAbortPars: afAbortAndHeuristicReqType;
             heurPars: afheuristicReqType;
             crbPars: cRollbackRspType;
         begin
           DwRollbackCnf := false;
           DwBeginInd := false;
           if (not Danyb) then
             begin
               {+ RBRSP +}
               if (not Nhd) then
                 output af.C_ROLLBACK_rsp( crbPars)
               else
                 begin
                   heurPars.mapping := rollbackRC;
                   heurPars.heuristicReport := recordedDamage(Nzt);
                   output af.AF_HEURISTIC_req( heurPars);
                 end
             end
           else if (Dtb or Dabp) then
             begin
               {+ AABrqUrbc +}
               abortPars.aType := userAbort;
               abortPars.userData := DtbDiag;
               abortPars.mapping := rollbackRC;
               output af.AF_ABORT_req( abortPars);
               {+ SDETrqF +}
               detPars.status := free_assoc;
               output saf.SAF_DETACH_ASSOCIATION_req( detPars);
               disconnect sao;
               {+ ABDET +}
               Danyb := true;
               Db := false;
               Dtb := false;
               Dabp := false;
             end;
         end;

from STATE_23_5
  when af.C_ROLLBACK_cnf
    provided (not Danyb and Dbr)
      to STATE_23_6
        begin
          DwRollbackCnf := false;
          DwBeginInd := true;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_5
  when af.C_ROLLBACK_cnf
    provided (not Dbr)
      to STATE_23_7
        var abortPars: afAbortReqType;
        begin
          if (Danyb) then
            begin
              if (Dtb) then
                begin
                  {+ AABrqUd +}
                  abortPars.userData := DtbDiag;
                  abortPars.mapping := dataRI;
                  abortPars.aType := userAbort;
                  output af.AF_ABORT_req( abortPars);
                end;
              {+ SDETrqF +}
              detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
              {+ ABDET +}
              Danyb := true;
              Db := true;
              Dabp := false;
              Dtb := false;
            end;
          DwRollbackCnf := false;
          DwBeginInd := false;
        end;

from STATE_23_5
  when af.AF_ABORT_ind
    provided ((param.mapping=rollbackRI) or (param.mapping=rollbackRC))
      to STATE_23_7
        var abortPars: tpUAbortIndType;
            crbPars: cRollbackRspType;
        begin
          {+ TUABi +}
          if (not Danyb) then
            begin
              abortPars.rollback := false;
              output tpsu.TP_U_ABORT_ind( abortPars);
            end;
          {+ NOTAVNXT +}
          Dbr := false;
          {+ RBRSP if necessary +}
          if (param.mapping=rollbackRI) then
            output af.C_ROLLBACK_rsp( crbPars);
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Danyb := true;
          Db := true;
          Dabp := false;
          Dtb := false;
          DwRollbackCnf := false;
          DwBeginInd := false;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans  { rollback initiated by sup before being completed in subtree }
  from STATE_23_3
    to STATE_23_4
      when af.C_ROLLBACK_ind
        begin
          DwRollbackCnf := false;
          Dbr := false;
        end;

  from STATE_23_3
    to STATE_23_4
      when af.AF_ABORT_ind
        provided ((param.aType=userAbort) and (param.mapping=rollbackRf))
          var abortPars: tpUAbortIndType;
          begin
            if (not Danyb) then
              begin
                {+ TUABi +}
                abortPars.rollback := false;
                output tpsu.TP_U_ABORT_ind( abortPars);
                {+ ABPTNR +}
                Dtb := false;
                Dabp := true;
                Danyb := true;
              end;
            {+ NOTAVNXT +}
            Dbr := false;
            DwRollbackCnf := false;
          end;

{*** Completing ROLLBACK initiated by superior TPPM  ********************}

trans
  when synchro.reportRollback
    from STATE_23_4
      provided (not Danyb and Dbr)
        to STATE_23_6
           var begDialRsp: afBeginDialRspType;
               heurPars: afheuristicReqType;
               crbPars: cRollbackRspType;
        begin
          if (Dah) then
            begin
              {+ ABDrsArbc +}
              begDialRsp.result := accepted;
              begDialRsp.mapping := rollbackRC;
              output af.AF_BEGIN_DIALOGUE_rsp( begDialRsp);
              Dah := false;
            end
          else if (not Nhd) then
            output af.C_ROLLBACK_rsp( crbPars) {+ RBSP +}
          else
            begin
              heurPars.mapping := rollbackRC;
              heurPars.heuristicReport := recordedDamage(Nzt);
              output af.AF_HEURISTIC_req( heurPars); {+ RBRTSP +}
            end;
          Da := true;
          DwBeginInd := true;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_23_4
  provided (not Dbr or Danyb)
    to STATE_23_4
      var abortPars: afAbortReqType;
          heurAbortPars: afAbortAndHeuristicReqType;
          heurPars: afheuristicReqType;
          crbRsp : cRollbackRspType;
      begin   { sending rollback to superior }
      {+ RBRSP +}
      if (not Nhd) then
        begin
          if (not Dtb) then
            begin
              output af.C_ROLLBACK_rsp( crbRsp);
            end
          else
            begin
              abortPars.aType := userAbort;
              abortPars.userData := DtbDiag;
              abortPars.mapping := rollbackRC;
              output af.AF_ABORT_req( abortPars);
              Dtb := false;
              Dabp := TRUE;
            end
        end
      else
        begin
          if (Dtb) then
            begin
              heurAbortPars.userData := DtbDiag;
              heurAbortPars.heuristicReport := recordedDamage(Nzt);
              heurAbortPars.mapping := rollbackRC;
              output af.AF_ABORT_HEURISTIC_req( heurAbortPars);
              Dtb := false;
              Dabp := TRUE;
            end
          else
            begin
              heurPars.heuristicReport := recordedDamage(Nzt);
              heurAbortPars.mapping := rollbackRC;
              output af.AF_ABORT_HEURISTIC_req( heurAbortPars);
            end
        end;
      if (Danyb) then
        begin
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
          {+ ABDET +}
          Dtb := false;
          Dabp := false;
          Db := true;
        end;
      Da := true;
      DwBeginInd := true;
    end;

from STATE_23_8
  to STATE_23_8
    begin
    end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when af.AF_ABORT_ind
    provided (param.mapping=abortRI)
      from STATE_23_3, STATE_23_4
        to STATE_23_8
          var abortPars: tpPAbortIndType;
          begin
            if (not Danyb) then
              begin
                {+ TPABiPF +}
                abortPars.diagnostic := param.diagnostic;
                output tpsu.TP_P_ABORT_ind( abortPars);
              end;
            {+ NOTAVNXT / ABDET +}
            Dbr := false;
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
            DwrollbackCnf := false;
          end;

    provided (param.mapping=abortRI)
      from STATE_23_5, STATE_23_6
        to STATE_23_7
          var abortPars: tpPAbortIndType;
          begin
            if (not Danyb) then
              begin
                {+ TPABiPF +}
                abortPars.diagnostic := param.diagnostic;
                output tpsu.TP_P_ABORT_ind( abortPars);
              end;
            {+ NOTAVNXT / ABDET +}
            Dbr := false;
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
            DwrollbackCnf := false;
          end;

    provided (Du and not Danyb and (param.mapping=dataRI))
      from STATE_23_6
        to STATE_23_7
          var abortPars: tpUAbortIndType;
              pAbortPars: tpPAbortIndType;
          begin
            if (param.aType=userAbort) then
              begin {+ TUABi +}
                abortPars.rollback := false;
                output tpsu.TP_U_ABORT_ind( abortPars);
              end;
            detPars.status := free_assoc;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
            {+ ABDET / NOTAVNXT +}
            Db := true;
            Danyb := true;
            Dbr := false;
            DwBeginInd := false;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when tpsu.TP_U_ABORT_req
    from STATE_23_3, STATE_23_4, STATE_23_5
      provided (not Danyb)
        begin
          {+ ABTPSUI +}
          DtbDiag := param.userData;
          Dtb := true;
          Danyb := true;
          {+ NOTAVNXT +}
          Dbr := false;
        end;

    from STATE_23_6
      provided (not Danyb)
        to STATE_23_7
          var abortPars: afAbortReqType;
          begin
            {+ AABrqUd +}
            abortPars.aType := userAbort;
            abortPars.mapping := dataRI;
            output af.AF_ABORT_req( abortPars);
            {+ SDETrqBF +}
            detPars.status := begin_fear;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
            {+ ABDET +}
            Danyb := true;
            Db := true;
            Dtb := false;
            Dabp := false;
          end;

    from STATE_23_7
      provided (not Danyb)
        to STATE_23_7
          var abortPars: afAbortReqType;
          begin
            {+ AABrqUd +}
            abortPars.aType := userAbort;
            if (Dbr) then
              begin
                abortPars.mapping := rollbackRI;
                detPars.status := rollback_cnf_expected;
              end
            else
              begin
                abortPars.mapping := dataRI;
                detPars.status := free_assoc
              end;
            output af.AF_ABORT_req( abortPars);
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
            sao.locked := false;
            {+ ABDET +}
            Danyb := true;
            Db := true;
            Dtb := false;
            Dabp := false;
            Dbr := false;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{** Entering next trans after rollback ***********************************}

 trans { entering next trans after rollback }
   from STATE_23_6
     when af.C_BEGIN_ind
       to STATE_23_7
         begin
           {+ CPSAP +}
           sao.locked := true;
           {+ CBEGAFTRB +}
           output af.C_BEGIN_rsp( noPars);
           Nzt := param.atomicActionId;
           Dzb := param.atomicBranchId;
           peerCntrl := Dsh;
           DwBeginInd := false;
         end;

 trans
   from STATE_23_4, STATE_23_5, STATE_23_7, STATE_23_8
     when synchro.rollbackComplete
       provided (Danyb)
         to STATE_1
           begin
             {+ DELBR +}
             Dl := false;
           end;

       provided (not Danyb and Dc)
         to STATE_2
           begin
             {+ OPSAP +}
             sao.locked := false;
             {+ DELBR +}
             Dl := Dbr;
             {+ RESETD +}
             De := false;
             Dg := false;
             Ddef := false;
             Denb := 0;
             Depnb := 0;
             if (Du) then Dbcr := false;
             peerCntrl := Dsh;
           end;

       provided (not Danyb and not Dc)
         to STATE_3
           begin
             {+ OPSAP +}
             sao.locked := false;
             {+ DELBR +}
             Dl := Dbr;
             {+ RESETD +}
             De := false;
             Dg := false;
             Ddef := false;
             Denb := 0;
             Depnb := 0;
             if (Du) then Dbcr := false;
             peerCntrl := Dsh;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*** DISCARDED Indications during rollback *******************************}

 trans
   from STATE_23_3, STATE_23_5
     when af.AF_U_ERROR_ind
       begin
       end;

     when af.AF_U_ERROR_cnf
       begin
       end;

     when af.AF_GRANT_CONTROL_ind
       begin
       end;


     when af.AF_REQUEST_CONTROL_ind
       begin
       end;

     when af.AF_HANDSHAKE_ind
       begin
       end;

     when af.AF_HANDSHAKE_GC_ind
       begin
       end;

  from STATE_23_3, STATE_23_5
     when af.AF_DEFER_ind
       begin
       end;

     when af.AF_PREPARE_ind
       begin
       end;

     when af.U_ASE_ind
       begin
       end;

{*************************************************************************}
{                                                                         }
{  Recovery                                                               }
{                                                                         }
{  Note: cases of double channels and unexpected CAF_GIVE-ind are handled }
{  in MACF module.                                                        }
{                                                                         }
{*************************************************************************}

trans
  from STATE_99
    provided retryRecovery
      delay( recoverDelay )
        var cplReqPars: cafPleaseType;
        begin
          output cpm.CAF_PLEASE_req(cplReqPars)
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans  {* recovery in READY state - initiated by superior TPPM *}
  when cpm.C_RECOVER_ind
    provided (param.recoveryState=commitState)
      from STATE_20_3, STATE_99
        to STATE_21_5
          var tpAbortPars: tpPAbortIndType;
              afAbortPars: afAbortReqType;
              recPars: cRecoverRspType;
              cDet: cafDetachType;
          begin
            {+ TPABiPF if not Danyb +}
            if (not Danyb) then
              begin
                tpAbortPars.diagnostic := ab_permanentFailure;
                tpAbortPars.rollback := false;
                output tpsu.TP_P_ABORT_ind( tpAbortPars);
              end;
            {+ AABrqPa if not Db +}
            if (not Db) then
              begin
                afAbortPars.aType := providerAbort;
                afAbortPars.mapping := abortRI;
                output af.AF_ABORT_req( afAbortPars);
              end;
            {+ ABDET +}
            Db := true;
            Danyb := true;
            Dtb := false;
            Dabp := false;
            {+ RECCOM +}
            DcmtIndRcvd := true;
            {+ CRErsRT +}  { duplicate channels are handled in macf }
            {+ CDETrq +}
            if (wantToWaitForOutcome) then
              begin
                recPars.recoveryState := retryLater;
                recPars.atomicActionId := Nzt;
                recPars.branchId := Dzb;
                output cpm.C_RECOVER_rsp( recPars);
                cdet.status := free_channel;
                output cpm.CAF_DETACH_req(cdet);
                Dchat := false;
              end;
          end;

  when cpm.C_RECOVER_cnf
    provided (param.recoveryState=unknownState)
      from STATE_20_3
        to STATE_23_8
          var cdet: cafDetachType;
          begin
            {+ TRBi +}
            output tpsu.TP_ROLLBACK_ind( noPars);
            {+ LOGDAMRB - see macf +}
            {+ CDETrq +}
            Dchat := false;
            cdet.status := free_channel;
            output cpm.CAF_DETACH_req( cdet);
            {+ INITRB +}
            Drolledback := true;
            DrollbackCnf := false;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {* abort on a channel *}
  when cpm.AF_ABORT_ind
    from STATE_20_3
      provided (param.mapping=abortRI)
        to STATE_99
           var cplPars: cafPleaseType;
               cdet: cafDetachType;
           begin
             Dchat := false;
             cdet.status := free_channel;
             output cpm.CAF_DETACH_req(cdet);
             output cpm.CAF_PLEASE_req(cplPars);
           end;

trans  {* recovery in DECIDED COMMIT state *}
 when cpm.C_RECOVER_ind
   provided (param.recoveryState=commitState)
     from STATE_21_5, STATE_21_6
       to STATE_21_5
         var tpAbortPars: tpPAbortIndType;
             afAbortPars: afAbortReqType;
             recPars: cRecoverRspType;
             cDet: cafDetachType;
         begin
           {+ TPABiPF if not Danyb +}
           if (not Danyb) then
             begin
               tpAbortPars.diagnostic := ab_permanentFailure;
               tpAbortPars.rollback := false;
               output tpsu.TP_P_ABORT_ind( tpAbortPars);
             end;
           {+ AABrqPa if not Db +}
           if (not Db) then
             begin
               afAbortPars.aType := providerAbort;
               afAbortPars.mapping := abortRI;
               output af.AF_ABORT_req( afAbortPars);
             end;
           {+ ABDET +}
           Db := true;
           Danyb := true;
           Dtb := false;
           Dabp := false;
           {+ RBNEXT if Dbr +}
           if (Dbr) then DrollbackNext := true;
           {+ if LDretry - CRErsRT and CDETrq +}
           if (wantToWaitForOutcome) then
             begin
               recPars.recoveryState := retryLater;
               recPars.atomicActionId := Nzt;
               recPars.branchId := Dzb;
               output cpm.C_RECOVER_rsp( recPars);
               cDet.status := free_channel;
               output cpm.CAF_DETACH_req(cDet);
               Dchat := false;
             end;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
   when cpm.C_RECOVER_cnf
     provided (param.recoveryState=retryLater)
       from STATE_20_3, STATE_21_5
         to STATE_99
           var cdet: cafDetachType;
           begin
             retryRecovery := true;
             {+ CDETrq +}
             Dchat := false;
             cdet.status := free_channel;
             output cpm.CAF_DETACH_req( cdet)
           end;

trans  {* completion of channel to superior TPPM *}
  from STATE_99
    when cpm.CAF_GIVE_ind
      to STATE_20_3
        var afRecPars: afRecoverReqType;
            cRecPars: cRecoverReqType;
            tokPars: afTokenGiveReqType;
        begin
          Dchat := true;
          {+ RECVRRDY +}
          if (Dhb) then
            begin
              afRecPars.mapping := recoverRdyRI;
              afRecPars.recoveryContextHandle := Dzhb;
              afRecPars.branchId := Dzb;
              afRecPars.atomicActionId := Nzt;
              output cpm.AF_RECOVER_req( afRecPars);
            end
          else
            begin
              cRecPars.recoveryState := readyState;
              cRecPars.branchId := Dzb;
              cRecPars.atomicActionId := Nzt;
              output cpm.C_RECOVER_req( cRecPars);
            end;
          output cpm.AF_TOKEN_GIVE_req(tokPars);
        end;

#include 'dialcom.e'

end; { body for superior dialogues }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{***  dialcom.e: behavior common to superior and subordinate dialogues **}
{***  this file is included by dial_sup_b.e and dial_sub_b.e *************}

{*****************************************************************************}
{                                                                             }
{ TP Data services                                                            }
{                                                                             }
{*****************************************************************************}

 trans
   when tpsu.TP_DATA_req
     from STATE_2, STATE_4, STATE_7
       provided ((Dsh or not peerCntrl) and (not Ds or not Dcr))
         begin
#include 'deldlg.e'
           {+ UASErq +}
           output af.U_ASE_req( param);
         end;

  when af.U_ASE_ind
    from STATE_2, STATE_3
      provided ((Dsh or peerCntrl) and (Ds or not Dcr))
        begin
          {+ TDTi +}
          output tpsu.TP_DATA_ind( param)
        end;

    from STATE_4
      begin
        {discard};
      end;

    from STATE_6, STATE_11
      provided (Dsh)
        begin
          if (Denb=0) then
             output tpsu.TP_DATA_ind( param)
          else
             {discard};
        end;

{*****************************************************************************}
{                                                                             }
{ Control transfer                                                            }
{                                                                             }
{*****************************************************************************}

 trans
   when tpsu.TP_GRANT_CONTROL_req
     from STATE_2, STATE_5
       provided (not Dsh and (not Ds or not Dcr))
         to STATE_3
           begin
#include 'deldlg.e'
             {+ AGCrq +}
             output af.AF_GRANT_CONTROL_req( param);
             peerCntrl := true
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when af.AF_GRANT_CONTROL_ind
    provided (not Dsh and (Ds or not Dcr))
      from STATE_3, STATE_4
        to STATE_2
           begin
             {+ TCGi +}
             output tpsu.TP_GRANT_CONTROL_ind( param);
             peerCntrl := false
           end;

 trans
   when tpsu.TP_REQUEST_CONTROL_req
     from STATE_3, STATE_7
       provided (not Ds or not Dcr)
         begin
#include 'deldlg.e'
           {+ ARCrq +}
           output af.AF_REQUEST_CONTROL_req( param);
         end;

  when af.AF_REQUEST_CONTROL_ind
    from STATE_2, STATE_6
      provided (not Dsh and (not Ds or not Dcr))
        begin
          {+ TRCi +}
          output tpsu.TP_REQUEST_CONTROL_ind( noPars);
        end;

    from STATE_3, STATE_4, STATE_11, STATE_15, STATE_20_1
      provided not Dsh
        begin
          {no control here}
        end;

{*************************************************************************}
{                                                                         }
{ User Error handling                                                     }
{                                                                         }
{*************************************************************************}

 trans
   when tpsu.TP_U_ERROR_req
     from STATE_2, STATE_14
       provided (not Dsh and (not Ds or not Dcr))
         to STATE_2
           begin
#include 'deldlg.e'
             peerCntrl := Dsh;
             {+ AUErq +}
             output af.AF_U_ERROR_req( errPars);
           end;

     from STATE_2, STATE_4
       provided (Dsh  and (not Ds or not Dcr))
         to STATE_4
           begin
#include 'deldlg.e'
             {+ AUErq +}
             output af.AF_U_ERROR_req( errPars);
             {+ VDenbINC +}
             Denb := Denb+1;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      from STATE_3
        provided (not Dcr or not Ds)
          to STATE_4
            begin
#include 'deldlg.e'
                {+ AUErq +}
                output af.AF_U_ERROR_req( errPars);
              end;

      from STATE_6
        provided (Dsh and (not Ds or not Dcr))
          to STATE_6
            begin
#include 'deldlg.e'
              {+ VDenbINC / AUErq +}
              Denb := Denb+1;
              output af.AF_U_ERROR_req( errPars);
            end;

      from STATE_7, STATE_12
        provided (not Ds or not Dcr)
          to STATE_2
            begin
#include 'deldlg.e'
              peerCntrl := Dsh;
              {+ AUErq +}
              output af.AF_U_ERROR_req( errPars);
            end;

      from STATE_8, STATE_9
        to STATE_6
          begin  {+ AUErq +}
            output af.AF_U_ERROR_req( errPars);
          end;

      from STATE_10
        to STATE_11
          begin {+ AUErq +}
            output af.AF_U_ERROR_req( errPars);
          end;

  when af.AF_U_ERROR_ind
    from STATE_2
      provided (not Dsh and (Ds or not Dcr))
        to STATE_5
          begin {+ TUEi +}
            output tpsu.TP_U_ERROR_ind( param);
          end;

      provided (Dsh and not Ds and not Dcr)
        begin {+ TUEi +}
          output tpsu.TP_U_ERROR_ind( param);
        end;

      provided (Dsh and Ds)
        begin {+ TUEi +}
          output tpsu.TP_U_ERROR_ind( param);
          if (Da) then {+ AUErs +}
            output af.AF_U_ERROR_rsp( param)
          else {+ VDepnbINC +}
            Depnb := Depnb+1;
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_3
  provided (Ds or not Dcr)
    to STATE_3
      begin
        {+ TUEi +}
        output tpsu.TP_U_ERROR_ind( param);
      end;

from STATE_4
  provided (Dsh and (Ds or not Dcr))
    to STATE_4
      begin
        {+ TUEi +}
        output tpsu.TP_U_ERROR_ind( param);
        {+ AUErs +}
        output af.AF_U_ERROR_rsp( param)
      end;

  provided (not Dsh)
    to STATE_4
      begin
      end;

from STATE_11, STATE_6, STATE_13
  provided (not Dsh and (not Ds or not Dcr))
    to STATE_3
      begin
        {+ TUEi +}
        output tpsu.TP_U_ERROR_ind( param);
      end;

  provided (Dsh and (Denb=0))
    to STATE_2
      begin
        {+ TUEi +}
        output tpsu.TP_U_ERROR_ind( param);
      end;

  provided (Dsh and (Denb>0))
    to STATE_4
      begin
        {+ TUEi +}
        output tpsu.TP_U_ERROR_ind( param);
      end;

when af.AF_U_ERROR_ind
 provided (Dsh)
  from STATE_7
    begin {+ TUEi +}
      output tpsu.TP_U_ERROR_ind( param);
      if (Da) then {+ AUErs +}
         output af.AF_U_ERROR_rsp( param)
      else {+ VDepnbINC +}
         Depnb := Depnb+1;
    end;

  from STATE_8, STATE_10
    to STATE_7
      begin
        {+ TUEi +}
        output tpsu.TP_U_ERROR_ind( param);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      from STATE_9
        to STATE_12
          begin
            {+ TUEi +}
            output tpsu.TP_U_ERROR_ind( param);
          end;

  trans
    when af.AF_U_ERROR_cnf
      from STATE_4
        provided (Dsh and (Denb>1))
          to SAME
            begin
              {+ DECDENB +}
              Denb := Denb-1;
              if (Denbb>0) then Denbb := Denbb-1;
            end;

        provided (Dsh and (Denb=1))
          to STATE_2
            begin
              {+ DECDENB +}
              Denb := Denb-1;
              if (Denbb>0) then Denbb := Denbb-1;
              peerCntrl := Dsh;
            end;

      from STATE_6, STATE_9, STATE_11, STATE_15, STATE_20_1
        provided (Dsh and (Denb>0))
          to SAME
            begin
              {+ DECDENB +}
              Denb := Denb-1;
              if (Denbb>0) then Denbb := Denbb-1;
            end;


{*****************************************************************************}
{                                                                             }
{ Handshake - Handshake and control                                           }
{                                                                             }
{*****************************************************************************}

 trans
   when tpsu.TP_HANDSHAKE_req
     from STATE_2, STATE_4
       provided ((Dsh or not peerCntrl) and Dh and (not Ds or not Dcr))
         to STATE_6
           begin
#include 'deldlg.e'
             {+ AHSrq +}
             output af.AF_HANDSHAKE_req( param);
           end;

     from STATE_7
       provided (Dsh and (not Ds or not Dcr))
         to STATE_8
           begin
#include 'deldlg.e'
             {+ AHSrq +}
             output af.AF_HANDSHAKE_req( param);
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when af.AF_HANDSHAKE_ind
  from STATE_2, STATE_3
    provided ((Dsh or peerCntrl) and Dh and (Ds or not Dcr))
      to STATE_7
        begin {+ THSi +}
          output tpsu.TP_HANDSHAKE_ind( param)
        end;

  from STATE_4
    provided (Dh and Dsh and (Ds or not Dcr) and (Denb>1))
      to STATE_4
        begin
          {+ DECDENB +}
          Denb := Denb-1;
          if (Denbb>0) then Denbb := Denbb-1;
        end;

    provided (Dh and Dsh and (Ds or not Dcr) and (Denb=1))
      to STATE_2
        begin
          {+ DECDENB +}
          Denb := Denb-1;
          if (Denbb>0) then Denbb := Denbb-1;
        end;

    provided (Dh and not Dsh and (Ds or not Dcr))
      to STATE_2
        begin
          peerCntrl := Dsh;
          {+ THSi +}
          output tpsu.TP_HANDSHAKE_ind( param)
        end;

 when af.AF_HANDSHAKE_ind
  from STATE_11
    provided (Dh and Dsh and (Denb>0))
      to STATE_11
        begin
          {+ DECDENB +}
          Denb := Denb-1;
          if (Denbb>0) then Denbb := Denbb-1;
        end;

    provided (Dh and not Dsh and (Denb=0))
      to STATE_10
        begin {+ THSi +}
          output tpsu.TP_HANDSHAKE_ind( param)
        end;

  from STATE_6
    provided (Dh and Dsh and (Denb>0))
      to STATE_6
        begin {+ DECDENB +}
          Denb := Denb-1;
          if (Denbb>0) then Denbb := Denbb-1;
        end;

    provided (Dh and not Dsh and (Denb=0))
      to STATE_8
        begin  {+ THSi +}
          output tpsu.TP_HANDSHAKE_ind( param)
        end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
   when tpsu.TP_HANDSHAKE_rsp
     from STATE_7
       provided (not Dsh and (not Ds or not Dcr))
         to STATE_3
            begin
#include 'deldlg.e'
              {+ AHSrs +}
              output af.AF_HANDSHAKE_rsp( param);
              peerCntrl := true;
            end;

       provided (Dsh and (not Ds or not Dcr))
         to STATE_2
            begin
#include 'deldlg.e'
              peerCntrl := true;
              {+ AHSrs +}
              output af.AF_HANDSHAKE_rsp( param);
            end;

     from STATE_8
       to STATE_6
          begin
#include 'deldlg.e'
            peerCntrl := true;
            {+ AHSrs +}
            output af.AF_HANDSHAKE_rsp( param);
          end;

     from STATE_10
       to STATE_11
          begin
#include 'deldlg.e'
             output af.AF_HANDSHAKE_rsp( param);
          end;

   when af.AF_HANDSHAKE_cnf
     from STATE_6
       provided (Dsh and (Denb>0))
         to STATE_4
           begin
             peerCntrl := true;
             {+ THSc +}
             output tpsu.TP_HANDSHAKE_cnf( param)
           end;

       provided (not Dsh or (Denb=0))
         to STATE_2
           begin
             peerCntrl := Dsh;
             {+ THSc +}
             output tpsu.TP_HANDSHAKE_cnf( param)
           end;

     from STATE_8
       to STATE_7
         begin
           peerCntrl := true;
           {+ THSc +}
           output tpsu.TP_HANDSHAKE_cnf( param)
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      from STATE_9
        to STATE_12
          begin
           {+ THSc +}
            output tpsu.TP_HANDSHAKE_cnf( param)
          end;

 trans
   when tpsu.TP_HANDSHAKE_GC_req
     from STATE_2
       provided (not Dsh and Dh and (not Ds or not Dcr))
         to STATE_13
           begin
#include 'deldlg.e'
             {+ AHSGCrq +}
             output af.AF_HANDSHAKE_GC_req( param);
           end;

   when af.AF_HANDSHAKE_GC_ind
     provided (not Dsh and Dh and (Ds or not Dcr))
       from STATE_3
         to STATE_14
           begin
             {+ THSGCi +}
             output tpsu.TP_HANDSHAKE_GC_ind( param);
             peerCntrl := false
           end;

       from STATE_4
         to STATE_2
           begin
             {+ THSGCi +}
             output tpsu.TP_HANDSHAKE_GC_ind( param);
             peerCntrl := false
           end;


 trans
   when tpsu.TP_HANDSHAKE_GC_rsp
     from STATE_14
       to STATE_2
         provided (not Ds or not Dcr)
           begin
#include 'deldlg.e'
             {+ AHSGCrs +}
             output af.AF_HANDSHAKE_GC_rsp( param);
             peerCntrl := false;
           end;

   when af.AF_HANDSHAKE_GC_cnf
     from STATE_13
       to STATE_3
         begin
           {+ THSGCc +}
           peerCntrl := true;
           output tpsu.TP_HANDSHAKE_GC_cnf( param)
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{ Dialogue termination                                                    }
{                                                                         }
{*************************************************************************}

 trans
   when tpsu.TP_END_DIALOGUE_req
     from STATE_2, STATE_4
       provided (not param.confirmation and not Dl and not Dcr and
                 (Dsh or not peerCntrl))
         to STATE_1
            var endDialPars: afEndDialReqType;
            begin
#include 'deldlg.e'
              {+ AEDrq +}
              endDialPars.confirmation := false;
              output af.AF_END_DIALOGUE_req( endDialPars);
              {+ SETFEAR +}
              sao.Abf := (Du and Ds);
              {+ SDETrqF +}
              if (Du and Ds) then
                detPars.status := begin_fear
              else
                detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
            end;

   when tpsu.TP_END_DIALOGUE_req
     from STATE_2, STATE_4
       provided (param.confirmation and not Dcr and not Dl and
                 (Dsh or not peerCntrl))
         to STATE_11
            var endDialPars: afEndDialReqType;
            begin
#include 'deldlg.e'
              {+ AEDrq +}
              peerCntrl := Dsh;
              endDialPars.confirmation := true;
              output af.AF_END_DIALOGUE_req( endDialPars);
            end;

   when af.AF_END_DIALOGUE_ind
     from STATE_2, STATE_3, STATE_4, STATE_6, STATE_11
       to STATE_1
         provided (not param.confirmation and (Ds or not Dcr) and not Dl and
                   (Dsh or peerCntrl))
           begin
             {+ TEDi +}
             output tpsu.TP_END_DIALOGUE_ind( param);
             detPars.status := free_assoc;
             {+ SDETrqF +}
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
from STATE_2, STATE_3
  to STATE_12
    provided (param.confirmation and not Dcr and not Dl and
         (Dsh or peerCntrl))
      begin
        peerCntrl := true;
        {+ SDETrqF +}
        output tpsu.TP_END_DIALOGUE_ind( param);
      end;

from STATE_4
  provided (param.confirmation and not Dl and not Dsh)
    to STATE_2
      begin
        {+ TEDi +}
        output tpsu.TP_END_DIALOGUE_ind( param)
      end;

  provided (param.confirmation and not Dl and Dsh and (Denb=1))
    to STATE_2
      begin
        {+ DECDENB +}
        Denb := Denb-1;
        if (Denbb>0) then Denbb := Denbb-1;
      end;

  provided (param.confirmation and not Dl and Dsh and (Denb>1))
    to STATE_4
      begin
        {+ DECDENB +}
        Denb := Denb-1;
        if (Denbb>0) then Denbb := Denbb-1;
      end;

from STATE_6
  provided (param.confirmation and Dsh and not Dl and (Ds or not Dcr) and
            (Denb=0))
    to STATE_9
      begin
        {+ TEDi +}
        output tpsu.TP_END_DIALOGUE_ind( param)
      end;

  provided (param.confirmation and Dsh and not Dl and (Ds or not Dcr) and
            (Denb>0))
    to STATE_6
      begin
        {+ DECDENB +}
        Denb := Denb-1;
        if (Denbb>0) then Denbb := Denbb-1;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
   when af.AF_END_DIALOGUE_ind
      from STATE_11
       provided (param.confirmation and not Dl and Dsh and (Denb=0))
         to STATE_1
           var abortPars: tpPAbortIndType;
           begin
             {+ TPABiED +}
             abortPars.diagnostic := ab_endDialogueCollision;
             abortPars.rollback := false;
             output tpsu.TP_P_ABORT_ind( abortPars);
             {+ SDETrqF +}
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
           end;

       provided (param.confirmation and not Dl and Dsh and (Denb>0))
         to STATE_11
           begin
             {+ DECDENB +}
             Denb := Denb-1;
             if (Denbb>0) then Denbb := Denbb-1;
           end;

   when tpsu.TP_END_DIALOGUE_rsp
     from STATE_9
       to STATE_1
         begin
#include 'deldlg.e'
           {+ AEDrs +}
           output af.AF_END_DIALOGUE_rsp( param);
           {+ SDETrqF +}
           detPars.status := free_assoc;
           output saf.SAF_DETACH_ASSOCIATION_req( detPars);
           disconnect sao;
         end;

   when af.AF_END_DIALOGUE_cnf
     from STATE_11
       to STATE_1
          var endDialPars: tpEndDialogueCnfType;
          begin
            {+ TEDc +}
            output tpsu.TP_END_DIALOGUE_cnf( endDialPars);
            {+ SDETrqF +}
            detPars.status := free_assoc;
            output saf.SAF_DETACH_ASSOCIATION_req( detPars);
            disconnect sao;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{************************************************************************}
{                                                                        }
{ Aborts on dialogues with commitment level none                         }
{                                                                        }
{************************************************************************}
 trans
   when tpsu.TP_U_ABORT_req
     from STATE_2, STATE_3, STATE_4, STATE_5, STATE_11, STATE_12,
          STATE_6, STATE_7, STATE_8, STATE_9, STATE_10, STATE_13, STATE_14
       provided (not Dl and (not Ds or not Dcr))
         to STATE_1
            var abortPars: afAbortReqType;
            begin
#include 'deldlg.e'
              {+ AABrqUd +}
              abortPars.aType := userAbort;
              abortPars.mapping := dataRI;
              output af.AF_ABORT_req( abortPars);
              {+ SDETrqF +}
              if (Du and Ds) then
                detPars.status := begin_fear
              else
                detPars.status := free_assoc;
              output saf.SAF_DETACH_ASSOCIATION_req( detPars);
              disconnect sao;
            end;

    when af.AF_ABORT_ind
      from STATE_2, STATE_3, STATE_4, STATE_5, STATE_11, STATE_12,
           STATE_6, STATE_7, STATE_8, STATE_9, STATE_10, STATE_13, STATE_14
        provided (not Dl and ((param.aType=providerAbort) or Ds or not Dcr))
          to STATE_1
            var uAbortPars: tpUAbortIndType;
                pAbortPars: tpPAbortIndType;
            begin
              if (param.aType=userAbort) then
                begin
                  {+ TUABi +}
                  uAbortPars.rollback := false;
                  output tpsu.TP_U_ABORT_ind( uAbortPars)
                end
              else
                begin
                  {+ TPABiPF +}
                  pAbortPars.diagnostic := param.diagnostic;
                  pAbortPars.rollback := false;
                  output tpsu.TP_P_ABORT_ind( pAbortPars)
                end;
              if (param.mapping<>abortRI) then
                begin {+ SDETrqF +}
                  detPars.status := free_assoc;
                  output saf.SAF_DETACH_ASSOCIATION_req( detPars);
                  disconnect sao;
                end
          end;

 trans  {releasing dialogue module}
  from STATE_1
    provided sao.dead
      begin
        dead := true;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ *** deldlg.e : action to delimit a dialogue with superior ***************}
{ ****************************************************************************}

      {+ DELIMIT +}  { Nrn and Ncr are set by macf }
      begin
        if (Ds and not Da) then
           begin
             beginDialPars.mapping := dataRI;
             beginDialPars.result := accepted;
             beginDialPars.dialChanCorrel := savedBDInd.dialChanCorrel;
             beginDialPars.functionalUnits := savedBDInd.functionalUnits;
             output af.AF_BEGIN_DIALOGUE_rsp( beginDialPars);
             if (Dl) then
                 output af.C_BEGIN_rsp( beginPars);
             while (Depnb>0) do
               begin
                 Denb := Denb-1;
                 output af.AF_U_ERROR_rsp( errPars);
               end;
             Dcr := false;
             Da := true;
           end;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{** cpm.e: channel manager module ***************************************}

{************************************************************************}
{*                                                                      *}
{*  A single CPM module exists in any give site, at any given time.  The *}
{*  module manages a collection of channels established in support of   *}
{*  transaction branch recovery.                                        *}
{*                                                                      *}
{*  Channels are created following a request from a local TPPM or an    *}
{*  indication of channel establishment initiated by a distant CPM.  When *}
{*  a channel is established, and after the corresponding TPPM is located, *}
{*  the channel is transferred under the TPPM's control (see channel    *}
{*  module below).  After recovery is complete, the channel is put back *}
{*  under CPM control.  The CPM may choose to release an unused channel *}
{*  anytime, but the procedures involved are outside the scope of this  *}
{*  standard.                                                           *}
{*                                                                      *}
{*  The CPM is also responsible for re-attempting recovery on channels  *}
{*  that failed while under CPM control and while the TPPM is waiting   *}
{*  for a new channel.                                                  *}
{*                                                                      *}
{************************************************************************}

 module cpm_m activity;

  ip
     tppm: array[1..MAX_TPPMS] of CPM_TPPM_service( cpm); {tppm ports}

  end; {module interface definition - cpm_m}

 body cpm_b for cpm_m;

#include 'chan_m.e'

 function acceptableChannel(cplPars,attrs:cafPleaseType):boolean; primitive;
   {do the attributes of an existing channel match those of a caf_please}
   {request ?                                                          }

 function allocateNewChanIdx: integer; primitive;
   {creates a new local identifier for a channel}

 procedure deallocateChanIdx( id: integer); primitive;
   {releases local identifier for a channel}

 ip
   chanAccess: array[1..MAX_CHANS] of CPM_service( user);

 modvar
   chan: array[1..MAX_CHANS] of cpm_channel_m;
   retryChan: cpm_channel_m;

 initialize
   begin
   end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {channel requested by local TPPM}
  any idx: 1..MAX_TPPMS do
    when tppm[idx].TPPM_TO_CPM_req
      provided (param.order=cafPleaseReq)
        var cIdx: integer;
        begin
           forone c: cpm_channel_m suchthat c.free
              and acceptableChannel(param.cafPleasePars, c.attributes) do
                 begin
                   disconnect c;
                   cIdx := c.tppmIdx;
                 end
             otherwise
                 begin
                   cIdx := allocateNewChanIdx;
                   init chan[cidx] with cpm_channel_b( nullAssocId, true);
                   chan[cidx].idx := cIdx;
                 end;
          chan[cidx].tppmIdx := idx;
          chan[cidx].branchId := param.branchId;
          chan[cidx].atomicActionId := param.atomicActionId;
          connect chanAccess[cidx] to chan[cIdx].tppm;
          output chanAccess[cidx].CAF_PLEASE_req( param.cafPleasePars)
        end;

trans {retry channel after channel failure undetected by TPPM}
  provided exist c: cpm_channel_m suchthat c.retryChannel
    delay ( recoverDelay )
      var cafPlPars: cafPleaseType;
          cidx: integer;
      begin
        forone c: cpm_channel_m suchthat c.retryChannel do
            begin
              retryChan := c;
              cafPlPars := c.attributes;
            end;
        forone c: cpm_channel_m
          suchthat (c.free and acceptableChannel(cafPlPars,c.attributes))
             do begin
                  disconnect c;
                  cidx := c.tppmidx;
                end
             otherwise
                begin
                  cidx := allocateNewChanIdx;
                  init chan[cidx] with cpm_channel_b( nullAssocId, true);
                  chan[cidx].idx := cidx;
                end;
        chan[cidx].tppmIdx := retryChan.tppmIdx;
        chan[cidx].branchId := retryChan.branchId;
        chan[cidx].atomicActionId := retryChan.atomicActionId;
        connect chanAccess[cidx] to chan[cidx].tppm;
        output chanAccess[cidx].CAF_PLEASE_req( cafPlPars);
        retryChan.retryChannel := false
      end;
```

IECNORM.COM: Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans {channel creation initiated by distant CPM}
  provided pendingTPBeginChannelInd
    var assocId: assocIdType;
        cidx: integer;
    begin
      getAssocIdForPendingTPChanInd( assocId);
      cidx := allocateNewChanIdx;
      init chan[cidx] with cpm_channel_b( assocId, false);
      connect chanAccess[cidx] to chan[cidx].tppm;
    end;


trans {channel release}
  provided exist c: cpm_channel_m suchthat c.terminated
    begin
      all c: cpm_channel_m do
        if c.terminated then
          begin
            deallocateChanIdx(c.idx);
            terminate c;
          end;
    end;

{*************************************************************************}
{* Relay between channel modules and TPPMs (and vice versa)              *}
{*************************************************************************}

{* channel -> TPPM ******************************************************}
{* if channel is not already attached to a TPPM (ie, if the indication  }
{* is a CAF-GIVE or a CAF-RECOVER-ind, then the cpm checks the logs to  }
{* verify that the TPPM still exists - if it does, the CPM forwards the }
{* primitive  to the TPPM, otherwise, the CPM replies to it.            }
{* Other primitives are systematically forwarded to the TPPM.           }

trans
  any idx: 1..MAX_CHANS do
    when chanAccess[idx].CAF_GIVE_ind
      var cafDetPars: cafDetachType;
          cgInd: recoveryIndType;
      begin
        if (existNodeRecord(chan[idx].atomicActionId)) then
          begin
            cgInd.order := cafGiveInd;
            cgInd.chanIdx := idx;
            chan[idx].tppmIdx :=
                          getTPPMIdx(param.atomicActionId,param.branchId);
            output tppm[chan[idx].tppmIdx].CPM_TO_TPPM_ind(cgInd);
          end
        else
          begin
            cafDetPars.status:= not_used;
            output chanAccess[idx].CAF_DETACH_req(cafDetPars)
          end
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  any idx: 1..MAX_CHANS do
    when chanAccess[idx].CAF_RECOVER_ind(param)
      var heurPars: afHeuristicReqType;
          recRspPars: cRecoverRspType;
          recovOrderPars: recoveryIndType;
          aid : atomicActionIdType;
          bid : branchIdType;
          cafDetPars: cafDetachType;
      begin
        if (existNodeRecord(chan[idx].atomicActionId)) then
          begin
            recovOrderPars.order := cafRecoverInd;
            recovOrderPars.chanIdx := idx;
            recovOrderPars.cafRecoverIndPars := param;
            output tppm[chan[idx].tppmIdx].CPM_TO_TPPM_ind(recovOrderPars);
            chan[idx].tppmIdx :=
                              getTPPMIdx(param.atomicActionId,param.branchId);
          end
        else {* transaction completed in this node *}
          begin
            aid := chan[idx].atomicActionId;
            bid := chan[idx].branchId;
            if (param.recoveryState=commitState) then
              begin
                if (existLogDamage(aid)) then
                  begin
                    heurPars.mapping := recoverDoneRC;
                    heurPars.heuristicReport := recordedDamage(aid);
                    heurPars.atomicActionId := aid;
                    heurPars.branchId := bid;
                    output chanAccess[idx].AF_HEURISTIC_req( heurPars);
                  end
                else
                  begin
                    recRspPars.recoveryState := doneState;
                    recRspPars.atomicActionId := aid;
                    recRspPars.branchId := bid;
                    output chanAccess[idx].C_RECOVER_rsp( recRspPars);
                  end;
              end
            else if (param.recoveryState=readyState) then
              begin
                recRspPars.recoveryState := unknownState;
                recRspPars.atomicActionId := aid;
                recRspPars.branchId := bid;
                output chanAccess[idx].C_RECOVER_rsp( recRspPars);
              end;
            cafDetPars.status := not_used;
            output chanAccess[idx].CAF_DETACH_req(cafDetPars);
          end;
      end;

  any idx: 1..MAX_CHANS do
    when chanAccess[idx].C_RECOVER_ind
      var recInd: recoveryIndType;
      begin
        recInd.order := cRecoverInd;
        recInd.chanIdx := idx;
        recInd.cRecoverIndPars := param;
        output tppm[chan[idx].tppmIdx].CPM_TO_TPPM_ind( recInd)
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    when chanAccess[idx].C_RECOVER_cnf
      var recInd: recoveryIndType;
      begin
        recInd.order := cRecoverCnf;
        recInd.chanIdx := idx;
        recInd.cRecoverCnfPars := param;
        output tppm[chan[idx].tppmIdx].CPM_TO_TPPM_ind( recInd)
      end;

    when chanAccess[idx].AF_HEURISTIC_ind
      var recInd: recoveryIndType;
      begin
        recInd.order := afHeurRepInd;
        recInd.chanIdx := idx;
        recInd.afHeurRepIndPars := param;
        output tppm[chan[idx].tppmIdx].CPM_TO_TPPM_ind( recInd)
      end;

    when chanAccess[idx].AF_ABORT_ind
      var recInd: recoveryIndType;
      begin
        recInd.order := afAbortInd;
        recInd.chanIdx := idx;
        recInd.afAbortIndPars := param;
        output tppm[chan[idx].tppmIdx].CPM_TO_TPPM_ind( recInd)
      end;

{* TPPM -> channel ********************************************************}
{* simply forwards recovery primitive to appropriate channel.             }

trans
  any idx: 1..MAX_TPPMS do
   when tppm[idx].TPPM_TO_CPM_req
     provided (param.order=cafDetachReq)
       begin
        output chanAccess[param.chanIdx].CAF_DETACH_req(param.cafDetachPars)
       end;

   when tppm[idx].TPPM_TO_CPM_req
     provided (param.order=cRecoverReq)
       begin
        output chanAccess[param.chanIdx].C_RECOVER_req(param.cRecoverReqPars)
       end;

   when tppm[idx].TPPM_TO_CPM_req
     provided (param.order=afRecoverReq)
       begin
        output chanAccess[param.chanIdx].
                                        AF_RECOVER_req(param.afRecoverReqPars)
       end;


   when tppm[idx].TPPM_TO_CPM_req
     provided (param.order=cRecoverRsp)
       begin
        output chanAccess[param.chanIdx].C_RECOVER_rsp(param.cRecoverRspPars)
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when tppm[idx].TPPM_TO_CPM_req
    provided (param.order=afHeurRepReq)
      begin
       output chanAccess[param.chanIdx].
                                    AF_HEURISTIC_req(param.afHeurRepReqPars)
      end;

  when tppm[idx].TPPM_TO_CPM_req
    provided (param.order=afTokenGiveReq)
      begin
       output chanAccess[param.chanIdx].
                                   AF_TOKEN_GIVE_req(param.afTokenGiveReqPars)
      end;

  when tppm[idx].TPPM_TO_CPM_req
    provided (param.order=afAbortReq)
      begin
       output chanAccess[param.chanIdx].AF_ABORT_req(param.afAbortReqPars)
      end;

end; {cpm_b module body}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{** chan_m.e: TP channel module ******************************************}

{************************************************************************}
{*                                                                      *}
{* Channel modules.                                                     *}
{*                                                                      *}
{* These modules are created by the site's CPM  after  a CAF_PLEASE_req *}
{* (initiator side) from a local TPPM transaction branch or after an    *}
{* af_begin_dialogue_ind from a distant site (responder side).          *}
{*                                                                      *}
{* The underlying channel may be under the control of either the CPM or *}
{* the TPPM.  In the latter case, it simply forwards primitives received *}
{* from the association to the TPPM, and vice versa.                    *}
{*                                                                      *}
{* A channel may be detached from a TPPM by invoking CAF_DETACH_req.    *}
{* It will then return under the control of the CPM and may be attributed *}
{* to a new transaction branch.                                         *}
{*                                                                      *}
{* In case of abnormal termination of a channel under the CPM's control *}
{* the module may set its "retryChannel" variable asking the CPM module to *}
{* to retry establishing a channel later on.                            *}
{*                                                                      *}
{************************************************************************}
 module cpm_channel_m activity(
   assocId: assocIdType; {passed to modules linked with loser side assocs}
   initiator: boolean
  );
 ip
   tppm: CPM_Service( provider) individual queue;

 export
  free: boolean; { can be used by local TPPMs to initiate recovery }
  terminated: boolean; { terminated - retry channel may be necessary }
  retryChannel: boolean; { failed and recovery should be retried }
  attributes: cafPleaseType; { attributes - used to dtermine if a channel }
                      { can be used to perform recovery of a given branch. }
  atomicActionId: atomicActionIdType; { of current transaction branch }
  branchId: branchIdType; { of current transaction branch }
  idx: integer; {channel mgmt variable; index of channel in cpm}
  tppmIdx: integer; {channel mgmt variable; index of attached tppm (if any)}
 end;

 body cpm_channel_b for cpm_channel_m;

#include 'sao.e'

 state
   STATE_1,  {* initial channel state *}
   STATE_2,  {* available for one or two way reovery - token owned *}
   STATE_3,  {* available for two way recovery - token not owned *}
   STATE_4,  {* under TPPM control- attached to a transaction branch*}
   STATE_5,  {* under CPM control - waiting for token *}
   STATE_6,  {* under CPM control - waiting for begin_dial_conf. *}
   STATE_7,  {* under CPM control - cleanup after detach. *}
   RELEASED; {* terminated, possibly leading to retrying establishment  *}

 modvar
   sao: sao_m;

 ip
   af: AF_Service( user) common queue;
   saf: SAF_Service( user) common queue;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
 var
   nullBeginDialPars: afBeginDialReqType;
   detPars: safDetachAssocReqType;
   ptkPars: afTokenGiveReqType;
   tkPlsPars: afTokenPleaseReqType;
   afBeginDialRspPars: afBeginDialRspType;
   afBeginDialReqPars: afBeginDialReqType;
   cRecRsp: cRecoverRspType;
   cafGPars: cafGiveType;
   cafRecPars: cafRecoverIndType;
   cRecPars: cRecoverIndType;
   endDialPars: afEndDialReqType;
   cinit: boolean;

 initialize
   to STATE_1
     begin
       cinit := false;
       free := false;
       if (not initiator) then
         begin
           init sao with sao_b( true, assocId, nullBeginDialPars, false);
           connect sao.af to af;
           connect sao.saf to saf;
         end;
       retryChannel := false;
     end;

{*************************************************************************}
{*                                                                       *}
{*  Channel Establishment.                                               *}
{*                                                                       *}
{*************************************************************************}

{*** RESPONDER SIDE *****************************************************}
 trans
   from STATE_1
     when af.AF_BEGIN_DIALOGUE_ind
       provided not rejectDialInd( param)
         to STATE_3
            begin
              {+ ABDrsAd +}
              afBeginDialRspPars.result := accepted;
              afBeginDialRspPars.mapping := dataRI;
              output af.AF_BEGIN_DIALOGUE_rsp( afBeginDialRspPars);
              {+ AtwrT if necessary +}
              sao.Atwr := twRecovery(param);
            end;

       provided rejectDialInd( param)
         to RELEASED
           begin
             {+ ABDrsRPd +}
             afBeginDialRspPars.result := rejected_p;
             afBeginDialRspPars.mapping := dataRI;
             afBeginDialRspPars.diagnostic := dr_noReason;
             output af.AF_BEGIN_DIALOGUE_rsp( afBeginDialRspPars);
             {+ SDETrqF +}
             detPars.status := free_assoc;
             output saf.SAF_DETACH_ASSOCIATION_req( detPars);
             disconnect sao;
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*** INITIATOR SIDE ** new channel ************************************}

  trans
    when tppm.CAF_PLEASE_req
       from STATE_1
         to STATE_6
           begin
             {+ SETID - see cpm +}
             {+ AtwrT if necessary +}
             mapCAFPleaseToAF( param, afBeginDialReqPars);
             sao.Atwr := twRecovery(afBeginDialReqPars);
             {+ ABDrqCO +}
             init sao with sao_b(true,nullAssocId,afBeginDialReqPars,false);
             connect sao.af to af;
             connect sao.saf to saf;
             attributes := param;
             cinit := true;
           end;

     when af.AF_BEGIN_DIALOGUE_cnf
       from STATE_6
         provided (param.result=accepted)
           to STATE_4
             begin
               {+ VAtppmT +}
               sao.Atppm := true;
               {+ CGIVi +}
               cafGPars.branchId := branchId;
               cafGPars.atomicActionId := atomicActionId;
               output tppm.CAF_GIVE_ind( cafGPars);
             end;

         provided (param.result<>accepted)
           to RELEASED
             begin
               retryChannel := true;
               {+ SDETrqF +}
               detPars.status := free_assoc;
               output saf.SAF_DETACH_ASSOCIATION_req( detPars);
               disconnect sao;
             end;

    when saf.SAF_ASSOCIATION_LOST_ind
      from STATE_6
        to RELEASED
          begin
            retryChannel := true
          end;

      from STATE_5, STATE_3
        to RELEASED
          begin
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*** INITIATOR SIDE ** existing channel ***********************************}

    when tppm.CAF_PLEASE_req
       from STATE_2
         to STATE_4
           begin
             free := false;
             {+ VAtppmT +}
             sao.Atppm := true;
             {+ CGIVi +}
             cafGPars.branchId := branchId;
             cafGPars.atomicActionId := atomicActionId;
             output tppm.CAF_GIVE_ind(cafGPars)
           end;

    when tppm.CAF_PLEASE_req
       from STATE_3
         to STATE_5
           begin
             free := false;
             {+ VAtppmT +}
             sao.Atppm := true;
             cafGPars.branchId := branchId;
             cafGPars.atomicActionId := atomicActionId;
             output af.AF_TOKEN_PLEASE_req(tkPlsPars);
           end;

    when af.AF_TOKEN_GIVE_ind
      from STATE_5
        to STATE_4
          begin
             {+ VAtppmT +}
             sao.Atppm := true;
             {+ CGIVi +}
             cafGPars.branchId := branchId;
             cafGPars.atomicActionId := atomicActionId;
             output tppm.CAF_GIVE_ind(cafGPars)
          end;

{*** TOKEN MGMT while under CPM control **********************************}

  trans
    when af.AF_TOKEN_GIVE_ind
      from STATE_3
        provided (sao.Atwr)
          to STATE_2
            begin
              {+ VAtokxF +}
              sao.Atokx := false;
            end;

    when af.AF_TOKEN_PLEASE_ind
      from STATE_2
        to STATE_3
          provided sao.Atwr and hasToken( sao.assocId)
            begin
              ptkPars.reason := two_way_recovery;
              output af.AF_TOKEN_GIVE_req(ptkPars)
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{*                                                                       *}
{*  Recovery primitives - UNDER CPM CONTROL                              *}
{*                                                                       *}
{*  Recovery indications received are transferred to CPM module which,   *}
{*  as in the case of CAF-GIVE-ind, will check that the TPPM still exists*}
{*  and either forward it or respond to it appropriately.                *}
{*                                                                       *}
{*************************************************************************}
  trans
    from STATE_3, STATE_5
      when af.C_RECOVER_ind
        provided (not sao.Atokx)
          to STATE_4
            begin
              {+ SETTOKX +}
              if (sao.Atwr) then sao.Atokx := true;
              {+ CREi +}
              branchId :=  param.branchId;
              atomicActionId :=  param.atomicActionId;
              output tppm.CAF_RECOVER_ind(param);
              {+ VAtppmT +}
              sao.Atppm := true;
            end;

      when af.AF_RECOVER_ind
        provided (not sao.Atokx)
          to STATE_4
            begin
              {+ SETTOKX +}
              if (sao.Atwr) then sao.Atokx := true;
              {+ CAFREi - will be rejected if node is gone - see cpm +}
              branchId :=  param.branchId;
              atomicActionId :=  param.atomicActionId;
              cafRecPars.branchId := branchId;
              cafRecPars.atomicActionId := atomicActionId;
              if (param.mapping=recoverRdyRI) then
                 cafRecPars.recoveryState := readyState
              else
                 cafRecPars.recoveryState := commitState;
              output tppm.CAF_RECOVER_ind( cafRecPars);
              {+ VAtppmT +}
              sao.Atppm := true;
            end;

{*************************************************************************}
{*                                                                       *}
{*  Channel Under TPPM control                                           *}
{*                                                                       *}
{*  Recovery primitives are simply forwarded from/to TPPM to/from sacf.  *}
{*  In cases of an abort, the channel module terminates.                 *}
{*  In cases of P_TOKEN_req from TPPM, the channel module forwards it if *}
{*  it performs two-way recovery.                                        *}
{*                                                                       *}
{*************************************************************************}

 trans {TPPM -> SACF}
   from STATE_4
     when tppm.C_RECOVER_req
       begin
         output af.C_RECOVER_req(param);
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    when tppm.AF_RECOVER_req
      begin
        output af.AF_RECOVER_req(param);
      end;

    when tppm.AF_HEURISTIC_req
      begin
        output af.AF_HEURISTIC_req(param)
      end;

    when tppm.C_RECOVER_rsp
      begin
        output af.C_RECOVER_rsp(param);
      end;

    when tppm.AF_TOKEN_GIVE_req
      begin
        param.reason := two_way_recovery;
        if (sao.Atwr and hasToken(sao.assocId)) then
          output af.AF_TOKEN_GIVE_req( param)
      end;

    when tppm.AF_ABORT_req
      to RELEASED
        begin
          output af.AF_ABORT_req( param);
          free := false;
        end;

trans {SACF -> TPPM}
  from STATE_4
    when af.AF_ABORT_ind
      to RELEASED
        begin
          free := false;
          output tppm.AF_ABORT_ind( param);
          {+ SDETrqF +}
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
          disconnect sao;
        end;

    when af.AF_RECOVER_ind
      begin
        if (sao.Atwr) then sao.Atokx := true;
        if (param.mapping = recoverRdyRI) then
          cRecPars.recoveryState := readyState
        else
          cRecPars.recoveryState := commitState;
        output tppm.C_RECOVER_ind( cRecPars)
      end;

    when af.C_RECOVER_ind
      begin
        if (sao.Atwr) then sao.Atokx := true;
        output tppm.C_RECOVER_ind( param)
      end;

    when af.AF_HEURISTIC_ind
      begin
        output tppm.AF_HEURISTIC_ind( param)
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      when af.C_RECOVER_cnf
        begin
          if (sao.Atwr) then sao.Atokx := true;
          output tppm.C_RECOVER_cnf( param)
        end;

      when af.AF_TOKEN_GIVE_ind
        provided (sao.Atwr and sao.Atokx)
          begin
            sao.Atokx := false;
          end;

      when af.AF_TOKEN_PLEASE_ind
        provided (sao.Atwr)
          begin
          end;

{*************************************************************************}
{*                                                                       *}
{*  Channel Detach and Aborts                                            *}
{*                                                                       *}
{*************************************************************************}
  trans {channel detach by tppm / cpm }
    when tppm.CAF_DETACH_req
      from STATE_4
        provided ((param.status=free_channel) and
                  ((sao.Atwr and hasToken(sao.assocId)) or
                   (not sao.Atwr and cinit)))
          to STATE_2
            begin
              free := true;
              sao.Atppm := false;
            end;

        provided ((param.status=free_channel) and
                  ((sao.Atwr and not hasToken(sao.assocId)) or
                   (not sao.Atwr and not cinit)))
          to STATE_3
            begin
              free := true;
              sao.Atppm := false;
            end;

        provided ((param.status=not_used) and (sao.Atwr or cinit))
          to STATE_2
            begin
              free := true;
              sao.Atppm := false;
            end;

        provided ((param.status=not_used) and not sao.Atwr and not cinit)
          to STATE_2
            begin
              free := true;
              sao.Atppm := false;
            end;

        provided (param.status = cleanup)
          to STATE_7
            begin
              sao.Atppm := false;
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  when tppm.AF_ABORT_req
    from STATE_4
      to RELEASED
        begin
          free := false
        end;

  when af.AF_ABORT_ind
    from STATE_2, STATE_3, STATE_7
      to RELEASED
          begin
            free := false;
          end;

    from STATE_5, STATE_6
      to RELEASED
        begin
          free := false;
          retryChannel := true;
        end;

trans {cleanup after detach}
  from STATE_7
    when af.C_RECOVER_cnf
      provided (sao.Atwr)
        to STATE_3
          begin
          end;

      provided (not sao.Atwr)
        to STATE_2
          begin
          end;

    when af.C_RECOVER_ind
      provided (sao.Atwr)
        to STATE_3
          begin
            cRecRsp.recoveryState := retryLater;
            cRecRsp.atomicActionId := param.atomicActionId;
            cRecRsp.branchId := param.branchId;
            output af.C_RECOVER_rsp(cRecRsp);
            sao.Atokx := true;
          end;

      provided (not sao.Atwr)
        to STATE_2
          begin
            cRecRsp.recoveryState := retryLater;
            cRecRsp.atomicActionId := param.atomicActionId;
            cRecRsp.branchId := param.branchId;
            output af.C_RECOVER_rsp(cRecRsp);
            sao.Atokx := true;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { ending free channel }
  from STATE_2
    provided terminateChannel and hasToken(sao.assocId)
      to RELEASED
        begin
          endDialPars.confirmation := false;
          output af.AF_END_DIALOGUE_req( endDialPars);
          detPars.status := free_assoc;
          output saf.SAF_DETACH_ASSOCIATION_req( detPars);
        end;

  from STATE_5, STATE_3
    when af.AF_END_DIALOGUE_ind
      to RELEASED
        begin
        end;

trans {clean up}
  from RELEASED
    provided sao.dead and not retryChannel
      begin
        terminated := true;
      end;

end; {of channel module }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ ** sao.e: SACF & TPASE module ********************************************* }

{ ************************************************************************** }
{                                                                            }
{ Module for Single Association Object.                                      }
{                                                                            }
{ The module has two interfaces: one to the dialogue module and one to the   }
{ module modeling the interface with the underlying association.  The        }
{ latter is not fully specified since that aspect is a local matter.  Only   }
{ the service it provides are described (in a module header definition).     }
{ When the sao module is destroyed, the interface to the association is also }
{ destroyed.  The actual association it represents may, however, survive the }
{ dialogue destruction and be released to the pool.  Access to the pool of   }
{ associations maintained by the site is provided by Estelle primitives      }
{ whose implementation is also a local matter.                               }
{                                                                            }
{ The sao module is created during dialogue establishment. If it is intended }
{ to support a superior dialogue, the identifier of the association on which }
{ the corresponding primitive was received is passed to the MACF, dialogue,  }
{ and sao module.  The latter gets the attributes of this association from   }
{ the site using a primitive function call.  Cases where the site rejects    }
{ the association establishement are handled at the level of the pool        }
{ When the sao module is intended to support a subordinate module, it must   }
{ first get an association (either from the pool or establish a new one).    }
{                                                                            }
{ The SAO module has two interaction points with the dialogue: one for AF,   }
{ presentation, and CCR primitives, and one for SAF primitives.  However, AF }
{ and SAF requests are received in the same queue.  Indeed, SAF service      }
{ requests are always the last thing the macf sends on an association, and   }
{ the only effect they have is that when some service requests ahead of them }
{ in the queue are processed, receipt of a SAF detach request determines     }
{ whether a detach indication is sent to the dialogue/channel or not.  The   }
{ way this is implemented in Estelle is to require that the dialogue/channel }
{ disconnect its interaction point from the sao module when it sends a       }
{ SAF_DETACH_req.  The net result is that the sao can send a SAF_DETACH_ind  }
{ regardless of whether a SAF_detach_ind is in the queue or not (if it is,   }
{ the SAF_ASSOCIATION_LOST indication will be lost anyway).                  }
{                                                                            }
{ The dialogue has access to attributes of the association and               }
{ can lock the presentation queue in the last phase of commitment.           }
{ ************************************************************************** }
 module sao_m activity(
    sup: boolean;            { subordinate dialogue }
    aId: assocIdType;    { identifier of assigned association if not sup }
    beginDialPars: afBeginDialReqType; { for subordinate dialogues }
    dial: boolean            { association supporting a dialogue } );
  ip
   af: AF_Service( provider) common queue;
   saf: SAF_Service( provider) common queue;

  export
   locked: boolean;    { is the presentation queue locked.              }
   dead: boolean;      { is module waiting for parent to destroy it; i.e.,  }
                       { has the underlying association been released/destroyed}
   Aahr: recoveryContextHandleType; { handle for association }
   Ahr: boolean;       { was recovery handle received on the association }
   Atokx: boolean;     { tokens expected }
   Abf: boolean;       { begin fear condition }
   Atppm: boolean;     { association supporting a dialogue (not a channel) }
   Atwr: boolean;    { is this association supporting two-way recovery }
   assocId:  assocIdType; { id of association }
end { module sao_m header definition };
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
body sao_b for sao_m;

{************************************************************************ }
{                                                                         }
{ SACF states and local variables                                         }
{                                                                         }
{ ************************************************************************}

state
  ESTABLISHED,              { association available for subordinate dialogue }
  FREE,          { dial to superior - association is available - waiting for }
                 { dial indication or bid indication.                        }
  STRAY,         { begin dialogue request was invoked - waiting for confirm }
  BIDDING,                { loser waiting for bid confirm                   }
  BID_CNF_RCVD,           { loser bid successfully                          }
  BID_IND_RCVD, { winner side - bid indication pending - waiting for token }
  BID_IND_ACCEPTED,                          { winner side - bid was accepted }
  BUSY,                                         { dialogue fully established }
  RB_IND_EXPECTED,                      { waiting for rollback ind from peer }
  BEGIN_EXPECTED,                { waiting for begin ind from peer to detach }
  RB_CNF_EXPECTED,        { waiting for rollback confirm from peer to detach }
  RELEASED;                                                 { sacf is released }

 var
  savedBeginDialParam: afBeginDialReqType;

  Abtr: boolean; { dialogue abort with begin transaction reject diagnostic }
  savedAbortPars: tpAbortRIType;
  Adru: boolean; { dialogue reject by user, mapped to rollbackRI }
  savedBegDialRC: tpBeginDialRCType;

{************************************************************************}
{                                                                         }
{ Interface with association.                                             }
{                                                                         }
{ When created with an association Id, the interface module is assumed to  }
{ connect to the association and retrieve its attributes.                 }
{                                                                         }
{ ************************************************************************}

  module assoc_m activity( assocId: assocIdType);
     ip
        access: SACF_lower( provider) individual queue;
     export
        Aahr: recoveryContextHandleType;
        Abm: boolean;
        Adc: correlatorType;
        Abf: boolean;
        Ahr: boolean;
        Alpi: correlatorType;
        Anfd: boolean;
        Atokr: boolean;
        Atokx: boolean;
        Aw: boolean;
  end;

  body assoc_b for assoc_m; external;{ implementation of interface is a local issue
}

  modvar
    assoc: assoc_m;  { assigned association }
  ip
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
peer : SACF_lower( user) individual queue; { assoc to association }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ ************* initialization of SACF ********************************* }
{ ********************************************************************** }

  initialize
    provided (sup)
      to FREE
         begin { create interface to association on which ind. was received}
           assocId := aId;
           init assoc with assoc_b( assocId); { will retrieve attributes }
           connect peer to assoc.access;
           Atppm := dial;
           Ahr := assoc.Ahr;
           Aahr := assoc.Aahr;
           Abf := assoc.Abf;
           Atokx := assoc.Atokx;
           Abtr := false;
           Adru := false;
         end;

    provided otherwise  { get new or used association from pool }
      to ESTABLISHED
        begin
          assocId := getPoolAssocId( beginDialPars);
          init assoc with assoc_b( assocId); { will retrieve attributes }
          connect peer to assoc.access;
          Atppm := dial;
          Ahr := assoc.Ahr;
          Aahr := assoc.Aahr;
          Abf := assoc.Abf;
          Atokx := assoc.Atokx;
          Abtr := false;
          Adru := false;
        end;

{*********     SAF service primitives ************************************ }
{********************************************************************** }

 trans
   when saf.SAF_DETACH_ASSOCIATION_req
     from STRAY, BID_CNF_RCVD, BUSY
       provided (param.status = free_assoc)
          to RELEASED
            var tokPars: pTokenGiveReqType;
                tokRI: tpTokenGiveRIType;
            begin
              {+ RETTOKEN +}
              if (not assoc.Aw and hasToken(assocId)) then
                 begin
                   tokRI.reason := regular;
                   encTokenGiveRI( tokRI, tokPars.userData);
                   output peer.P_TOKEN_GIVE_req( tokPars);
                 end;
              assoc.Abf := Abf;
              assoc.Atokx := Atokx;
              releasePoolAssoc( assocId);
              dead := true
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    from STRAY, BUSY
      provided (param.status = begin_fear)
         to RELEASED
           var tokPars: pTokenGiveReqType;
               tokRI: tpTokenGiveRIType;
           begin
             {+ RETTOKEN +}
             if (not assoc.Aw and hasToken(assocId)) then
               begin
                 tokRI.reason := regular;
                 encTokenGiveRI( tokRI, tokPars.userData);
                 output peer.P_TOKEN_GIVE_req( tokPars);
               end;
             assoc.Abf := true;
             assoc.Atokx := Atokx;
             releasePoolAssoc( assocId);
             dead := true
           end;

    from BUSY
      provided (param.status = rollback_cnf_expected)
        to RB_CNF_EXPECTED
          begin
          end;

      provided (param.status = c_begin_expected)
        to BEGIN_EXPECTED
          begin
          end;

      provided (param.status = rollback_ind_expected)
        to RB_IND_EXPECTED
          begin
          end;


{ ***   dialogue establishment; initiator (superior) side *************  }
{                                                                         }
{ - if the superior is the contention winner, then it may have to wait   }
{   for the token or for the purging of the begin fear before sending     }
{   the tp_begin_dialogue_RI pdu;  in this case, the module remains in    }
{   ESTABLISHED state and as a consequence all further requests are       }
{   queued (until conditions change, i.e., receives token or purges      }
{   begin fear;                                                           }
{                                                                         }
{ - if the superior is the contention winner or if conditions are such   }
{   that it does not need nor want to bid, the begin dialogue APDU is    }
{   sent to the peer.                                                     }
{                                                                         }
{ - If the superior is the contention loser and the commit functional    }
{   is selected, further requests are queued pending the receipt of       }
{   tokens;                                                               }
{                                                                         }
{ - if the superior is the contention loser and needs to bid (possibly   }
{   based on a local condition), then it saves the begin dial req para-   }
{   meters, issues a bid request, and enters the BIDDING state; all       }
{   further requests are queued until the bid_cnf is received             }
{                                                                         }
{ ***********************************************************************  }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { handling of the begin dialogue request received upon creation  }
  from ESTABLISHED
    provided ((assoc.Aw and not assoc.Abf)
        or (not assoc.Aw and
            not (assoc.Abm or assoc.Abf or wantToBid)))
      to STRAY
        var beginDialRI: tpBeginDialRIType;
        begin
          {+ VAdcNEW +}
          assoc.Adc := newDialChannelCorrelator;
          {+ SETCORR +}
          mapBeginDialReqAFtoTP( beginDialPars, beginDialRI);
          beginDialRI.dialChanCorrel := assoc.Adc;
          {+ VAlpiCORR or SETLPI +}
          if (assoc.Aw) then
            assoc.Alpi := assoc.Adc
          else if (assoc.Anfd) then
            beginDialRI.lastPartnerId := assoc.Alpi;
          {+ VafndT if winner +}
          if (assoc.Aw) then
            assoc.Anfd := true;
          {+ PASSTHRU +}
          output peer.TP_BEGIN_DIALOGUE_RI( beginDialRI);
        end;

  from ESTABLISHED
    provided (not assoc.Aw and (assoc.Abm or assoc.Abf or wantToBid))
      to BIDDING
        var bidRI: tpBidRIType;
        begin
          {+ BIDREQ +}
          if (assoc.Anfd) then
            bidRI.lastPartnerId := assoc.Alpi;
          bidRI.ccrTokenRequested := beginDialPars.beginTransaction;
          output peer.TP_BID_RI( bidRI);
          assoc.Anfd := true;
          {+ for VadcNEW, SETCORR, VaqT, QUEUE - see BIDDING state once +}
          {+ bid confirm is received                                     +}
        end;

 from BIDDING
   when peer.TP_BID_RC
     provided (not locked and (param.result=accepted))
       to BID_CNF_RCVD
          var beginDialRI: tpBeginDialRIType;
          begin
            {+ VadcNEW +}
            assoc.Adc := newDialChannelCorrelator;
            {+ SETCORR +}
            mapBeginDialReqAFtoTP( beginDialPars, beginDialRI);
            beginDialRI.dialChanCorrel := assoc.Adc;
            output peer.TP_BEGIN_DIALOGUE_RI( beginDialRI);
            {+ VAfF +}
            assoc.Abf := false;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      provided (not locked and (param.result<>accepted))
        to RELEASED
          var assocDetachPars: safAssocLostIndType;
          begin
            {+ SALi - if already detached, this has no effect +}
            assocDetachPars.status := dt_noReason;
            output saf.SAF_ASSOCIATION_LOST_ind( assocDetachPars);
            dead := true;
            {+ VAfF +}
            Abf := false;
            {+ RESETS +}
            assoc.Abf := Abf;
            assoc.Atokx := Atokx;
            releasePoolAssoc( assocId)
          end;

  trans
    when peer.TP_BEGIN_DIALOGUE_RC
      from STRAY
        provided (not locked and not match(param.dialChanCorrel,assoc.Adc))
          begin
            { discard }
          end;

        provided (not locked and match(param.dialChanCorrel,assoc.Adc))
          to BUSY
             var sentParam: afBeginDialCnfType;
             begin
               mapBeginDialRCtoAF( param, sentParam);
               output af.AF_BEGIN_DIALOGUE_cnf( sentParam)
             end;

      from FREE, BIDDING
        begin
          { discard }
        end;

{ *******  Dialogue establishment - responder (subordinate) **********    }
{ ************************************************************************ }

  trans
    when peer.TP_BID_RI
      from STRAY, RB_IND_EXPECTED, RB_CNF_EXPECTED
        provided (assoc.Aw)
          to SAME
             begin
               { collison; discard }
             end;

    when peer.TP_BID_RI
      from FREE
        provided (assoc.Aw
            and assoc.Anfd and not match(param.lastPartnerId,assoc.Alpi))
          to RELEASED
             begin
               { collison; discard }
               dead := true;
               assoc.Abf := Abf;
               assoc.Atokx := Atokx;
               releasePoolAssoc( assocId)
             end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    provided (assoc.Aw and not acceptBid
          and (not assoc.Anfd or match(param.lastPartnerId,assoc.Alpi)))
      to RELEASED
         var bidRC: tpBidRCType;
         begin
           {+ ABIDrsR +}
           bidRC.result := rejected_u;
           output peer.TP_BID_RC( bidRC);
           {+ return to pool +}
           dead := true;
           assoc.Abf := Abf;
           assoc.Atokx := Atokx;
           releasePoolAssoc( assocId)
         end;

when peer.TP_BID_RI
  from FREE
    provided (assoc.Aw
          and (not assoc.Anfd or match(param.lastPartnerId,assoc.Alpi))
          and not param.ccrTokenRequested
          and acceptBid)
      to BID_IND_ACCEPTED
         var bidRC: tpBidRCType;
         begin
           {+ ABIDrsA +}
           bidRC.result := accepted;
           output peer.TP_BID_RC( bidRC);
         end;

    provided (assoc.Aw
          and (not assoc.Anfd or match(param.lastPartnerId,assoc.Alpi))
          and param.ccrTokenRequested
          and acceptBid
          and (hasToken(assocId) or not delayBidUntilToken))
      to BID_IND_ACCEPTED
         var bidRC: tpBidRCType;
             tokenGivePars: pTokenGiveReqType;
             tokRI: tpTokenGiveRIType;
         begin
           {+ ABIDrsA +}
           bidRC.result := accepted;
           output peer.TP_BID_RC( bidRC);
           {+ PTOKGrq if we can, otherwise VAtokrT +}
           tokRI.reason := regular;
           encTokenGiveRI( tokRI, tokenGivePars.userData);
           if (hasToken(assocId)) then
             output peer.P_TOKEN_GIVE_req( tokenGivePars)
           else
             assoc.Atokr := true;
         end;

    provided (assoc.Aw
           and (not assoc.Anfd or match(param.lastPartnerId,assoc.Alpi))
           and param.ccrTokenRequested
           and acceptBid
           and not hasToken(assocId) and delayBidUntilToken)
      to BID_IND_RCVD
         begin
           {+ VAtokrT +}
           assoc.Atokr := true;
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans { decision to reject bid after initial intent to accept it }
  from BID_IND_RCVD
    provided rejectDelayedBid
      to RELEASED
        var bidRC: tpBidRCType;
        begin
          {+ ABIDrsR +}
          bidRC.result := rejected_u;
          output peer.TP_BID_RC( bidRC);
          {+ clean up stuff +}
          dead := true;
          assoc.Abf := Abf;
          assoc.Atokx := Atokx;
          releasePoolAssoc( assocId)
        end;

trans { receipt of token after bid }
  when peer.P_TOKEN_GIVE_ind
    from BID_IND_RCVD
      to BID_IND_ACCEPTED
        var bidRC: tpBidRCType;
            tokPars: tpTokenGiveRIType;
        begin
          {+ ABIDrsA +}
          bidRC.result := accepted;
          output peer.TP_BID_RC( bidRC);
          {+ PTOKGrq +}
          tokPars.reason := regular;
          encTokenGiveRI( tokPars, param.userData);
          output peer.P_TOKEN_GIVE_req( param);
          {+ VAtokrF +}
          assoc.Atokr := false;
        end;

    from BID_IND_ACCEPTED
      var tokPars: tpTokenGiveRIType;
      begin
        if (assoc.Atokr) then
          begin
            {+ PTOKGrq +}
            tokPars.reason := regular;
            encTokenGiveRI( tokPars, param.userData);
            output peer.P_TOKEN_GIVE_req( param);
            {+ VAtokrF +}
            assoc.Atokr := false;
          end
        else
          Abf := false;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when peer.TP_BEGIN_DIALOGUE_RI
    from FREE
      provided (not locked and assoc.Aw and not assoc.Abm
          and (not assoc.Anfd or match( assoc.Alpi, param.dialChanCorrel))
            and associationReserved)
        to RELEASED
          var tpBDpars: tpBeginDialRCType;
          begin
            {+ ABDrsRPdAR +}
            tpBDPars.result := rejected_p;
            tpBDPars.diagnostic := dr_associationReserved;
            output peer.TP_BEGIN_DIALOGUE_RC( tpBDPars);
            {+ cleanup +}
            dead := true;
            assoc.Abf := Abf;
            assoc.Atokx := Atokx;
            releasePoolAssoc( assocId)
          end;

      provided (not locked and assoc.Aw and not assoc.Abm and assoc.Anfd
            and not match(assoc.Alpi,param.dialChanCorrel))
        to RELEASED
          begin
            { suppress the primitive - clean up }
            dead := true;
            assoc.Abf := Abf;
            assoc.Atokx := Atokx;
            releasePoolAssoc( assocId)
          end;

trans
  when peer.TP_BEGIN_DIALOGUE_RI
    from FREE
      provided (not locked and (not Assoc.Aw
            or (not associationReserved and not assoc.Abm and
          (not assoc.Anfd or match(assoc.Alpi,param.dialChanCorrel)))))
        to BUSY
          var sentParam: afBeginDialIndType;
              tpKeepToken: tpTokenGiveRIType;
              tokPars: pTokenGiveReqType;
          begin
            {+ VAdcCORR +}
            assoc.Adc := param.dialChanCorrel;
            {+ PASSTHRU - see tppm for CREATEMACF +}
            mapBeginDialRItoAF( param, sentParam);
            output af.AF_BEGIN_DIALOGUE_ind( sentParam);
            if (assoc.Aw and param.functionalUnits.commit_fu) then
              if (hasToken(assocId)) then
                begin
                  {+ AKTrq +}
                  tpKeepToken.reason := keep;
                  tpKeepToken.correlator := assoc.Adc;
                  encTokenGiveRI( tpKeepToken, tokPars.userData);
                  output peer.P_TOKEN_GIVE_req( tokPars);
                end
              else {+ VAtokrT +}
                assoc.Atokr := true
            else if (not assoc.Aw) then
                {+ VAlpiCORR +}
                assoc.Alpi := assoc.Adc;
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    from BID_IND_ACCEPTED
      provided (not locked)
        to BUSY
           var sentParam: afBeginDialIndType;
           begin
             {+ VAdcCORR +}
             assoc.Adc := param.dialChanCorrel;
             {+ PASSTHRU +}
             mapBeginDialRItoAF( param, sentParam);
             output af.AF_BEGIN_DIALOGUE_ind( sentParam);
           end;

    from STRAY
      provided not locked and assoc.Aw and not assoc.Abm
        begin
          { collision on winner side - discard }
        end;

trans
  when peer.TP_BEGIN_DIALOGUE_RI
    from STRAY, BIDDING
      provided not locked and not assoc.Aw
        to RELEASED
           var sentParam: afBeginDialIndType;
               assocDetachPars: safAssocLostIndType;
           begin
            { collision on loser side - detach from current MACF -          }
            { the dialogue indication will be pending on the association    }
            { after it is released to the pool, leading the site to create}
                                                                            }
            { a new macf.
             {+ VAnfdT +}
             assoc.Anfd := true;
             {+ SALi - noeffect if already detached +}
             assocDetachPars.status := dt_noReason;
             output saf.SAF_ASSOCIATION_LOST_ind( assocDetachPars);
             {+ ValpiCORR - VAdcCORR - new macf +}
             assoc.Alpi := param.dialChanCorrel;
             assoc.Adc := param.dialChanCorrel;
             makeBeginDialIndPending( assocId, param);
             {+ clean up +}
             dead := true;
             assoc.Abf := Abf;
             assoc.Atokx := Atokx;
             releasePoolAssoc( assocId)
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  trans
    when af.AF_BEGIN_DIALOGUE_rsp
      from BUSY
        var rollbackRIpars: cRollbackReqType;
            rollbackRCpars: cRollbackRspType;
            beginDialRC: tpBeginDialRCType;
        begin
          mapBeginDialRspAFtoTP( param, beginDialRC);
          {+ SETCORR +}
          beginDialRC.dialChanCorrel := assoc.Alpi;
          case param.mapping of
            dataRI:
              output peer.TP_BEGIN_DIALOGUE_RC( beginDialRC );
            rollbackRI:
              begin
                encBeginDialRC( beginDialRC, rollbackRIpars.userData);
                if (param.result=rejected_u) then
                  begin
                    Adru := true;
                    savedBegDialRC := beginDialRC;
                  end;
                output peer.C_ROLLBACK_req( rollbackRIpars);
              end;
            rollbackRC:
              begin
                encBeginDialRC( beginDialRC, rollbackRCpars.userData);
                output peer.C_ROLLBACK_rsp( rollbackRCpars);
              end;
          end;
        end;

{ ******** Ending dialogue ************************************************ }
{ ********************************************************************** }
   trans
     when af.AF_END_DIALOGUE_req
      from STRAY, BID_CNF_RCVD, BUSY
        begin
          output peer.TP_END_DIALOGUE_RI( param)
        end;

     when peer.TP_END_DIALOGUE_RI
        provided not locked
          from FREE, BIDDING, STRAY
            begin
            end;

          from BUSY
            begin
              output af.AF_END_DIALOGUE_ind( param);
            end;

     when af.AF_END_DIALOGUE_rsp
       from BUSY
        begin
          output peer.TP_END_DIALOGUE_RC( param)
        end;

     when peer.TP_END_DIALOGUE_RC
       provided not locked
         from FREE, BIDDING, STRAY
           begin
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        from BUSY
          begin
            output af.AF_END_DIALOGUE_cnf( param);
          end;

{ *******  U_ERROR ************************************************  }
{ ********************************************************************* }

  trans
    when af.AF_U_ERROR_req
      from STRAY, BID_CNF_RCVD, BUSY
        begin
          output peer.TP_U_ERROR_RI( param)
        end;

    when peer.TP_U_ERROR_RI
      provided not locked
        from FREE, BIDDING, STRAY, RB_IND_EXPECTED, RB_CNF_EXPECTED
          begin
          end;

        from BUSY
          begin
            output af.AF_U_ERROR_ind( param)
          end;

    when af.AF_U_ERROR_rsp
      from BUSY
        begin
          output peer.TP_U_ERROR_RC( param)
        end;

    when peer.TP_U_ERROR_RC
      provided not locked
        from FREE, BIDDING, STRAY
          begin
          end;

        from BUSY
          begin
            output af.AF_U_ERROR_cnf( param)
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ *******  AF_ABORT ********************************************************  }
{ ********************************************************************** }

    when af.AF_ABORT_req
      from STRAY, BID_CNF_RCVD, BUSY
        var abortRIpdu: tpAbortRIType;
            abortRIpars: aAbortReqType;
            commitRIpars: cCommitReqType;
            commitRCpars: cCommitRspType;
            rollbackRIpars: cRollbackReqType;
            rollbackRCpars: cRollbackRspType;
      begin
        mapAbortReqAFtoTP( param, abortRIpdu);
        case param.mapping of
          dataRI:
            output peer.TP_ABORT_RI( abortRIpdu);
          commitRI:
            begin
              encAbortRI( abortRIpdu, commitRIpars.userData);
              output peer.C_COMMIT_req( commitRIpars);
            end;
          commitRC:
            begin
              encAbortRI( abortRIpdu, commitRCpars.userData);
              output peer.C_COMMIT_rsp( commitRCpars);
            end;
          rollbackRI:
            begin
              encAbortRI( abortRIpdu, rollbackRIpars.userData);
              Abtr := (param.diagnostic=ab_beginTransactionReject);
              if (Abtr) then savedAbortPars := abortRIpdu;
              output peer.C_ROLLBACK_req( rollbackRIpars);
             end;
          rollbackRC:
            begin
              encAbortRI( abortRIpdu, rollbackRCpars.userData);
              output peer.C_ROLLBACK_rsp( rollbackRCpars);
            end;
          abortRI:
            begin
              encAbortRI( abortRIpdu, abortRIpars.userData);
              output peer.A_ABORT_req( abortRIpars);
              dead := true;
            end;
        end;
      end;

    when peer.TP_ABORT_RI
      provided not locked
        from FREE, BIDDING, STRAY, RB_IND_EXPECTED, RB_CNF_EXPECTED
          begin
          end;

        from BEGIN_EXPECTED
          to RELEASED
            begin
              dead := true;
              assoc.Abf := Abf;
              assoc.Atokx := Atokx;
              releasePoolAssoc( assocId)
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        from BUSY
          var sentPars: afAbortIndType;
          begin
            mapAbortRItoAF( param, sentPars);
            output af.AF_ABORT_ind( sentPars)
          end;

{ *******  AF_G_CONTROL/AF_REQ_CONTROL ********************************** }
{ ************************************************************************ }

  trans
    when af.AF_GRANT_CONTROL_req
      from STRAY, BUSY, BID_CNF_RCVD
        begin
          output peer.TP_GRANT_CONTROL_RI( param)
        end;

    when peer.TP_GRANT_CONTROL_RI
      provided not locked
        from FREE, BIDDING, STRAY
          begin
          end;

        from BUSY
          begin
            output af.AF_GRANT_CONTROL_ind( param);
          end;

    when af.AF_REQUEST_CONTROL_req
      from BUSY, STRAY, BID_CNF_RCVD
        begin
          output peer.TP_REQUEST_CONTROL_RI( param)
        end;

    when peer.TP_REQUEST_CONTROL_RI
      provided not locked
        from FREE, BIDDING, STRAY
          begin
            { suppress }
          end;

        from BUSY
          begin
            output af.AF_REQUEST_CONTROL_ind( param);
          end;

{ *******  HANDSHAKE ************************************************** }
{ ************************************************************************ }

  trans
    when af.AF_HANDSHAKE_req
      from STRAY, BID_CNF_RCVD, BUSY
        begin
          output peer.TP_HANDSHAKE_RI( param)
        end;

    when peer.TP_HANDSHAKE_RI
      provided not locked
        from FREE, BIDDING, STRAY
          begin
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when peer.TP_HANDSHAKE_RI
  provided not locked
    from BUSY
      begin
        output af.AF_HANDSHAKE_ind( param)
      end;

when af.AF_HANDSHAKE_rsp
  from BUSY
    begin
      output peer.TP_HANDSHAKE_RC( param)
    end;

when peer.TP_HANDSHAKE_RC
  provided not locked
    from FREE, BIDDING, STRAY
      begin
      end;

    from BUSY
      begin
        output af.AF_HANDSHAKE_cnf( param)
      end;

when af.AF_HANDSHAKE_GC_req
  from STRAY, BID_CNF_RCVD, BUSY
    begin
      output peer.TP_HANDSHAKE_GC_RI( param)
    end;

when peer.TP_HANDSHAKE_GC_RI
  provided not locked
    from FREE, BIDDING, STRAY
      begin
      end;

    from BUSY
      begin
        output af.AF_HANDSHAKE_GC_ind( param)
      end;

when af.AF_HANDSHAKE_GC_rsp
  from BUSY
    begin
      output peer.TP_HANDSHAKE_GC_RC( param)
    end;

when peer.TP_HANDSHAKE_GC_RC
  provided not locked
    from FREE, BIDDING, STRAY
      begin
      end;

    from BUSY
      begin
        output af.AF_HANDSHAKE_GC_cnf( param)
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ *******  AF_DEFER        ********************************************  }
{ ************************************************************************ }

  trans
    when af.AF_DEFER_req
      from STRAY, BID_CNF_RCVD, BUSY
        begin
          output peer.TP_DEFER_RI( param)
        end;

    when peer.TP_DEFER_RI
      provided not locked
        from BUSY
          begin
            output af.AF_DEFER_ind( param)
          end;

{ *******  AF_PREPARE      ********************************************  }
{ ************************************************************************ }

  trans
    when af.AF_PREPARE_req
      from BUSY, STRAY, BID_CNF_RCVD
        var cPreparePars: cPrepareReqType;
        begin
          encPrepRI( param, cPreparePars.userData);
          output peer.C_PREPARE_req( cPreparePars)
        end;

    when peer.C_PREPARE_ind
      provided not locked
        from BUSY
          begin
            output af.AF_PREPARE_ind( extractPrepareRI( param.userData))
          end;

{ *******  AF/C RECOVER  ********************************************  }
{ ************************************************************************ }
  trans
    when af.C_RECOVER_req
      from BUSY
        provided (hasToken(assocId))
          begin
            output peer.C_RECOVER_req( param);
          end;

    when af.AF_RECOVER_req
      from BUSY
        provided (hasToken(assocId))
          var recoverReqPars: cRecoverReqType;
              tpRecoverRIpdu: tpRecoverRIType;
       begin
        tpRecoverRIpdu.recoveryContextHandle := param.recoveryContextHandle;
        case param.mapping of
            recoverComRI: recoverReqPars.recoveryState := commitState;
           recoverRdyRI: recoverReqPars.recoveryState := readyState;
          end;
        recoverReqPars.atomicActionId := param.atomicActionId;
        recoverReqPars.branchId := param.branchId;
        encRecRI( tpRecoverRIpdu, recoverReqPars.userData);
        output peer.C_RECOVER_req( recoverReqPars);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    when peer.C_RECOVER_ind
      provided not locked
        from BUSY
          var recoverIndPars: afRecoverIndType;
              tpRecoverRIpdu: tpRecoverRIType;
          begin
            if ( embeddedTPrecoverRI( param.userData)) then
              begin
                tpRecoverRIpdu := extractRecoverRI( param.userData);
                if (param.recoveryState = readyState) then
                  recoverIndPars.mapping := recoverRdyRI
                else
                  recoverIndPars.mapping := recoverComRI;
                recoverIndPars.recoveryContextHandle :=
                                        tpRecoverRIpdu.recoveryContextHandle;
                recoverIndPars.atomicActionId := param.atomicActionId;
                recoverIndPars.branchId := param.branchId;
                output af.AF_RECOVER_ind( recoverIndPars);
              end
            else
              output af.C_RECOVER_ind( param)
          end;

    when af.C_RECOVER_rsp
      from BUSY
       begin
         output peer.C_RECOVER_rsp( param)
       end;

    when peer.C_RECOVER_cnf
      provided not locked
        from BUSY
           var heurRepPars: afHeuristicIndType;
               embeddedHeurRI: tpHeuristicRepRIType;
        begin
          if (embeddedTPheurRI( param.userData)) then
            begin
              embeddedHeurRI := extractHeurRI( param.userData);
              heurRepPars.mapping := recoverDoneRC;
              heurRepPars.heuristicReport :=
                                        embeddedHeurRI.heuristicReport;
              heurRepPars.branchId := param.branchId;
              heurRepPars.atomicActionId := param.atomicActionId;
              output af.AF_HEURISTIC_ind( heurRepPars);
            end
          else output af.C_RECOVER_cnf( param);
        end;

{ *******  U_ASE                ***************************************** }
{ ********************************************************************** }

     when af.U_ASE_req
       from BUSY, STRAY, BID_CNF_RCVD
           begin
             output peer.U_ASE_req( param)
           end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    when peer.U_ASE_ind
      provided not locked
        from FREE, STRAY, BIDDING
          begin
            { discard }
          end;

        from BUSY
          begin
            output af.U_ASE_ind( param)
          end;

{ *******  HEURISTIC_REP       ***************************************** }
{ ************************************************************************ }

  trans
    when af.AF_HEURISTIC_req
      from BUSY
        var commitRCPars: cCommitRspType;
            rollbackRCPars: cRollbackRspType;
            rollbackRIPars: cRollBackReqType;
            recoverRCPars: cRecoverRspType;
            tpHeuristicRIpdu:  tpHeuristicRepRIType;
      begin
        tpHeuristicRIpdu.heuristicReport := param.heuristicReport;
        case param.mapping of
          commitRC:
              begin
                encHeurRI( tpHeuristicRIpdu, commitRCPars.userData);
                output peer.C_COMMIT_rsp( commitRCPars);
              end;
          rollbackRC:
              begin
                encHeurRI( tpHeuristicRIpdu, rollbackRCPars.userData);
                output peer.C_ROLLBACK_rsp( rollbackRCPars);
              end;
          rollbackRI:
              begin
                encHeurRI( tpHeuristicRIpdu, rollbackRIPars.userData);
                output peer.C_ROLLBACK_req( rollbackRIPars);
              end;
          recoverDoneRC:
              begin
                encHeurRI( tpHeuristicRIpdu, recoverRCPars.userData);
                recoverRCPars.recoveryState := doneState;
                recoverRCPars.atomicActionId := param.atomicActionId;
                recoverRCPars.branchId := param.branchId;
                output peer.C_RECOVER_rsp( recoverRCPars);
              end;
        end;
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    when af.AF_ABORT_HEURISTIC_req
      from BUSY
        var commitRCPars: cCommitRspType;
            rollbackRCPars: cRollbackRspType;
            rollbackRIPars: cRollBackReqType;
            recoverRCPars: cRecoverRspType;
            tpHeuristicRIpdu:  tpHeuristicRepRIType;
            tpAbortRIpdu: tpAbortRIType;
            userData: userDataType;
      begin
        tpAbortRIpdu.atype := userAbort;
        tpAbortRIpdu.userData := param.userData;
        tpHeuristicRIpdu.heuristicReport := param.heuristicReport;
        userData := concatenate( tpAbortRIpdu, tpHeuristicRIpdu);
        case param.mapping of
          commitRC:
              begin
                commitRCPars.userData := userData;
                output peer.C_COMMIT_rsp( commitRCPars);
              end;
          rollbackRC:
              begin
                rollbackRCPars.userData := userData;
                output peer.C_ROLLBACK_rsp( rollbackRCPars);
              end;
          rollbackRI:
              begin
                rollbackRIPars.userData := userData;
                output peer.C_ROLLBACK_req( rollbackRIPars);
              end;
        end;
      end;

{ *******  C_ROLLBACK      ************************************  }
{ ************************************************************************ }

trans
  when af.C_ROLLBACK_req
     from STRAY, BUSY, BID_CNF_RCVD
       begin
         output peer.C_ROLLBACK_req( param)
       end;

  when af.C_ROLLBACK_rsp
    from BUSY
       begin
         output peer.C_ROLLBACK_rsp( param)
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when peer.C_ROLLBACK_ind
  provided not locked
    from BUSY, BID_CNF_RCVD, STRAY
      to BUSY
         var embeddedAbortRI: tpAbortRIType;
             embeddedHeurRI: tpHeuristicRepRIType;
             embeddedBeginDialRC: tpBeginDialRCType;
             abortHeurPars: afAbortAndHeuristicIndType;
             abortPars: afAbortIndType;
             heurPars: afHeuristicIndType;
             beginDialPars: afBeginDialCnfType;
       begin
        if (embeddedTPHeurAndAbortRI( param.userData)) then
          begin
            embeddedAbortRI := extractAbortRI( param.userData);
            embeddedHeurRI := extractHeurRI( param.userData);
            abortHeurPars.heuristicReport := embeddedHeurRI.heuristicReport;
            abortHeurPars.mapping := rollbackRI;
            output af.AF_ABORT_HEURISTIC_ind( abortHeurPars);
          end
        else if (embeddedTPAbortRI( param.userData)) then
          begin
            embeddedAbortRI := extractAbortRI( param.userData);
            abortPars.mapping := rollbackRI;
            abortPars.aType := embeddedAbortRI.aType;
            output af.AF_ABORT_ind( abortPars);
          end
        else if (embeddedTPheurRI( param.userData)) then
          begin
            embeddedHeurRI := extractHeurRI( param.userData);
            heurPars.heuristicReport := embeddedHeurRI.heuristicReport;
            heurPars.mapping := rollbackRI;
            output af.AF_HEURISTIC_ind( heurPars);
          end
        else if (embeddedTPBeginDialRC( param.userData)) then
          begin
            embeddedBeginDialRC := extractBeginDialRC( param.userData);
            mapBeginDialRCtoAF( embeddedBeginDialRC, beginDialPars);
            beginDialPars.mapping := rollbackRI;
            output af.AF_BEGIN_DIALOGUE_cnf( beginDialPars);
          end
        else output af.C_ROLLBACK_ind( param);
       end;

       from RB_IND_EXPECTED
          to RELEASED
            var rollbackPars: cRollbackRspType;
                tokRI: tpTokenGiveRIType;
                tokPars: pTokenGiveReqType;
            begin {+ CRBrs +}
              output peer.C_ROLLBACK_rsp( rollbackPars);
              if (hasToken(assocId) and not assoc.Aw) then
                begin {+RETTOKEN+}
                  tokRI.reason := regular;
                  encTokenGiveRI( tokRI, tokPars.userData);
                  output peer.P_TOKEN_GIVE_req( tokPars);
                end;
              dead := true;
              assoc.Abf := Abf;
              assoc.Atokx := Atokx;
              releasePoolAssoc( assocId)
            end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
trans
  when peer.C_ROLLBACK_ind
    from RB_CNF_EXPECTED
      provided not locked
        to RELEASED
           var rollbackPars: cRollbackRspType;
               tokRI: tpTokenGiveRIType;
               tokPars: pTokenGiveReqType;
           begin
             if (Abtr) then
               begin
                 Abtr := false;
                 encAbortRI( savedAbortPars, rollbackPars.userData);
               end
             else if (Adru) then
               begin
                 Adru := false;
                 encBeginDialRC( savedBegDialRC, rollbackPars.userData);
               end;
             output peer.C_ROLLBACK_rsp(rollbackPars);
             {+ RETTOKEN +}
             if (hasToken(assocId) and not assoc.Aw) then
               begin
                 tokRI.reason := regular;
                 encTokenGiveRI( tokRI, tokPars.userData);
                 output peer.P_TOKEN_GIVE_req( tokPars);
               end;
             dead := true;
             assoc.Abf := Abf;
             assoc.Atokx := Atokx;
             releasePoolAssoc( assocId)
           end;

  when peer.C_ROLLBACK_CNF
    provided not locked
      from BUSY, STRAY, BID_CNF_RCVD
        to BUSY
          var embeddedAbortRI: tpAbortRIType;
              embeddedHeurRI: tpHeuristicRepRIType;
              embeddedBeginDialRC: tpBeginDialRCType;
              abortHeurPars: afAbortAndHeuristicIndType;
              abortPars: afAbortIndType;
              heurPars: afHeuristicIndType;
              beginDialPars: afBeginDialCnfType;
              ptokPars: pTokenGiveReqType;
      begin
        if (embeddedTPHeurAndAbortRI( param.userData)) then
          begin
            embeddedAbortRI := extractAbortRI( param.userData);
            embeddedHeurRI := extractHeurRI( param.userData);
            abortHeurPars.heuristicReport := embeddedHeurRI.heuristicReport;
            abortHeurPars.mapping := rollbackRC;
            output af.AF_ABORT_HEURISTIC_ind( abortHeurPars);
          end
        else if (embeddedTPAbortRI( param.userData)) then
          begin
            embeddedAbortRI := extractAbortRI( param.userData);
            abortPars.mapping := rollbackRC;
            abortPars.aType := embeddedAbortRI.aType;
            output af.AF_ABORT_ind( abortPars);
          end
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
          else if (embeddedTPheurRI( param.userData)) then
            begin
              embeddedHeurRI := extractHeurRI( param.userData);
              heurPars.heuristicReport := embeddedHeurRI.heuristicReport;
              heurPars.mapping := rollbackRC;
              output af.AF_HEURISTIC_ind( heurPars);
            end
          else if (embeddedTPBeginDialRC( param.userData)) then
            begin
              embeddedBeginDialRC := extractBeginDialRC( param.userData);
              mapBeginDialRCtoAF( embeddedBeginDialRC, beginDialPars);
              beginDialPars.mapping := rollbackRC;
              output af.AF_BEGIN_DIALOGUE_cnf( beginDialPars);
            end
          else output af.C_ROLLBACK_cnf( param);
        end;

    when peer.C_ROLLBACK_CNF
      provided not locked
        from RB_CNF_EXPECTED
          to RELEASED
            var tokPars: pTokenGiveReqType;
                tokRI: tpTokenGiveRIType;
            begin
              {+ RETTOKEN +}
              if (not assoc.Aw and hasToken(assocId)) then
                begin
                  tokRI.reason := regular;
                  encTokenGiveRI( tokRI, tokPars.userData);
                  output peer.P_TOKEN_GIVE_req( tokPars);
                end;
              Abtr := false;
              Adru := false;
              dead := true;
              assoc.Abf := Abf;
             assoc.Atokx := Atokx;
             releasePoolAssoc( assocId)
          end;

{ *******  C_COMMIT          ***************************************  }
{ ************************************************************** }

 trans
   from BUSY
     when af.C_COMMIT_req
       begin
         output peer.C_COMMIT_req( param)
       end;

     when af.C_COMMIT_C_BEGIN_req
       begin
         output peer.C_COMMIT_C_BEGIN_req( param)
       end;

     when peer.C_COMMIT_C_BEGIN_ind
       begin
         output af.C_COMMIT_C_BEGIN_ind( param)
       end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
   when af.C_COMMIT_rsp
     begin
       output peer.C_COMMIT_rsp( param)
     end;

trans
  when peer.C_COMMIT_ind
    from FREE, STRAY, BIDDING
      begin
      end;
    from BUSY
      provided not locked
        var abortIndPars: afAbortIndType;
            embeddedAbortPars: tpAbortRIType;
        begin
          if (embeddedTPAbortRI( param.userData)) then
             begin
               embeddedAbortPars := extractAbortRI( param.userData);
               abortIndPars.atype := embeddedAbortPars.atype;
               if (abortIndPars.atype=userAbort) then
                 abortIndPars.userData := embeddedAbortPars.userData
               else
                 abortIndPars.diagnostic := embeddedAbortPars.diagnostic;
               abortIndPars.mapping := commitRI;
               output af.AF_ABORT_ind( abortIndPars);
             end
          else output af.C_COMMIT_ind( param);
        end;

  when peer.C_COMMIT_cnf
    provided not locked
      from BUSY
        var embeddedAbortRI: tpAbortRIType;
            embeddedHeurRI: tpHeuristicRepRIType;
            abortHeurPars: afAbortAndHeuristicIndType;
            abortPars: afAbortIndType;
            heurPars: afHeuristicIndType;
      begin
        if (embeddedTPHeurAndAbortRI( param.userData)) then
          begin
            embeddedAbortRI := extractAbortRI( param.userData);
            embeddedHeurRI := extractHeurRI( param.userData);
            abortHeurPars.heuristicReport := embeddedHeurRI.heuristicReport;
            abortHeurPars.mapping := commitRC;
            output af.AF_ABORT_HEURISTIC_ind( abortHeurPars);
          end
        else if (embeddedTPAbortRI( param.userData)) then
          begin
            embeddedAbortRI := extractAbortRI( param.userData);
            abortPars.mapping := commitRC;
            abortPars.aType := embeddedAbortRI.aType;
            output af.AF_ABORT_ind( abortPars);
          end
        else if (embeddedTPheurRI( param.userData)) then
          begin
            embeddedHeurRI := extractHeurRI( param.userData);
            heurPars.heuristicReport := embeddedHeurRI.heuristicReport;
            heurPars.mapping := commitRC;
            output af.AF_HEURISTIC_ind( heurPars);
          end
        else output af.C_COMMIT_cnf( param);
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ *******  C_BEGIN and token control ********************************** }
{ ************************************************************************ }

  trans
   when af.C_BEGIN_req
     from STRAY, BIDDING, BID_CNF_RCVD, BUSY
       provided hasToken(assocId) { queued until token is rcvd }
         to SAME
           begin
             output peer.C_BEGIN_req( param)
           end;

   when peer.P_TOKEN_GIVE_ind
      provided (not embeddedTPpdu( param.userData) or
                regularTPTokenGiveRI(param.userData))
        from STRAY
          var tpTokRI: tpTokenGiveRIType;
          begin
           tpTokRI.reason := regular;
           encTokenGiveRI( tpTokRI, param.userData);
           if (not assoc.Aw) then
             output peer.P_TOKEN_GIVE_req( param);
           assoc.Abf := false;
          end;

   when peer.P_TOKEN_GIVE_ind
      provided (not embeddedTPpdu( param.userData) or
                regularTPTokenGiveRI(param.userData))
        from BIDDING
          var tpTokRI: tpTokenGiveRIType;
          begin
           tpTokRI.reason := regular;
           encTokenGiveRI( tpTokRI, param.userData);
           output peer.P_TOKEN_GIVE_req( param)
          end;

   when peer.P_TOKEN_GIVE_ind
      provided (not embeddedTPpdu( param.userData) or
                regularTPTokenGiveRI(param.userData) or
                keepTPTokenGiveRI(param.userData))
        from ESTABLISHED, FREE
          begin
           if (not assoc.Aw) then
             output peer.P_TOKEN_GIVE_req( param)
           else
             assoc.Abf := false;
          end;

   when peer.P_TOKEN_GIVE_ind
      provided (not embeddedTPpdu( param.userData) or
                regularTPTokenGiveRI(param.userData))
        from RB_CNF_EXPECTED
          begin
            assoc.Abf := false;
          end;

     provided (not embeddedTPpdu( param.userData))
       from BUSY
         begin
           {+ PASSTOKEN +}
           {** token must be passed to the U-ASE **}
         end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
when peer.P_TOKEN_GIVE_ind
  provided (embeddedTPpdu( param.userData) and
            regularTPTokenGiveRI(param.userData))
    from BID_CNF_RCVD
      begin
        assoc.Abf := false;
      end;

  provided (embeddedTPpdu( param.userData) and
            regularTPTokenGiveRI(param.userData) and
            assoc.Aw and assoc.Atokr)
    from BUSY
      var tpTokRI: tpTokenGiveRIType;
      begin
        tpTokRI.reason := keep;
        tpTokRI.correlator := assoc.Adc;
        encTokenGiveRI( tpTokRI, param.userData);
        output peer.P_TOKEN_GIVE_req( param);
        assoc.Atokr := false;
      end;

  provided (embeddedTPpdu( param.userData) and
            keepTPTokenGiveRI(param.userData) and
            not assoc.Aw)
    from BID_CNF_RCVD, BUSY
      var tpTokRI: tpTokenGiveRIType;
      begin
        tpTokRI := extractTPTokenGiveRI( param.userData);
        if (not match(tpTokRI.correlator,assoc.Adc)) then
          begin
            tpTokRI.reason := regular;
            encTokenGiveRI( tpTokRI, param.userData);
            output peer.P_TOKEN_GIVE_req( param);
          end
        else
          begin
            assoc.Abf := false;
            {+ PASSTOKEN +}
            {** token to the U-ASE **}
          end;
      end;

  provided (embeddedTPpdu( param.userData) and
            twrTPTokenGiveRI(param.userData))
    from BUSY
      var tpTokRI: tpTokenGiveRIType;
      begin
        tpTokRI := extractTPTokenGiveRI( param.userData);
        output af.AF_TOKEN_GIVE_ind( tpTokRI);
      end;

when peer.C_BEGIN_ind
  provided not locked
    from FREE, BEGIN_EXPECTED
      to RB_CNF_EXPECTED
          var rollbackPars: cRollbackReqType;
          begin
            assoc.Abf := false;
            output peer.C_ROLLBACK_req( rollbackPars)
          end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    provided not locked
      from BUSY
        begin
          output af.C_BEGIN_ind( param)
        end;

    provided not locked and not assoc.Aw and assoc.Abf
      from BIDDING, STRAY
        to RB_CNF_EXPECTED
           var rollbackPars: cRollbackReqType;
               detPars: safAssocLostIndType;
           begin
             detPars.status := dt_noReason;
             output saf.SAF_ASSOCIATION_LOST_ind( detPars);
             output peer.C_ROLLBACK_req( rollbackPars);
             assoc.Abf := false;
           end;

  when af.AF_TOKEN_GIVE_req
    from BUSY
      provided (hasToken(assocId))
        var tokPars: pTokenGiveReqType;
        begin
          encTokenGiveRI( param, tokPars.userData);
          output peer.P_TOKEN_GIVE_req( tokPars)
        end;

  when af.P_TOKEN_GIVE_req
    from BUSY
      begin
        output peer.P_TOKEN_GIVE_req( param)
      end;

  when peer.C_BEGIN_cnf
    provided not locked
      from BUSY
        begin
          output af.C_BEGIN_cnf( param)
        end;

trans
  from BUSY
    when af.C_BEGIN_rsp
      begin
        output peer.C_BEGIN_rsp( param)
      end;

{ ******* P_TOKEN_PLEASE                    ********************************** }
{ ************************************************************************** }

trans
  when af.AF_TOKEN_PLEASE_req
    from BUSY
      var tokPls: pTokenPleaseReqType;
      begin
        output peer.P_TOKEN_PLEASE_req(tokPls)
      end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
 trans
   when peer.P_TOKEN_PLEASE_ind
     from BUSY
       var tokPls: afTokenPleaseIndType;
       begin
         output af.AF_TOKEN_PLEASE_ind(tokPls)
       end;

   when peer.P_TOKEN_PLEASE_ind
     from FREE, STRAY, BIDDING
       begin
       end;

{ *******  C_READY_req/ind                ********************************* }
{ ************************************************************************** }

 trans
   from BUSY
     when af.C_READY_req
       begin
         output peer.C_READY_req( param)
       end;

     when peer.C_READY_ind
       begin
         output af.C_READY_ind( param)
       end;

{ *******  provider Abort             ********************************  }
{ ************************************************************************** }

 trans
   provided not locked
     when peer.A_ABORT_ind
       from STRAY, BIDDING, BID_CNF_RCVD, BUSY
         to RELEASED
           var abortPars: afAbortIndType;
           begin
             abortPars.mapping := abortRI;
             abortPars.aType := providerAbort;
             dead := true;
             output af.AF_ABORT_ind( abortPars)
           end;

       from FREE, ESTABLISHED, BID_IND_RCVD, BID_IND_ACCEPTED,
            RB_IND_EXPECTED, RB_CNF_EXPECTED, BEGIN_EXPECTED
         to RELEASED
           begin
             dead := true
           end;

 end; { of sao_b module body }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{** types.e: type definitions ***************************************}

{*******************************************************************}
{     type definitions for pdus and service primitive parameters    }
{                                                                   }
{ - Types of individual parameters or fields are left unspecified unless }
{   they are explicitely defined in the Englsih text ( we use the "..."  }
{   Estelle construct for that purpose)                             }
{                                                                   }
{ ***************************************************************   }

const

  MAX_CHANS = any integer;
  MAX_TPPMS = any integer;
  MAX_DIALS = any integer;
  recoverDelay = any integer;
  nullAssocId = 0;

type

  assocIdType = integer;          { local identifier for association }

  DIAL_IDX = 0.. MAX_DIALS;

  apTitleType = ...;              { see ISO/IEC 7498-3 }

  apiIdType = ...;                { see ISO/IEC 7498-3 }

  aeQualifierType = ...;          { see ISO/IEC 7498-3 }

  aeiIdType = ...;                { see ISO/IEC 7498-3 }

  aeTitleType = ...;                  { see ISO/IEC 8649 }

  aeInvocationIdType = ...;   { see ISO/IEC 8749 }

  apInvocationIdType = ...;   { see ISO/IEC 8749 }


  functionalUnitsType = record
       shared: boolean;
       hdsk_fu: boolean;
       case commit_fu: boolean of
            true: (chained: boolean);
            false:()
   end;

  qualityOfServiceType = ...; { see ISO/IEC SC 21 }

  applContextNameType = ...;  { see ISO/IEC 8649 }

  tpsuTitleType = ...;

  confirmationType = (always, negative);

  resultType = ( accepted, rejected_p, rejected_u, rejectedPermanent,
     rejectedTransient);

  resultSourceType = ( acseUser, asceProvider, presProvider);

  pAbortReasonType = ...;         { see ISO  8822 }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
beginDialDiagnosticType = ( dr_recipientUnknown,
   dr_recipientTpsuTitleUnknown,
   dr_tpsuNotAvailablePermanent, dr_tpsuNotAvailableTransient,
   dr_recipientTpsuTitleRequired, dr_functionalUnitNotSupported,
   dr_functionalUnitCombinationNotSupported, dr_noReason,
   dr_twoWayRecoveryNotSupported, dr_associationReserved);

pAbortDiagnosticType = ( ab_permanentFailure, ab_transientFailure,
   ab_beginTransactionReject, ab_endDialogueCollision,
   ab_beginTransactionEndDialCollision);

assocRejectDiagnosticType = ( ar_noReasonGiven, ar_noCommonACSEVersion,
   ar_applContextNameNotSupported, ar_callingApplTitleUnknown,
   ar_callingAEQualifierUnknown, ar_callingAPInvocationIdUnknown,
   ar_callingAEInvocationIdUnknown, ar_calledAPTitleUnknown,
   ar_calledQualifierUnknown, ar_calledAPInvocationIdUnknown,
   ar_calledAEInvocationIdUnknown);

urgencyType = (urgent, normal);

transactionState = ( logInitial, logFinal);

heuristicReportType = ( heuristicMix, heuristicHazard, hr_notPresent,
   heuristicFinal, heuristicInitial);

commitmentLevelType = ( commitment, none);

contentionWinnerType = ( initiator, recipient);

assignmentOfTokensType = ( requestorSide, recipientSide);

ACSEModes = ( normalMode, X_410_84);

assocEstablDiagnosticType = ( ccr2Version2NotAvailable);

channelUtilizationType = ( oneWayRecovery, twoWayRecovery);

mappingType = ( dataRI, rollbackRI, rollbackRC, abortRI, commitRI,
    commitRC, recoverDoneRC, recoverComRI, recoverRdyRI);

abortType = (userAbort, providerAbort);

deferredServiceType = ( endDialogue, grantControl);

recoveryStateType = ( readyState, commitState, unknownState, retryLater,
   doneState);

detachStatusType = ( free_assoc, rollback_cnf_expected, begin_fear,
   rollback_ind_expected, c_begin_expected);

detachDiagnosticType = ( dt_recipientUnknown, dt_tpsuNotAvailablePermanent,
   dt_tpsuNotAvailableTransient, dt_noReason);

dialType = ( dialogueUsage, channelUsage);

emptyParameterList = record    end;

userDataType = ...;

protocolVersionType = ...;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  recoveryContextHandleType = ...;

  correlatorType = ...;
  presAddressType = ...;                    { see ISO 8822 }

  presContextDefListType = ...;             { see ISO 8822 }

  defaultContextNameType = ...;             { see ISO 8822 }

  presRequirementType = ...;                { see ISO 8822 }

  sessionRequirementType = ...;             { see ISO 8822 }

  serialNbrType = ...;                      { see ISO 8822 }

  atomicActionPrefixType = ...;                  { see ISO/IEC 9804 }

  atomicActionSuffixType = ...;                  { see ISO/IEC 9804 }

  atomicActionIdType = record                    { see ISO/IEC 9804 }
        prefix: atomicActionPrefixType;
        suffix: atomicActionSuffixType;
   end;

  branchPrefixType = ...;                        { see ISO/IEC 9804 }

  branchSuffixType = ...;                        { see ISO/IEC 9804 }

  branchIdType = record                          { see ISO/IEC 9804 }
        prefix: branchPrefixType;
        suffix: branchSuffixType;
   end;

type

{ ***************  ACSE Service Primitives ******************************  }
{                                                                           }
{  *** for the types of individual parameters, refer to ISO 8649           }
{                                                                           }
{ ***************************************************************************  }


  aAssociateReqType = record                          { 8.2.1 }
       mode: ACSEModes;
       applicationContextName: applContextNameType;
       callingApTitle: apTitleType;
       callingAEQualifier: aeQualifierType;
       calledAPInvocationId: apInvocationIdType;
       calledAEInvocationId: aeInvocationIdType;
       callingAPInvocationId: apInvocationIdType;
       callingAEInvocationId: aeInvocationIdType;
       userInformation: userDataType;
       callingPresAddress: presAddressType;
       calledPresAddress: presAddressType;
       presContextDefList: presContextDefListType;
       defaultPresContextName: defaultContextNameType;
       qualityOfService: qualityOfServiceType;
       presRequirements: presRequirementType;
       sessionRequirements: sessionRequirementType;
       initialSynchroPointSerialNbr: serialNbrType;
       initialAssignmentOfTokens: assignmentOfTokensType;
  end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
aAssociateIndType = aAssociateReqType;

aAssociateRspType = record
      mode: ACSEModes;
      applicationContextName: applContextNameType;
      calledApTitle: apTitleType;
      respondingApTitle: apTitleType;
      respondingAETitle: aeTitleType;
      respondingAPInvocationId: apInvocationIdType;
      respondingAEInvocationId: aeInvocationIdType;
      userInformation: userDataType;
      result: resultType;
      diagnostic: assocRejectDiagnosticType;
      respondingPresAddress: presAddressType;
      presContextDefResultList: presContextDefListType;
      defaultPresContextResult: defaultContextNameType;
      qualityOfService: qualityOfServiceType;
      presRequirements: presRequirementType;
      sessionRequirements: sessionRequirementType;
      initialSynchroPointSerialNbr: serialNbrType;
      initialAssignmentOfTokens: assignmentOfTokensType;
 end;

aAssociateCnfType = record
      mode: ACSEModes;
      applicationContextName: applContextNameType;
      calledApTitle: apTitleType;
      respondingApTitle: apTitleType;
      respondingAETitle: aeTitleType;
      respondingAPInvocationId: apInvocationIdType;
      respondingAEInvocationId: aeInvocationIdType;
      userInformation: userDataType;
      result: resultType;
      resultSource: resultSourceType;
      diagnostic: assocRejectDiagnosticType;
      respondingPresAddress: presAddressType;
      presContextDefResultList: presContextDefListType;
      defaultPresContextResult: defaultContextNameType;
      qualityOfService: qualityOfServiceType;
      presRequirements: presRequirementType;
      sessionRequirements: sessionRequirementType;
      initialSynchroPointSerialNbr: serialNbrType;
      initialAssignmentOfTokens: assignmentOfTokensType;
 end;

aReleaseReqType = emptyParameterList;

aReleaseIndType = emptyParameterList;

aReleaseRspType = record
      result: boolean;
 end;

aReleaseCnfType = aReleaseRspType;

aAbortReqType = record
      userData: userDataType;
 end;

aAbortIndType = record
      userData: userDataType;
 end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  aPAbortIndType = record
       providerReason: pAbortReasonType;
   end;

{ ************** CCR Service Primitives ********************************* }
{ ************************************************************************ }

  cBeginReqType = record
       atomicActionId: atomicActionIdType;
       atomicBranchId: branchIdType;
   end;

  cCommitBeginReqType = record
       atomicActionId: atomicActionIdType;
       atomicBranchId: branchIdType;
   end;

  cBeginIndType = cBeginReqType;

  cBeginRspType = emptyParameterList;

  cBeginCnfType = emptyParameterList;

  cPrepareReqType = record
       userData: userDataType;
   end;

  cPrepareIndType = cPrepareReqType;

  cReadyReqType = emptyParameterList;

  cReadyIndType = cReadyReqType;

  cCommitReqType = record
       userData: userDataType;
   end;

  cCommitIndType = cCommitReqType;

  cCommitRspType = cCommitReqType;

  cCommitCnfType = cCommitReqType;

  cRollbackReqType = record
       userData: userDataType;
   end;

  cRollbackIndType = cRollbackReqType;

  cRollbackRspType = cRollbackReqType;

  cRollbackCnfType = cRollbackReqType;

  cRecoverReqType = record
       recoveryState: recoveryStateType;
       atomicActionId: atomicActionIdType;
       branchId: branchIdType;
       userData: userDataType;
   end;

  cRecoverIndType = cRecoverReqType;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  cRecoverRspType = cRecoverReqType;

  cRecoverCnfType = cRecoverReqType;

{ ** Presentation token control primitives ***************************** }

  pTokenPleaseReqType = record
        userData: userDataType;
      end;

  pTokenPleaseIndType = pTokenPleaseReqType;

  pTokenGiveReqType = record
        userData: userDataType;
      end;

  pTokenGiveIndType = pTokenGiveReqType;

{ ** TP APDUs ********************************************************* }

   tpBeginDialRIType = record
         functionalUnits: functionalUnitsType;
         dialChanCorrel: correlatorType;
         lastPartnerId: correlatorType;
         case dialType of
              dialogueUsage: (
                   initiatingTPSUtitle: tpsuTitleType;
                   recipientTPSUTtitle: tpsuTitleType;
                   beginTransaction: boolean;
                   confirmation: confirmationType;
                   userData: userDataType;
                  );
               channelUsage: (
                   channelUtilization: channelUtilizationType;
                  );
    end;

   tpBeginDialRCType = record
         result: resultType;
         diagnostic: beginDialDiagnosticType;
         dialChanCorrel: correlatorType;
         case dialType of
               dialogueUsage: (
                   functionalUnits: functionalUnitsType;
                   userData: userDataType;
                  );
               channelUsage: ( );
    end;

   tpBidRIType = record
         ccrTokenRequested: boolean;
         lastPartnerId: correlatorType;
    end;

   tpBidRCType = record
         result: resultType;
    end;

   tpEndDialRIType = record
         confirmation: boolean;
    end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
tpEndDialRCType = emptyParameterList;

tpUErrorRIType = emptyParameterList;

tpUErrorRCType = emptyParameterList;

tpAbortRIType = record
       case atype: abortType of
          userAbort: (
                userdata: userDataType;
             );
          providerAbort: (
                diagnostic: pAbortDiagnosticType;
             );
 end;


tpGrantControlRIType= emptyParameterList;

tpRequestControlRIType= emptyParameterList;

tpHandshakeRIType = record
      confirmationUrgency: urgencyType;
 end;

tpHandshakeRCType = emptyParameterList;


tpHandshakeGCRIType = tpHandshakeRIType;

tpHandshakeGCRCType = tpHandshakeRCType;

tpDeferRIType = record
       deferType: deferredServiceType;
 end;

tpPrepareRIType = record
       dataPermitted: boolean;
 end;

tpHeuristicRepRIType = record
       heuristicReport: heuristicReportType;
 end;

tpRecoverRIType = record
       recoveryContextHandle: recoveryContextHandleType;
 end;

tpTokenGiveRIType = record
       reason : (regular, keep, two_way_recovery);
       correlator: correlatorType;
 end;

tpTokenPleaseRIType = emptyParameterList;

tpKeepTokenRIType = record
       alpi: correlatorType;
 end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ * AF Service Primitive  Parameters ********************************** }

   safDetachAssocReqType = record
         status: detachStatusType;
    end;

   safAssocLostIndType = record
         status: detachDiagnosticType;
    end;

   afBeginDialReqType = record
         functionalUnits: functionalUnitsType;
         dialChanCorrel:  correlatorType;
         case dialType of
              dialogueUsage: (
                   initiatingTPSUtitle: tpsuTitleType;
                   recipientTPSUTtitle: tpsuTitleType;
                   lastPartnerId: correlatorType;
                   beginTransaction: boolean;
                   confirmation: confirmationType;
                   userData: userDataType;
                  );
              channelUsage: ( );
    end;

   afBeginDialIndType = afBeginDialReqType;

   afBeginDialRspType = record
         result: resultType;
         diagnostic: beginDialDiagnosticType;
         mapping: mappingType;
         case dialType of
               dialogueUsage: (
                   dialChanCorrel: correlatorType;
                   functionalUnits: functionalUnitsType;
                   userData: userDataType;
                  );
               channelUsage: ( );
    end;

  afBeginDialCnfType = record
         result: resultType;
         diagnostic: beginDialDiagnosticType;
         dialChanCorrel:  correlatorType;
         mapping: mappingType;
         case dialType of
              dialogueUsage: (
                   functionalUnits: functionalUnitsType;
                   userData: userDataType;
                  );
              channelUsage: ( );
    end;

 afEndDialReqType = tpEndDialRIType;

 afEndDialIndType = tpEndDialRIType;

 afEndDialRspType = tpEndDialRCType;

 afEndDialCnfType = tpEndDialRCType;

 afUErrorReqType = emptyParameterList;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
afUErrorIndType = emptyParameterList;

afUErrorRspType = emptyParameterList;

afUErrorCnfType = emptyParameterList;

afAbortReqType = record
        mapping: mappingType;
        case atype: abortType of
           userAbort: (
                 userData: userDataType;
              );
           providerAbort: (
                 diagnostic: pAbortDiagnosticType;
              );
  end;

afAbortIndType = afAbortReqType;

afGrantControlReqType = emptyParameterList;

afGrantControlIndType = emptyParameterList;

afRequestControlReqType = emptyParameterList;

afRequestControlIndType = emptyParameterList;

afHandshakeReqType = tpHandshakeRIType;

afHandshakeIndType = tpHandshakeRIType;

afHandshakeRspType = emptyParameterList;

afHandshakeCnfType = emptyParameterList;

afHandshakeGCReqType = tpHandshakeRIType;

afHandshakeGCIndType = tpHandshakeRIType;

afHandshakeGCRspType = emptyParameterList;

afHandshakeGCCnfType = emptyParameterList;

afDeferReqType = tpDeferRIType;

afDeferIndType = tpDeferRIType;

afPrepareReqType = tpPrepareRIType;

afPrepareIndType = tpPrepareRIType;

afHeuristicReqType = record
        heuristicReport: heuristicReportType;
        mapping: mappingType;
        atomicActionId: atomicActionIdType;
        branchId: branchIdType;
  end;

afHeuristicIndType = afHeuristicReqType;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  afAbortAndHeuristicReqType = record
         mapping: mappingType;
         userData: userDataType;
         heuristicReport: heuristicReportType;
   end;

  afAbortAndHeuristicIndType = afAbortAndHeuristicReqType;

  afRecoverReqType = record
         mapping: mappingType;
         recoveryContextHandle: recoveryContextHandleType;
         atomicActionId: atomicActionIdType;
         branchId: branchIdType;
   end;

  afRecoverIndType = afRecoverReqType;

  afTokenGiveReqType = tpTokenGiveRIType;

  afTokenPleaseReqType = tpTokenPleaseRIType;

  afTokenGiveIndType = tpTokenGiveRIType;

  afTokenPleaseIndType = tpTokenPleaseRIType;


{ * TP Service Primitives ********************************************** }

  tpBeginDialogueReqType = record
         initiatingTPSUTitle: tpsuTitleType;
         recipientAPTitle: apTitleType;
         recipientAPIId: apiIdType;
         recipientAEQualifier: aeQualifierType;
         recipientAEIId: aeiIdType;
         recipientTPSUTitle: tpsuTitleType;
         functionalUnits: functionalUnitsType;
         qualityOfService: qualityOfServiceType;
         applicationContextName: applContextNameType;
         beginTransaction: boolean;
         confirmation: confirmationType;
         userData: userDataType;
   end;

  tpBeginDialogueIndType = record
         initiatingApTitle: apTitleType;
         initiatingAPIId: apiIdType;
         initiatingAEQualifier: aeQualifierType;
         initiatingAEIId: aeiIdType;
         initiatingTPSUTitle: tpsuTitleType;
         functionalUnits: functionalUnitsType;
         beginTransaction: boolean;
         confirmation: confirmationType;
         userData: userDataType;
   end;

  tpBeginDialogueRspType = record
         result: resultType;
         userData: userDataType;
   end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
tpBeginDialogueCnfType = record
       functionalUnits: functionalUnitsType;
       result: resultType;
       diagnostic: beginDialDiagnosticType;
       userData: userDataType;
       rollback: boolean;
 end;

tpEndDialogueReqType = tpEndDialRIType;

tpEndDialogueIndType = tpEndDialogueReqType;

tpEndDialogueRspType = emptyParameterList;

tpEndDialogueCnfType = emptyParameterList;

tpUErrorReqType = emptyParameterList;

tpUErrorIndType = emptyParameterList;

tpUAbortReqType = record
       userData: userDataType;
 end;

tpUAbortIndType = record
       userData: userDataType;
       rollback: boolean;
 end;

tpPAbortIndType = record
       diagnostic: pAbortDiagnosticType;
       rollback: boolean;
 end;

tpGrantControlReqType = emptyParameterList;

tpGrantControlIndType = emptyParameterList;

tpRequestControlReqType= emptyParameterList;

tpRequestControlIndType= emptyParameterList;

tpHandshakeReqType = tpHandshakeRIType;

tpHandshakeIndType = tpHandshakeRIType;

tpHandshakeRspType = emptyParameterList;

tpHandshakeCnfType = emptyParameterList;

tpHandshakeGCReqType = tpHandshakeGCRIType;

tpHandshakeGCIndType = tpHandshakeGCReqType;

tpHandshakeGCRspType = emptyParameterList;

tpHandshakeGCCnfType = emptyParameterList;

tpBeginTransactionReqType = emptyParameterList;

tpBeginTransactionIndType = emptyParameterList;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  tpDeferredEndDialogueReqType = emptyParameterList;

  tpDeferredEndDialogueIndType = emptyParameterList;

  tpDeferredGCReqType = emptyParameterList;

  tpDeferredGCIndType = emptyParameterList;

  tpPrepareReqType = afPrepareIndType;

  tpPrepareIndType = tpPrepareReqType;

  tpReadyIndType = emptyParameterList;

  tpCommitReqType = emptyParameterList;

  tpCommitIndType = emptyParameterList;

  tpDoneReqType = record
          heuristicReport: heuristicReportType;
   end;

  tpCommitCompleteIndType = emptyParameterList;

  tpRollbackReqType = emptyParameterList;

  tpRollbackIndType = emptyParameterList;

  tpRollbackCompleteIndType = emptyParameterList;

  tpHeuristicReportIndType = record
          heuristicReport: heuristicReportType;
   end;

  tpDataReqType =  ...;

  tpDataIndType =  ...;

  uAseReqType = tpDataReqType;

  uAseIndType = tpDataIndType;

{ ********************************************************************** }

 cafPleaseType = record
       AETitle: aeTitleType;
       APInvocationId: apInvocationIdType;
       AEInvocationId: aeInvocationIdType;
    end;

 cafGiveType = record
    atomicActionId: atomicActionIdType;
    branchId: branchIdType;
  end;

 cafDetachType = record
    status: (free_channel, cleanup, not_used);
  end;

 cafRecoverIndType = cRecoverIndType;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
recReqType = (cafPleaseReq, cafDetachReq, cRecoverReq, cRecoverRsp,
   afRecoverReq, afHeurRepReq, afTokenGiveReq, afAbortReq);

recIndType = (cafGiveInd, cafRecoverInd, cRecoverInd,
   cRecoverCnf, afHeurRepInd, afAbortInd);

recoveryReqType = record
   chanIdx: 1..MAX_CHANS;
   atomicActionId: atomicActionIdType;
   branchId: branchIdType;
   case order: recReqType of
     cafPleaseReq: ( cafPleasePars: cafPleaseType; );
     cafDetachReq: ( cafDetachPars: cafDetachType; );
     cRecoverReq: ( cRecoverReqPars: cRecoverReqType; );
     cRecoverRsp: ( cRecoverRspPars: cRecoverRspType; );
     afRecoverReq: ( afRecoverReqPars: afRecoverReqType; );
     afHeurRepReq: ( afHeurRepReqPars: afHeuristicReqType; );
     afTokenGiveReq: ( afTokenGiveReqPars: afTokenGiveReqType; );
     afAbortReq: ( afAbortReqPars: afAbortReqType; )
 end;

recoveryIndType = record
   chanIdx: 1..MAX_CHANS;
   case order: recIndType of
     cafGiveInd: ( cafGivePars: cafGiveType; );
     cafRecoverInd: ( cafRecoverIndPars: cafRecoverIndType; );
     cRecoverInd: ( cRecoverIndPars: cRecoverIndType; );
     cRecoverCnf: ( cRecoverCnfPars: cRecoverCnfType; );
     afHeurRepInd: ( afHeurRepIndPars: afHeuristicIndType; );
     afAbortInd: ( afAbortIndPars: afAbortIndType; )
 end;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{ **** comfun.e ************************************************************ }
{ **** miscellaneous functions for data mapping, encoding etc. ************* }

{ ************************************************************************** }
{ *                                                                        * }
{ *   Mapping/encoding functions.                                          * }
{ *                                                                        * }
{ *  Mapping functions among TP and AF services and TP Pdus are not fully * }
{ *  specified in Estelle; the mapping can easily be derived from the main* }
{ *  text part of the standard.                                           * }
{ ************************************************************************** }

  procedure mapBeginDialReqAFtoTP(
                param: afBeginDialReqType;
                var sentParam: tpBeginDialRIType); primitive;

  procedure mapBeginDialReqTPtoAF(
                param: tpBeginDialogueReqType;
                var sentParam: afBeginDialReqType); primitive;

  procedure mapCAFPleaseToAF(
                param: cafPleaseType;
                var sentParam: afBeginDialReqType); primitive;

  procedure mapBeginDialIndAFtoTP(
                afBegDialInd: afBeginDialIndType;
                var dialPars: tpBeginDialogueIndType); primitive;

  procedure mapBeginDialCnfAFtoTP(
                afBegDialCnf: afBeginDialCnfType;
                var dialPars: tpBeginDialogueCnfType); primitive;

  procedure mapBeginDialRItoAF(
                param: tpBeginDialRIType;
                var sentParam: afBeginDialIndType); primitive;

  procedure mapBeginDialRspAFtoTP(
                param: afBeginDialRspType;
                var sentParam: tpBeginDialRCType); primitive;

  procedure mapBeginDialRCtoAF(
                param: tpBeginDialRCType;
                var sentParam: afBeginDialCnfType); primitive;

  procedure mapAbortReqAFtoTP(
                param: afAbortReqType;
                var sentParam: tpAbortRIType); primitive;

  procedure mapAbortRItoAF(param: tpAbortRIType;
                var sentParam: afAbortIndType); primitive;

  function embeddedTPRecoverRI(param: userDataType): boolean; primitive;
      { is a TP_RECOVER_RI pdu embedded in received user data? }

  function embeddedTPheurAndAbortRI(param:userDataType):boolean; primitive;
      { does the user data contain a TP_HEURISTIC_REPORT_RI pdu }
      { and a TP_ABORT_RI pdu ?                                 }

  function embeddedTPAbortRI( param: userDataType): boolean; primitive;
      { does the user data contain a TP_ABORT_RI pdu ?          }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
function embeddedTPHeurRI( param: userDataType): boolean; primitive;
    { does the user data contain a TP_HEURISTIC_REPORT_RI pdu ?}

function embeddedTPBeginDialRC( param: userDataType): boolean; primitive;
    { does the user data contain a TP_BEGIN_DIALOGUE_RC pdu ?  }

function embeddedTPpdu( param: userDataType): boolean; primitive;
    { does the user data contain any TP pdu ?                  }

function extractAbortRI( param: userDataType): tpAbortRIType; primitive;
    { returns the TP_ABORT_RI pdu embedded in the user data }

function extractHeurRI( param: userDataType): tpHeuristicRepRIType;
    { returns the TP_HEURISTIC_REPORT_RI pdu embedded in the user data }
  primitive;

function extractRecoverRI( param: userDataType): tpRecoverRIType;
    { returns the TP_RECOVER_RI pdu embedded in the user data }
  primitive;

function extractPrepareRI( param: userDataType): tpPrepareRIType;
    { returns the TP_PREPARE_RI pdu embedded in the user data }
  primitive;

function extractBeginDialRC( param: userDataType): tpBeginDialRCType;
    { returns the TP_BEGIN_DIALOGUE_RC pdu embedded in the user data }
  primitive;

function extractTPTokenGiveRI( param: userDataType): tpTokenGiveRIType;
    { returns the TP_TOKEN_GIVE_RI pdu embedded in the user data }
  primitive;

function regularTPTokenGiveRI( param: userDataType): boolean;
    { is a TP_TOKEN_GIVE_RI(regular) pdu embedded in the user data }
  primitive;

function twrTPTokenGiveRI( param: userDataType): boolean;
    { is a TP_TOKEN_GIVE_RI(two-way-rec) pdu embedded in the user data }
  primitive;

function keepTPTokenGiveRI( param: userDataType): boolean;
    { is a TP_TOKEN_GIVE_RI(keep) pdu embedded in the user data }
  primitive;

procedure encAbortRI( pdu: tpAbortRIType; var param: userDataType);
    { encodes TP_ABORT_RI into user data }
  primitive;

procedure encTokenGiveRI(pdu: tpTokenGiveRIType; var param: userDataType);
    { encodes TP_TOKEN_GIVE_RI PDU into user data }
  primitive;

procedure encHeurRI( pdu: tpHeuristicRepRIType; var param: userDataType);
    { encodes TP_HEURISTIC_REPORT_RI into user data }
  primitive;

procedure encRecRI( pdu: tpRecoverRIType; var param: userDataType);
    { encodes TP_RECOVER_RI into user data }
  primitive;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
procedure encBeginDialRC(pdu: tpBeginDialRCType; var param: userDataType);
    { encodes TP_BEGIN_DIALOGUE_RC into user data }
  primitive;

procedure encPrepRI( pdu: tpPrepareRIType; var param: userDataType);
    { encodes TP_PREPARE_RI into user data }
  primitive;

function concatenate( pdu1: tpAbortRIType; pdu2: tpHeuristicRepRIType):
  userDataType; primitive;
    { concatenates the two pdus - return resulting structure }

function newDialChannelCorrelator: correlatorType; primitive;
    { return unique dialogue or channel correlator          }

function match( dCorrel1, dCorrel2: correlatorType): boolean; primitive;
    { indicates whether two dialogue/channel correlators match }

function newAtomicActionId: atomicActionIdType; primitive;

function newBranchId( nzt: atomicActionIdType): branchIdType; primitive;

function sameBranchId( Dzb: branchIdType;
              param: recoveryIndType): boolean; primitive;

function twRecovery( pars: afBeginDialReqType): boolean; primitive;

function getTPPMIdx( nzt: atomicActionIdType; bzt: branchIdType): integer;
  primitive;

function notNullBranchId(bid: branchIdType): boolean; primitive;
 { is this a null branch id }

function hasToken(assocId: assocIdType): boolean; primitive;
   { checks whether minor sync. token is owned on this association. }

function allocateDialIdx: integer; primitive;
   { creates a new id for a dialogue }

function beginTransactionReq(pars:tpBeginDialogueReqType):boolean;primitive;
   { is chained funct.unit selected, or begin transaction par set ? }

function commitFuncUnitBDind( pars: afBeginDialIndType): boolean;primitive;
   { is commit functional unit selected }

function unchainedFuncUnitBDind( pars: afBeginDialIndType): boolean;
   primitive;
   { is unchained functional unit selected }

function HandshakeFuncUnitBDind( pars: afBeginDialIndType): boolean;
   primitive;
  { is handshake functional unit selected }

function sharedFuncUnitBDind( pars: afBeginDialIndType): boolean; primitive;
  { is shared functional unit selected }

function beginTransactionInd(pars:afBeginDialIndType): boolean;primitive;
  { is chained funct.unit selected, or begin transaction par set ? }

function unknownRecipient(pars:tpBeginDialogueReqType): boolean;primitive;
   { is recipient of begin dialogue req unknown ?}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  function commitFuncUnitBDreq( pars: tpBeginDialogueReqType):boolean;
      primitive;
     { is commit functional unit selected }

  function unchainedFuncUnitBDreq( pars: tpBeginDialogueReqType): boolean;
      primitive;
     { is unchained functional unit selected }

  function HandshakeFuncUnitBDreq( pars: tpBeginDialogueReqType): boolean;
      primitive;
     { is handshake functional unit selected }

  function sharedFuncUnitBDreq( pars: tpBeginDialogueReqType): boolean;
       primitive;
     { is shared functional unit selected }

{ ********************************************************************** }
{ *                                                                    * }
{ *  Local Decision Functions.                                         * }
{ *                                                                    * }
{ ********************************************************************** }

 function rejectDialReq( pars: tpBeginDialogueReqType): boolean; primitive;
    { local decision to reject dialogue request }

 function rejectDialInd( pars: afBeginDialIndType): boolean; primitive;
    { local decision to reject dialogue indication }

 function wantToWaitForOutcome: boolean; primitive;
    { want to reject recovery until outcome of transaction is known }

 function wantToBid: boolean; primitive;
    { called when bid is not mandatory to determine whether the local }
    { contention loser will bid before attempting to begin a dialogue }


 function acceptBid: boolean; primitive;
    { called when contention winner receives a bid indication; the     }
    { decision to accept or reject the bid is based on local conside- }
    { rations.                                                         }

 function rejectDelayedBid: boolean; primitive;
    { when a contention winner decides to accept a bid from the peer   }
    { and decides to wait until it gets the token to do so, it may     }
    { later on revise its decision based on the local context (e.g.,   }
    { timers ...). This function returns true in that case.            }

 function delayBidUntilToken: boolean; primitive;
    { local choice to delay bid response until token gets here }

 function associationReserved: boolean; primitive;
    { local choice to reserv an association }

 function wantToDelayDeferredEnd: boolean; primitive;
    { local choice to delay deferred end request until prepare is sent }

 function wantToDelayDeferredGC: boolean; primitive;
    { local choice to delay deferred grant_control req until prepare is sent }

 function wantToDetachFirst: boolean; primitive;
    { local choice to detach first channel in case of duplicate channels }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
function heuristicDecision: boolean; primitive;
   { local choice to make a heuristic decision }

function heuristicReport: heuristicReportType; primitive;
   { heuristic value after a local heuristic is made }

function heuristicDecisionComp: boolean; primitive;
   { decision to compensate for a heuristic decision }

procedure setDataRdyState; primitive;
   { setting data in ready state - sometime later boundReady is true }

procedure setDataFinalState; primitive;
   { sets data in final state - sometime later boundFinal is true }

procedure setDataInitialState; primitive;
   { sets data in initial state - sometime later boundInitial is true }

function rollbackByTPPM: boolean; primitive;
    { rollback trigerred by TPPM with no precise reason }

function terminateChannel: boolean; primitive;
    { free channel terminated by cpm  }

{ ************************************************************************ }
{ *                                                                      * }
{ * Primitives for access to the pool of associations.                   * }
{ *                                                                      * }
{ ************************************************************************ }

procedure allocateNewTPPMId( var id: integer); primitive;
    { allocates a new local identifier for a (tppm,tpsui) pair }

procedure deallocateTPPMId( id: integer); primitive;
    { releases the local identifier of a (tppm,tpsui) pair }

procedure initAssociationPool; primitive;
    { initializes the local association pool }

function getPoolAssocId( bdPars: afBeginDialReqType): assocIdType; primitive;
    { returns the identifier of a local, free association }

procedure releasePoolAssoc( assocId: assocIdType); primitive;
    { returns an association to the pool of free associations }

function pendingTPBeginChannelInd: boolean; primitive;
    { indicates that a begin indication for a channel is pending }

procedure getAssocIdForPendingTPChanInd(var assocId:assocIdType); primitive;
    { returns id of association on which begin channel ind was received }

procedure makeBeginDialIndPending( assocId: assocIdType;
                        pars: tpBeginDialRIType); primitive;
    { put back a TP_BEGIN_DIALOGUE_RI pdu on an association - this }
    { happens after a begin dialogue collision and will lead to    }
    { the creation of a new macf, tpsui pair                       }

function pendingTPBeginDialogueOrBidInd: boolean; primitive;
    { indicates that a begin dialogue or bid indication is pending }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
procedure getAssocIdForPendingTPDialOrBidInd( var assocId:assocIdType);
    { returns id of association on which pending indication was received }
   primitive;

function pendingTPBeginDialogueReq: boolean; primitive;
    { indicates that a begin dialogue request is pending }

{ ************************************************************************ }
{ *                                                                      * }
{ * System Logs for heuristic reporting.                                 * }
{ *                                                                      * }
{ * These logs survive node crashes and transaction completions.         * }
{ *                                                                      * }
{ ************************************************************************ }

procedure logDamage( aaid: atomicActionIdType; report: heuristicReportType);
      primitive;
   { creates a log damage record  for the node as defined by the atomic  }
   { action Id, if such a record does not already exist; update record's }
   { heuristic value with the value received (heuristicHazard or         }
   { heuristicMix).                                                      }

function existLogDamage( aaid: atomicActionIdType) : boolean; primitive;
   { indicate whether a log damage record exists for the transaction }

function recordedDamage( aaid: atomicActionIdType) : heuristicReportType;
      primitive;
   { return damage recorded for node specified by atomic action id  }

procedure deleteLogDamage(aaid: atomicActionIdType); primitive;
   { removes log damage record for specified node.                   }

procedure logHeuristic( aaid:atomicActionIdType; log: heuristicReportType);
     primitive;
   { creates a log heuristic record  for the node defined by the atomic  }
   { action Id, if such a record does not already exist.  Update record's}
   { value with the value received: heuristicFinal, heuristInitial, or   }
   { heuristicMix.                                                       }

function existLogHeuristic( aaid: atomicActionIdType) : boolean; primitive;
   { indicate whether a log heuristic record exists for the specified }
   { transaction                                                      }

function recordedHeuristic( aaid: atomicActionIdType) : heuristicReportType;
     primitive;
   { return heuristic recorded for node specified by atomic action id.}

procedure deleteLogHeuristic(aaid: atomicActionIdType); primitive;
   { removes heuristic log record for specified node.                }

{ ************************************************************************ }
{ *                                                                      * }
{ * System Logs for recovery.                                            * }
{ *                                                                      * }
{ *  These logs survive site crashes but not transaction completion.     * }
{ *                                                                      * }
{ ************************************************************************ }

procedure createTransNodeRecord( aaid:atomicActionIdType; bid:branchIdType;
     supHndlFlag: boolean; supHndl: recoveryContextHandleType); primitive;
   { creates a log record for the node as defined by the atomic action id }
   { and the id of the branch to the superior, if any; records exist for  }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  { the duration of the transaction and store all branch ids and their   }
  { related recovery context handle, if any.  In addition, the record    }
  { stores the state of the node for the current transaction: logInitial }
  { or logFinal.                                                         }

procedure addSubordinateBranchRecord( aaid: atomicActionIdType;
      subBid: branchIdType; subHndlFlag: boolean;
      subHndl: recoveryContextHandleType); primitive;
  { adds a subordinate branch to an existing node record. }

procedure recordLogCommit( aaid: atomicActionIdType); primitive;
  { updates the log state for the specified transaction to logCommit. }

procedure recordLogReady( aaid: atomicActionIdType); primitive;
  { updates the log state for the specified transaction to logReady. }

function recordedTransNodeState(aaid:atomicActionIdType): transactionState;
   primitive;
  { returns the state recorded for the specified transaction }

function transNodeRecordLogCount: integer; primitive;
  { returns the number of records in the node log, i.e., the number of }
  { current transactions.                                              }

function logSubBranchCount(Nzt: atomicActionIdType): integer; primitive;
  { returns the number of subordinate branches for atomic action Nzt }

function logSubBranchIdAtIdx(Nzt:atomicActionIdType;
       idx:integer):branchIdType; primitive;
  { returns the id of the subordinate branch at index idx }

function transNodeRecordAtIdx(idx:integer):atomicActionIdType; primitive;
  { returns the node record at position "idx" in the node log.         }

function branchToSupInTransNodeLog(aaid:atomicActionIdType): branchIdType;
     primitive;
  { returns the branch id to the superior in the specified record }

procedure recoveryContextHndlInTransLog( aaid: atomicActionIdType;
    var hndlPresent: boolean; var hndl: recoveryContextHandleType);
    primitive;
  { returns the recovery context handle stored for the specified branch. }

procedure deleteSubordinateBranchRecord( aaid: atomicActionIdType;
    bid: branchIdType); primitive;
  { deletes from the transaction record the information related to the }
  { specified subordinate branch.                                      }

procedure deleteTransNodeRecord( aaid: atomicActionIdType); primitive;
  { deletes the transaction record from the log. }

function existNodeRecord( aaid: atomicActionIdType): boolean; primitive;
  { determines whether a node record exists for the specified transaction }
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*** Estelle channels for interaction among modules *********************}
{************************************************************************}
{                                                                        }
{   TP Service Interface.                                                }
{                                                                        }
{Note: all TP service requests/indications/responses/confirms that are   }
{      dialogue specific have a parameter that identifies the dialogue   }
{      to which they are related.                                        }
{                                                                        }
{************************************************************************}

 channel TP_Service( user, provider);
   by user:
      TP_BEGIN_DIALOGUE_req( param: tpBeginDialogueReqType);
      TP_BEGIN_DIALOGUE_rsp( param: tpBeginDialogueRspType);
      TP_END_DIALOGUE_req( param: tpEndDialogueReqType);
      TP_END_DIALOGUE_rsp( param: tpEndDialogueRspType);
      TP_U_ERROR_req( param: tpUErrorReqType);
      TP_U_ABORT_req( param: tpUAbortReqType);
      TP_GRANT_CONTROL_req( param: tpGrantControlReqType);
      TP_REQUEST_CONTROL_req( param: tpRequestControlReqType);
      TP_HANDSHAKE_req( param: tpHandshakeReqType);
      TP_HANDSHAKE_GC_req( param: tpHandshakeGCReqType);
      TP_HANDSHAKE_rsp( param: tpHandshakeRspType);
      TP_HANDSHAKE_GC_rsp( param: tpHandshakeGCRspType);
      TP_BEGIN_TRANSACTION_req( param: tpBeginTransactionReqType);
      TP_DEFERRED_END_DIALOGUE_req( param: tpDeferredEndDialogueReqType);
      TP_DEFERRED_GC_req( param: tpDeferredGCReqType);
      TP_PREPARE_req( param: tpPrepareReqType);
      TP_COMMIT_req( param: tpCommitReqType);
      TP_DONE_req( param: tpDoneReqType);
      TP_ROLLBACK_req( param: tpRollbackReqType);
      TP_DATA_req( param: tpDataReqType);

   by provider:

      TP_BEGIN_DIALOGUE_ind( param: tpBeginDialogueIndType);
      TP_BEGIN_DIALOGUE_cnf( param: tpBeginDialogueCnfType);
      TP_END_DIALOGUE_ind( param: tpEndDialogueIndType);
      TP_END_DIALOGUE_cnf( param: tpEndDialogueCnfType);
      TP_U_ERROR_ind( param: tpUErrorIndType);
      TP_U_ABORT_ind( param: tpUAbortIndType);
      TP_GRANT_CONTROL_ind( param: tpGrantControlIndType);
      TP_REQUEST_CONTROL_ind( param: tpRequestControlIndType);
      TP_HANDSHAKE_ind( param: tpHandshakeReqType);
      TP_HANDSHAKE_GC_ind( param: tpHandshakeGCIndType);
      TP_HANDSHAKE_cnf( param: tpHandshakeCnfType);
      TP_HANDSHAKE_GC_cnf( param: tpHandshakeGCCnfType);
      TP_BEGIN_TRANSACTION_ind( param: tpBeginTransactionIndType);
      TP_DEFERRED_END_DIALOGUE_ind( param: tpDeferredEndDialogueIndType);
      TP_DEFERRED_GC_ind( param: tpDeferredGCIndType);
      TP_PREPARE_ind( param: tpPrepareIndType);
      TP_READY_ind( param: tpReadyIndType);
      TP_COMMIT_ind( param: tpCommitIndType);
      TP_ROLLBACK_ind( param: tpRollbackIndType);
      TP_DATA_ind( param: tpDataIndType);
      TP_P_ABORT_ind( param: tpPAbortIndType);
      TP_COMMIT_COMPLETE_ind( param: tpCommitCompleteIndType);
      TP_ROLLBACK_COMPLETE_ind( param: tpRollbackCompleteIndType);
      TP_HEURISTIC_REPORT_ind( param: tpHeuristicReportIndType);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
channel TP_Service_aug( user, provider);
  by user:
     TP_BEGIN_DIALOGUE_req( param: tpBeginDialogueReqType; dial: integer);
     TP_BEGIN_DIALOGUE_rsp( param: tpBeginDialogueRspType; dial: integer);
     TP_END_DIALOGUE_req( param: tpEndDialogueReqType; dial: integer);
     TP_END_DIALOGUE_rsp( param: tpEndDialogueRspType; dial: integer);
     TP_U_ERROR_req( param: tpUErrorReqType; dial: integer);
     TP_U_ABORT_req( param: tpUAbortReqType; dial: integer);
     TP_GRANT_CONTROL_req( param: tpGrantControlReqType; dial: integer);
     TP_REQUEST_CONTROL_req( param: tpRequestControlReqType; dial: integer);
     TP_HANDSHAKE_req( param: tpHandshakeReqType; dial: integer);
     TP_HANDSHAKE_GC_req( param: tpHandshakeGCReqType; dial: integer);
     TP_HANDSHAKE_rsp( param: tpHandshakeRspType; dial: integer);
     TP_HANDSHAKE_GC_rsp( param: tpHandshakeGCRspType; dial: integer);
     TP_BEGIN_TRANSACTION_req( param: tpBeginTransactionReqType;
                dial: integer);
     TP_DEFERRED_END_DIALOGUE_req( param: tpDeferredEndDialogueReqType;
                dial: integer);
     TP_DEFERRED_GC_req( param: tpDeferredGCReqType; dial: integer);
     TP_PREPARE_req( param: tpPrepareReqType; dial: integer);
     TP_COMMIT_req( param: tpCommitReqType);
     TP_DONE_req( param: tpDoneReqType);
     TP_ROLLBACK_req( param: tpRollbackReqType);
     TP_DATA_req( param: tpDataReqType; dial: integer);

  by provider:

     TP_BEGIN_DIALOGUE_ind( param: tpBeginDialogueIndType; dial: integer);
     TP_BEGIN_DIALOGUE_cnf( param: tpBeginDialogueCnfType; dial: integer);
     TP_END_DIALOGUE_ind( param: tpEndDialogueIndType; dial: integer);
     TP_END_DIALOGUE_cnf( param: tpEndDialogueCnfType; dial: integer);
     TP_U_ERROR_ind( param: tpUErrorIndType; dial: integer);
     TP_U_ABORT_ind( param: tpUAbortIndType; dial: integer);
     TP_GRANT_CONTROL_ind( param: tpGrantControlIndType; dial: integer);
     TP_REQUEST_CONTROL_ind( param: tpRequestControlIndType; dial:integer);
     TP_HANDSHAKE_ind( param: tpHandshakeReqType; dial: integer);
     TP_HANDSHAKE_GC_ind( param: tpHandshakeGCIndType; dial: integer);
     TP_HANDSHAKE_cnf( param: tpHandshakeCnfType; dial: integer);
     TP_HANDSHAKE_GC_cnf( param: tpHandshakeGCCnfType; dial: integer);
     TP_BEGIN_TRANSACTION_ind( param: tpBeginTransactionIndType;
            dial: integer);
     TP_DEFERRED_END_DIALOGUE_ind( param: tpDeferredEndDialogueIndType;
            dial: integer);
     TP_DEFERRED_GC_ind( param: tpDeferredGCIndType; dial: integer);
     TP_PREPARE_ind( param: tpPrepareIndType; dial: integer);
     TP_READY_ind( param: tpReadyIndType; dial: integer);
     TP_COMMIT_ind( param: tpCommitIndType);
     TP_ROLLBACK_ind( param: tpRollbackIndType);
     TP_DATA_ind( param: tpDataIndType; dial: integer);
     TP_P_ABORT_ind( param: tpPAbortIndType; dial: integer);
     TP_COMMIT_COMPLETE_ind( param: tpCommitCompleteIndType);
     TP_ROLLBACK_COMPLETE_ind( param: tpRollbackCompleteIndType);
     TP_HEURISTIC_REPORT_ind( param: tpHeuristicReportIndType;
            dial: integer);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
{*************************************************************************}
{                                                                         }
{   SACF - (CCR, Pres. ASCE) interface.                                   }
{                                                                         }
{note: TP_APDUs mapped into P_DATA service primitives are presented in    }
{      the spec as non-embedded;                                          }
{                                                                         }
{*************************************************************************}

channel SACF_lower( user, provider);

  by user:
     A_RELEASE_req( param: aReleaseReqType);
     A_RELEASE_rsp( param: aReleaseRspType);
     A_ABORT_req( param: aAbortReqType);
     C_BEGIN_req( param: cBeginReqType);
     C_BEGIN_rsp( param: cBeginRspType);
     C_PREPARE_req( param: cPrepareReqType);
     C_READY_req( param: cReadyReqType);
     C_COMMIT_req( param: cCommitReqType);
     C_COMMIT_C_BEGIN_req( param: cCommitBeginReqType);
     C_COMMIT_rsp( param: cCommitRspType);
     C_ROLLBACK_req( param: cRollbackReqType);
     C_ROLLBACK_rsp( param: cRollBackRspType);
     C_RECOVER_req( param: cRecoverReqType);
     C_RECOVER_rsp( param: cRecoverRspType);
     P_TOKEN_PLEASE_req( param: pTokenPleaseReqType);
     P_TOKEN_GIVE_req( param: pTokenGiveReqType);
     U_ASE_req( param: uAseReqType);

  by provider:
     A_RELEASE_ind( param: aReleaseIndType);
     A_RELEASE_cnf( param: aReleaseCnfType);
     A_ABORT_ind( param: aAbortIndType);
     A_P_ABORT_ind( param: aPAbortIndType);
     C_BEGIN_ind( param: cBeginIndType);
     C_BEGIN_cnf( param: cBeginCnfType);
     C_PREPARE_ind( param: cPrepareIndType);
     C_READY_ind( param: cReadyIndType);
     C_COMMIT_ind( param: cCommitReqType);
     C_COMMIT_C_BEGIN_ind( param: cCommitBeginReqType);
     C_COMMIT_cnf( param: cCommitCnfType);
     C_ROLLBACK_ind( param: cRollbackIndType);
     C_ROLLBACK_cnf( param: cRollbackCnfType);
     C_RECOVER_ind( param: cRecoverIndType);
     C_RECOVER_cnf( param: cRecoverCnfType);
     P_TOKEN_PLEASE_ind( param: pTokenPleaseIndType);
     P_TOKEN_GIVE_ind( param: pTokenGiveIndType);
     U_ASE_ind( param: uAseIndType);


  by user, provider:

     {the following interactions represent TP_APDUs when they are  }
     {embedded in the user data field of P_DATA service primitives;}
     {when they are embedded in CCR or ASCE primitives, the text    }
     {shows that explicitly.                                        }
     TP_BEGIN_DIALOGUE_RI( param: tpBeginDialRIType);
     TP_BEGIN_DIALOGUE_RC( param: tpBeginDialRCType);
     TP_BID_RI( param: tpBidRIType);
     TP_BID_RC( param: tpBidRCType);
     TP_END_DIALOGUE_RI( param: tpEndDialRIType);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      TP_END_DIALOGUE_RC( param: tpEndDialRCType);
      TP_U_ERROR_RI( param: tpUErrorRIType);
      TP_U_ERROR_RC( param: tpUErrorRCType);
      TP_ABORT_RI( param: tpAbortRIType);
      TP_GRANT_CONTROL_RI( param: tpGrantControlRIType);
      TP_REQUEST_CONTROL_RI( param: tpRequestControlRIType);
      TP_HANDSHAKE_RI( param: tpHandshakeRIType);
      TP_HANDSHAKE_RC( param: tpHandshakeRCType);
      TP_HANDSHAKE_GC_RI( param: tpHandshakeGCRIType);
      TP_HANDSHAKE_GC_RC( param: tpHandshakeGCRCType);
      TP_DEFER_RI( param: tpDeferRIType);

{========== end SACF lower interface ==================================}

channel SAF_Service( user, provider);

  by user:
     SAF_DETACH_ASSOCIATION_req( param: safDetachAssocReqType);

  by provider:
     SAF_ASSOCIATION_LOST_ind( param: safAssocLostIndType);


channel AF_Service( user, provider);
  {interface between SACF and Dialogue/macf; includes CCR and presentation}
  {service primitives; excludes SAF_detach primitives (see above)          }

  by user:

     AF_BEGIN_DIALOGUE_req( param: afBeginDialReqType);
     AF_BEGIN_DIALOGUE_rsp( param: afBeginDialRspType);
     AF_END_DIALOGUE_req( param: afEndDialReqType);
     AF_END_DIALOGUE_rsp( param: afEndDialRspType);
     AF_U_ERROR_req( param: afUErrorReqType);
     AF_U_ERROR_rsp( param: afUErrorRspType);
     AF_ABORT_req( param: afAbortReqType);
     AF_GRANT_CONTROL_req( param: afGrantControlReqType);
     AF_REQUEST_CONTROL_req( param: afRequestControlReqType);
     AF_HANDSHAKE_req( param: afHandshakeReqType);
     AF_HANDSHAKE_rsp( param: afHandshakeRspType);
     AF_HANDSHAKE_GC_req( param: afHandshakeGCReqType);
     AF_HANDSHAKE_GC_rsp( param: afHandshakeGCRspType);
     AF_DEFER_req( param: afDeferReqType);
     AF_PREPARE_req( param: afPrepareReqType);
     AF_HEURISTIC_req( param: afHeuristicReqType);
     AF_ABORT_HEURISTIC_req( param: afAbortAndHeuristicReqType);
     AF_RECOVER_req( param: afRecoverReqType);
     AF_TOKEN_GIVE_req( param: afTokenGiveReqType);
     AF_TOKEN_PLEASE_req( param: afTokenPleaseReqType);
     C_BEGIN_req( param: cBeginReqType);
     C_BEGIN_rsp( param: cBeginRspType);
     C_READY_req( param: cReadyReqType);
     C_COMMIT_req( param: cCommitReqType);
     C_COMMIT_C_BEGIN_req( param: cCommitBeginReqType);
     C_COMMIT_rsp( param: cCommitRspType);
     C_ROLLBACK_req( param: cRollbackReqType);
     C_ROLLBACK_rsp( param: cRollBackRspType);
     C_RECOVER_req( param: cRecoverReqType);
     C_RECOVER_rsp( param: cRecoverRspType);
     P_TOKEN_PLEASE_req( param: pTokenPleaseReqType);
     P_TOKEN_GIVE_req( param: pTokenGiveReqType);
     U_ASE_req( param: uAseReqType);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
  by provider:
     AF_BEGIN_DIALOGUE_ind( param: afBeginDialIndType);
     AF_BEGIN_DIALOGUE_cnf( param: afBeginDialCnfType);
     AF_END_DIALOGUE_ind( param: afEndDialIndType);
     AF_END_DIALOGUE_cnf( param: afEndDialCnfType);
     AF_U_ERROR_ind( param: afUErrorIndType);
     AF_U_ERROR_cnf( param: afUErrorCnfType);
     AF_ABORT_ind( param: afAbortIndType);
     AF_GRANT_CONTROL_ind( param: afGrantControlIndType);
     AF_REQUEST_CONTROL_ind( param: afRequestControlIndType);
     AF_HANDSHAKE_ind( param: afHandshakeIndType);
     AF_HANDSHAKE_cnf( param: afHandshakeCnfType);
     AF_HANDSHAKE_GC_ind( param: afHandshakeGCIndType);
     AF_HANDSHAKE_GC_cnf( param: afHandshakeGCCnfType);
     AF_DEFER_ind( param: afDeferIndType);
     AF_PREPARE_ind( param: afPrepareIndType);
     AF_HEURISTIC_ind( param: afHeuristicIndType);
     AF_ABORT_HEURISTIC_ind( param: afAbortAndHeuristicIndType);
     AF_RECOVER_ind( param: afRecoverIndType);
     AF_TOKEN_GIVE_ind( param: afTokenGiveIndType);
     AF_TOKEN_PLEASE_ind( param: afTokenPleaseIndType);
     C_BEGIN_ind( param: cBeginIndType);
     C_BEGIN_cnf( param: cBeginCnfType);
     C_READY_ind( param: cReadyIndType);
     C_COMMIT_ind( param: cCommitReqType);
     C_COMMIT_cnf( param: cCommitCnfType);
     C_COMMIT_C_BEGIN_ind( param: cCommitBeginReqType);
     C_ROLLBACK_ind( param: cRollbackIndType);
     C_ROLLBACK_cnf( param: cRollbackCnfType);
     C_RECOVER_cnf( param: cRecoverCnfType);
     C_RECOVER_ind( param: cRecoverIndType);
     P_TOKEN_PLEASE_ind( param: pTokenPleaseIndType);
     P_TOKEN_GIVE_ind( param: pTokenGiveIndType);
     U_ASE_ind( param: uAseIndType);

{======= end SACF upper interface ======================================}


{*************************************************************************}
{                                                                         }
{   CPM/TPPM Interface.                                                   }
{                                                                         }
{All AF/CAF service primitives exchanged between the TPPM and the CPM     }
{(including those leading to a recovery order on a channel) are embedded  }
{in two interactions: CPM_TO_TPPM_ind and TPPM_TO_CPM_req.                }
{                                                                         }
{*************************************************************************}

 channel CPM_TPPM_Service( cpm, tppm);

   by cpm:
      CPM_TO_TPPM_ind( param: recoveryIndType);

   by tppm:
      TPPM_TO_CPM_req( param: recoveryReqType);

 channel CPM_Service( user, provider);

   by user:

      CAF_PLEASE_req( param: cafPleaseType);
      CAF_DETACH_req( param: cafDetachType);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
      C_RECOVER_req( param: cRecoverReqType);
      C_RECOVER_rsp( param: cRecoverRspType);
      AF_RECOVER_req( param: afRecoverReqType);
      AF_HEURISTIC_req( param: afHeuristicReqType);
      AF_TOKEN_GIVE_req( param: afTokenGiveReqType);
      AF_ABORT_req( param: afAbortReqType);

   by provider:

      CAF_GIVE_ind( param: cafGiveType);
      CAF_DETACH_ind( param: cafDetachType);
      CAF_RECOVER_ind( param: cafRecoverIndType);
      C_RECOVER_ind( param: cRecoverIndType);
      C_RECOVER_cnf( param: cRecoverCnfType);
      AF_HEURISTIC_ind( param: afHeuristicIndType);
      AF_ABORT_ind( param: afAbortIndType);


{ ********************************************************************** }
{                                                                        }
{      The channel below includes synchronization messages sent by the   }
{      macf to individual transaction branches.                          }
{                                                                        }
{ ********************************************************************** }
  channel synchroMess( macf, dial);

   by macf:
      rPend; { rollback of next transaction is pending }
      sendReady; {order from macf to send ready to superior}
      sendCommitOrder; {order from macf to send commit order}
      rollbackAll; {order from macf to go to decided rollback state}
      commitComplete; {completion of commitment in the node}
      reportRollback; {report rollback to superior}
      rollbackComplete; {complete rollback in node}
      restartInReady; {restart in ready state after node crash}
      restartInCommit; {restart in commit state after node crash}
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Annex H
(informative)

LOTOS description of the TP protocol

H.1 Introduction

This annex presents a LOTOS description of the TP Protocol, which consists of the possible protocol/TP Service Primitive sequencing relating to one particular TPSUI. Therefore, the actions of one particular TPPM are specified, together with related actions of the CPM. The possible presence of more than one TPPM within the AEI is only modelled in so far as these other TPPMs may affect the behaviour of the TPPM of interest - for example, associations previously used may be unavailable due to use by another TPPM. The architecture of this specification reflects as far as possible the architecture given in the Protocol Document. Figure H.1 presents the organization of the top level of the LOTOS specification, showing the processes and their gates. Figure H.2 shows the details of the organization within the SAO.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

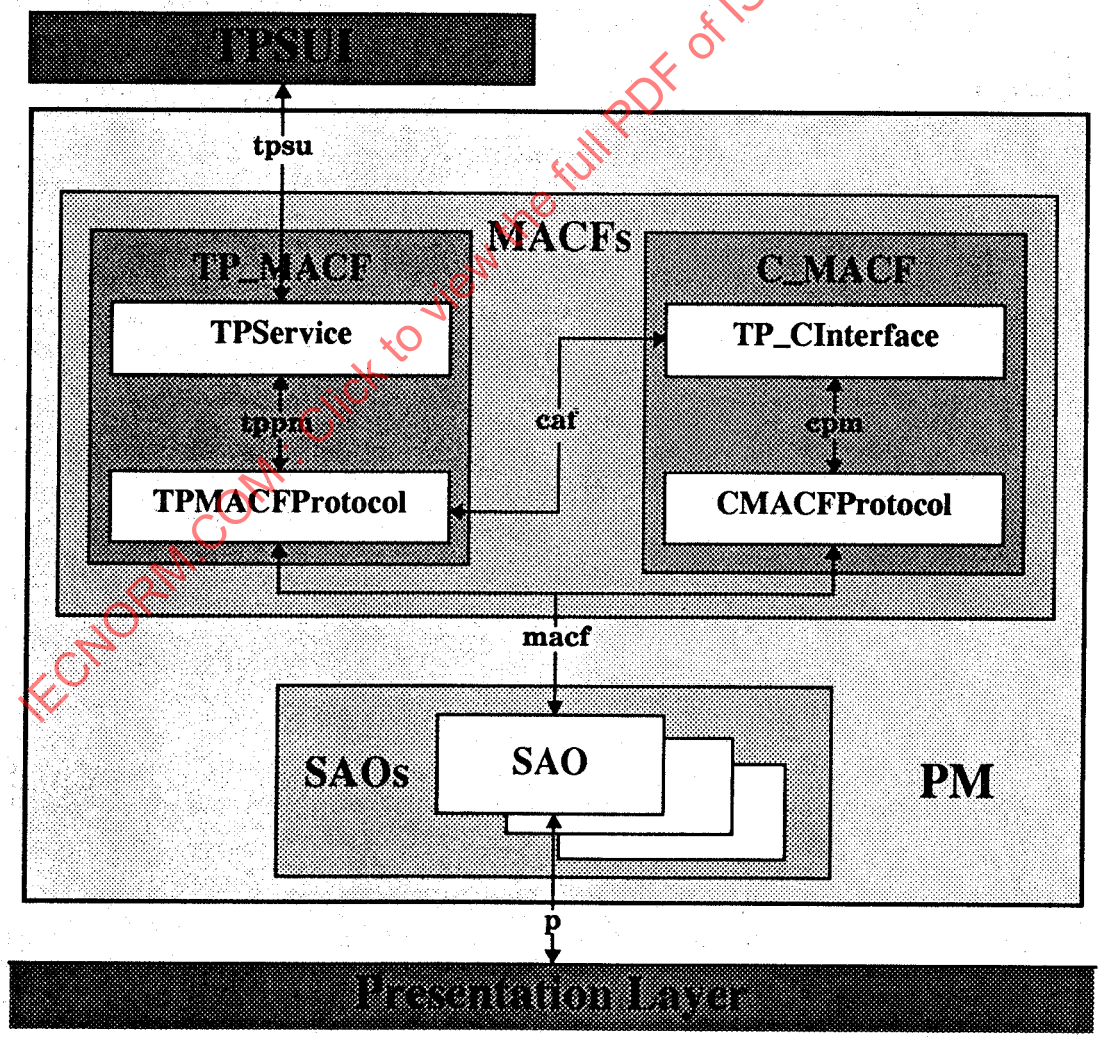
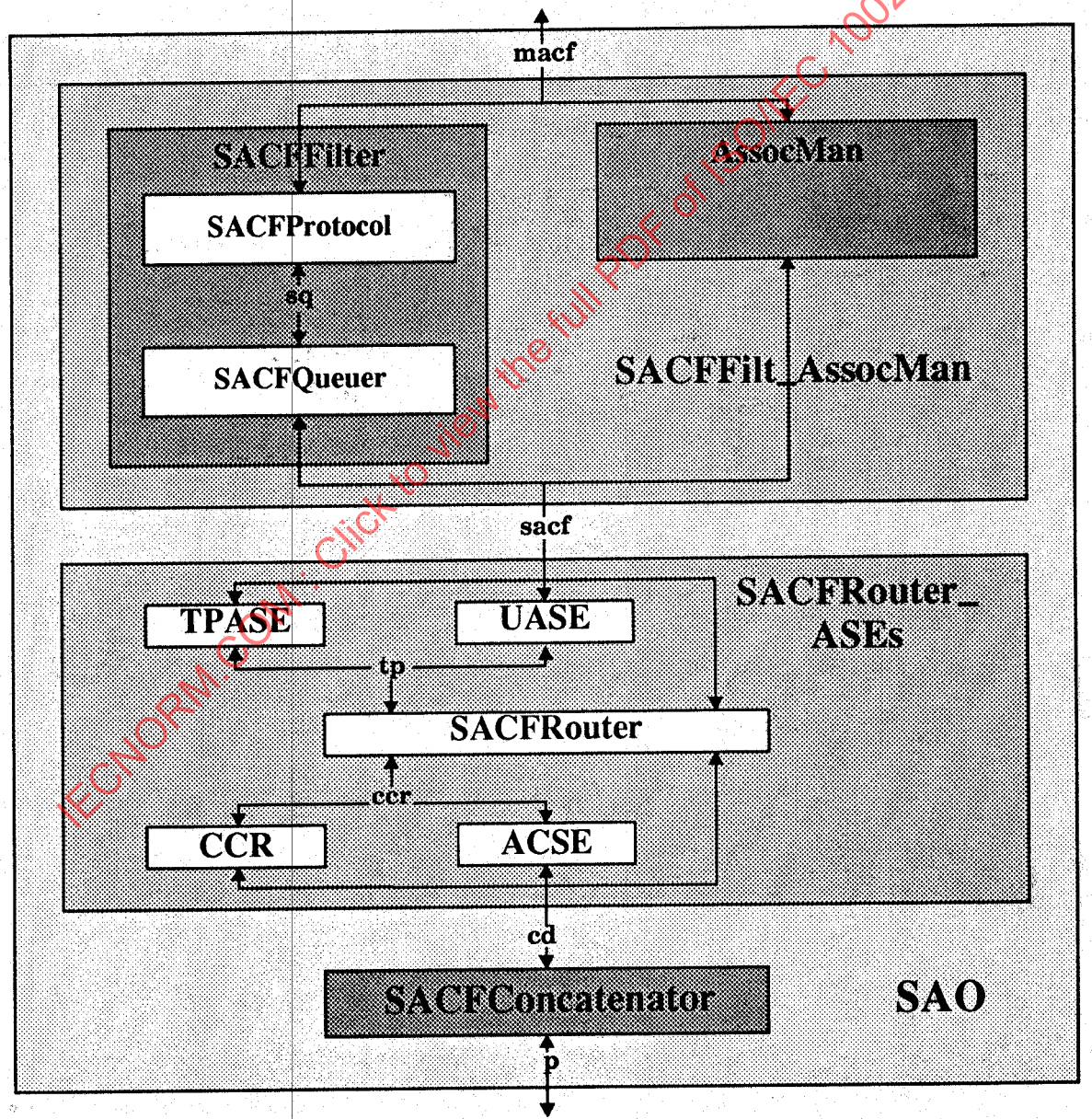


Figure H.1 - The Top Level Architecture

The refinements to the architecture in the Protocol Document are as follows:-

- In figure H.1 there are no TPPM or CPM processes as such - this is to allow the dynamic allocation of SAOs to a particular MACF, and hence SAO transfer etc. The TPPM is represented by the TP MACF, together with those SAO processes that are attached to it; similarly for the CPM.
- The TP MACF contains a TPService process. This process models the sequencing constraints imposed on TP Service Primitives by the Service Document - its inclusion is necessary to detect when Service Primitives violate these constraints, and therefore constitute either a protocol error or an internal error.
- The C MACF contains an explicit buffering process, necessary to prevent deadlock occurring between the TPPM and CPM.
- In figure H.2 there is no single SACF process, but rather three separate processes. Each represents a distinct and independent part of the SACF functionality.
- There is an Association Manager Process within each SAO, which models the constraints the association management procedures place on the behaviour of an SAO.



IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Figure H.2 - Details of the SAO Architecture

The TPMACFProtocol process iteratively calls a TPProcedures process. The TPProcedures process is a choice between a number of smaller sub-processes, each of which corresponds to a subclause within the (TPPM) MACF Procedures clause of the Protocol document. The operation of the Protocol process is then to map an input offered on one of its gates to the corresponding output(s), by means of the Procedures sub-process corresponding to the appropriate clause in the Protocol document. The operation of the Protocol processes in the C_MACF and SACFFilter is exactly the same. The TPASE is similar, but here the TPASE process itself iteratively calls the TPASEProcedures process.

The remaining processes shown in figures H.1 and H.2 have functionality as suggested by the naming, and are all covered in more detail in the appropriate subclauses. Table H.1 summarises which gates are used to communicate between which processes, and details which service primitives or PDUs are exchanged.

Table H.1 - Summary of Gate Usage

Gate	Communicates Between		Internal to	Information Passed (PToken sps are PTokenGive Req/Ind, and PTokenPlease Req/Ind)
tpsu	TPSUI	PM	-	TP sps
caf	TP_MACF	C_MACF	MACFs	CAF sps
cpm	TP_CInterface	CMACFProtocol	C_MACF	CAF sps
tppm	TP Service	TPMACFProtocol	TP_MACF	TP sps
macf	MACFs	SAOs	PM	AF, SAF, CCR sps, PTokenGive Req
sq	SACFProtocol	SACFQueuer	SACFFilter	AF, CCR sps, ARelease Ind/Rsp, PTokenGive Req
sacf	SACFFilter_ AssocMan	SACFRouter_ ASEs	SAO	AF, CCR, ACSE sps, PToken sps
tp	TPASE UASE	SACFRouter	SACFRouter_ ASEs	TP, UASE PDUs; PToken sps to UASE; AAbortReq/Ind or CCR sps with embedded APDUs.
ccr	SACFRouter	CCR ACSE	SACFRouter_ ASEs	ACSE sps, CCR sps
cd	ASEs	SACFConcatenator	SAO	TP, CCR, ACSE APDUs; PToken sps
p	PM	Presentation Layer	-	P sps

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

There is another feature of the architecture of this specification, not shown in figures H.1 and H.2. As outlined above, each of the processes corresponds to a component of the PM described in the Protocol Document. Each of the gates shown is used for transferring data between these processes. However, the specification also contains a number of other processes and gates which are used to coordinate the actions of the processes shown in figures H.1 and H.2. The coordination processes are responsible principally for maintaining the action sequence rules. The coordination processes communicate with each other and the PM components on the coordination gates aei. This gate is also used by the PM to communicate data other than service primitives or PDUs - for example, the TP MACF uses an event on the aei gate to tell the SACFConcatenator to close the psap. Coordination is explained in section H.1.4 and the addressing convention used is explained in section H.1.5. Both these subclauses may be omitted by those not wishing to understand the precise details of the operation of the specification. In particular section H.1.4 requires a high level of understanding of LOTOS.

H.1.1 Top Level Specification

```
specification OSITP[tpsu,p]: noexit behaviour
    AEI[tpsu,p]
```

where

The Application Layer process represents the constraints placed on the sequencing by the PM as already described, together with the second constraint,conformance_requirements, which ensures that the static conformance requirements are met.

```
process AEI[tpsu,p]: noexit :=
    PM[tpsu,p] |[tpsu,p]| conformance_requirements[tpsu,p]
endproc
```

There is a set, possibly empty, of SAOs associated with the MACFs of the PM at any point in time. The following process allows the MACFs and SAOs to synchronize on the macf gate with co-ordination provided on the aei gate.

```
process PM[tpsu,p] :noexit :=
    hide macf,aei in
    MACFs[tpsu,macf,aei] |[macf,aei]| SAOs[macf,p,aei]
endproc
```

H.1.2 MACFs

```
process MACFs[tpsu,macf,aei]:noexit:=
    hide caf in
    TP_MACF[tpsu,caf,macf,aei] |[caf]| C_MACF[caf,macf,aei]
endproc
```

The TP_MACF contains a MACFProtocol process and a TPService process. As the TPPM is never forced to exit, the TP_MACF process is "noexit". (The TPPM invocation after a node crash is represented by the same TP_MACF process). Similarly the C_MACF is "noexit". Consequently, all their component processes are also "noexit".

The variables for the TPMACFProtocol process are state vectors. The first is of sort set_of_assocs, and contains a possibly empty list of terms from sort assoc_entry. Each term of assoc_entry contains all the information relating to a particular dialogue or channel owned by this TPPM. The initial value is therefore just empty_assocs. The second argument is of sort tppm_info, and contains all the information the TPPM needs that doesn't relate to a particular dialogue (such as the transaction state). It is set to an initial value specified in the datatyping. The internal gate tlk is used for coordination purposes and as such is part of the action sequence mechanism.

```
process TP_MACF[tpsu,caf,macf,aei]:noexit:=
    hide tppm,tlk in
    TPService[tpsu,tppm,tlk]
    |[tpsu,tppm,tlk]|
    TPMACFProtocol[tpsu,tppm,caf,macf,aei,tlk](empty_assocs, empty_tppm_info)
endproc
```

The TPMACFMap process synchronizes with the coordinating process, TPPMCoord, on the macf, caf and tlk gates. It is the TPPMCoord process which ensures that the action sequence rules for the TPPM are obeyed. The variable for the TPPMCoord process is the set of dialogue identifiers associated with this TPPM (see section H.1.5); its initial value is therefore empty-headed. The TPPMAllow process is used to permit the occurrence of events that the TPPMCoord process is not interested in. The normal operation of the TPMACFProtocol process can be interrupted at any time by a node crash. When the TPPM recovers from the node crash information from the log-record is passed to the TPPMAfterNodecrash process on the caf gate, whose actions correspond to those of the protocol procedure of the same name. The TPMACFProtocol process is subsequently restarted (provided there is a log record).

```
process TPMACFProtocol[tpsu,tppm,caf,macf,aei,tlk] (assocs: set_of_assocs, info: tppm_info):noexit:=
    TPMACFMap[tppm,caf,macf,aei,tlk](assocs,info)
    |[macf,caf,tlk]|
    (   TPPMCoord[tpsu,caf,macf,aei,tlk](empty_dids) ||| TPPMAllow[tpsu,caf,macf])
    [> NodeCrash[tpsu,tppm,caf,macf,aei,tlk]
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
process NodeCrash[tpsu,tppm,caf,macf,aei,tlk]:noexit:=
    i; caf !tppm_crash;
    (   caf !read_log_record ?log: log_record;
        (   TPPMAfterNodeCrash[tppm,caf](log)
        >> accept Assocs:set_of_assocs, info:tppm_info in
            TPMACFProtocol[tpsu,tppm,caf,macf,aei,tlk](Assocs,info)
        )
        []
        caf !log_record_absent; stop
    )
endproc
```

As described in section H.3, the Protocol processes are iterative calls to the appropriate Procedures process. For the TPPM this iterative call is made from within the TPMACFMap sub-process. Having performed the actions of a particular clause, the procedures process will exit, allowing the Protocol process to be recalled with the updated state vector argument(s). The other action of a Protocol process is to allow an unlock event to happen to signal the end of an action sequence (see section H.3.2).

```
process TPMACFMap[tppm,caf,macf,aei,tlk] (Assocs:set_of_assocs,tppm:tppm_info) :noexit :=
    TPPMProcedures[tppm,caf,macf,aei,tlk](Assocs,tppm)
    []
    tlk!get_dids(Assocs); exit(Assocs,tppm)
>> accept Assocs:set_of_assocs,tppm:tppm_info in
    TPMACFMap[tppm,caf,macf,aei,tlk](Assocs,tppm)
endproc
```

The C_MACF contains a Protocol process which communicates with the TP_MACF via a buffering process, TP_CInterface, which needs to be made explicit in the LOTOS to prevent deadlock. All (normal) interaction between the TPPM and CPM is via the TP_CInterface, the only exception being a TPPM node crash. The interface, as part of the CPM, is used to store the TPPMs log record during normal processing. This value is then passed back on the caf when reinvoking the TPPM after node crash The CPMCoord process ensures that the CPM action sequence rules are obeyed and the CPMAllow acts as the TPPMAllow process above.

```
process C_MACF[caf,macf,aei]:noexit:=
    hide cpm,clk in
    (   TP_CInterface[caf, cpm]
        |[cpm]|
        CMACFProtocol[cpm,macf,aei,clk](empty_Assocs)
    )
    |[clk,cpm]|
    (   CPMCoord[cpm,macf,aei,cpm](empty_dids) ||| CPMAllow[cpm,macf] )
endproc
```

The CMACFProtocol process acts in a similar way to TPMACFMap.

```
process CMACFProtocol[cpm,macf,aei,clk]
(Assocs:set_of_assocs) :noexit :=
    CPMProcedures[cpm,macf,aei,clk]( Assocs)
    []
    clk !get_dids(Assocs); exit( Assocs)
>> accept Assocs:set_of_assocs in  CMACFProtocol[cpm,macf,aei,clk](Assocs)
endproc
```

H.1.3 SAO

The SAOs process consists of a number of SAO processes acting in parallel, where the number is chosen as a local matter. The SAO process may exit.

```
process SAOs[macf,p,aei]:noexit:=
    (SAO[macf,p,aei] >> stop) ||| i;SAOs[macf,p,aei]
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

Each SAO consists of the SAO components as shown in figure 2, represented by the SAOComps process, with a coordinating process, SAOCoord, which maintains the action sequence rules, and also controls when the SAO exits. The parameter to the coordinator represents the current dialogue identifier assigned to the SAO, and hence is initially set to did_absent.

```
process SAO[macf,p,aei]: exit:=
    hide slk,sao in
    SAOComps[macf,p,aei,sao,slk] |[macf,aei,sao,slk,p]|
    SAOCoord[macf,p,aei,sao,slk](did_absent)
endproc
```

The SAOComps process has components which specify the procedures for association establishment, the sacf filtering procedures, the TP-ASE procedures and the concatenation rules along with the routing functionality of the SACF and the interaction of TP with other ASEs.

```
process SAOComps[macf,p,aei,sao,slk]:exit:=
    hide sacf,cd in
    (SACFFilt_AssocMan[macf,sacf,sao,slk] |[sacf,slk]|
    SACFRouter_ASEs[sacf,cd,sao,slk]  [> exit )
    |[cd]|
    (SACFConcatenator[cd,p,aei,sao] ||| ConcAllow[aei])
endproc
process SACFRouter_ASEs[sacf,cd,sao,slk]: noexit:=
    hide tp,ccr in
    (TPASE[sacf,tp,slk] |[slk]| UASE[sacf,tp,slk])
    |[tp,slk]|
    SACFRouter[sacf,tp,ccr,cd,sao,slk]
    |[ccr,slk]|
    (ACSE[ccr,cd,slk] |[slk]| CCR[ccr,cd,slk])
endproc
process TPASE[sacf,tp, lk]: noexit:=
    ( TPProcedures[sacf,tp] >> TPASE[sacf,tp, lk] ) [] lk?did:dc_id; TPASE[sacf,tp, lk]
endproc
```

H.1.3.1 SACF/Association Management

The SACFFilt_AssocMan process has an initial phase where an association is set up. This is performed by the EstablishAssociation process, which will either initiate the establishment of a new association as a local decision, or will accept an Association Establishment request. It exits if successful, otherwise it calls itself again.

The AssocMan process is responsible for ensuring that dialogue establishment requests received are compatible with this association.

```
process SACFFilt_AssocMan[macf,sacf,sao,slk] :noexit:=
    EstablishAssociation[macf,sacf]
>> accept cw : bool (* Contention Winner *), bm :bool  (* Bid Manadatory  *), amps :AMassocPars
    (*  Needed by assoc manager to test compatibility *)  in
    SACFFilter[macf,sacf,sao,slk](cw,bm) |[macf]| AssocMan[macf](amps)
endproc
```

The SACFQueuer provides all the queuing functionality for the SACF: the SACFProtocol process always issues sps on the sq gate irrespective of whether it is currently queuing. It is the responsibility of the SACFQueuer process to either pass the events directly onto the sacf gate, or hold onto them until the queue is flushed, as appropriate. The Protocol process instructs the Queuer when to queue, flush, discard etc. with events on the sq gate.

```
process  SACFFilter[macf,sacf,sao,slk](cw:bool,bm:bool):noexit:=
    hide sq in  SACFProtocol[macf,sq,sao,slk](new_sf_info(cw,bm)) |[sq,slk]|  SACFQueuer[sq,sacf,slk]
endproc
```

The SACFProtocol process is the same as the CMACFProtocol in structure. The SACFProtocol has one state vector parameter, of sort sf_info.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
process SACFProtocol[macf,sq,sao,slk](info:sf_info):noexit:=
    SACFProcedures[macf,sq,sao](info)  []  slk!get_did(info); exit(info)
    >> accept info:sf_info in SACFProtocol[macf,sq,sao,slk](info)
endproc
```

H.1.4 Coordination of the PM processes

H.1.4.1 Action Sequence Coordination

Each of the PM parts - the TP_MACF, C_MACF and SAO processes - contains an internal "lk" gate (the tlk, clk and slk gates respectively), and a coordination process. Within each part, the component processes synchronize with each other and with the coordinator of that part on the lk gate. The exception to this rule is the SACFConcatenator, as it does not form part of the action sequences and therefore has no lk gate.

Each component of each part will only allow an "unlock" event to occur if there is no processing currently being performed, so the relevant coordinator can tell when the action sequence has finished. The unlock events take the form tlk?dids:set_of_dids, clk?dids: set_of_dids and slk?did:dc_id respectively for each of the three PM parts. The parameter on this event is used by the MACF protocol processes to update the coordinator as to which SAOs are currently attached to it, and the SACFprotocol process uses the event to communicate the did currently assigned to the SAO to the coordinator.

Coordination of atomicity within each part then operates in principle as follows. The coordinator has two principal states: "unlocked" from where it will accept "locking events" which will change it to the locked state, and "locked" where it will only accept an "unlock" event indicating the end of the current action sequence, resetting the process.

Locking events to the TPPM are the input events on the tpsu or caf gates; an internal lock issued at the beginning of an internal action sequence; an begin dialogue indication issued on the macf gate, or a lock event issued by an SAO.

When an SAO is not in a pool, it must issue the aei!SAO_lock?did:dc_id event before beginning to process input from below. This event then acts as a locking event to the TPPM, and so can only be issued if the TPPM is not currently in an action sequence. The TPPMCoordinator determines if the lock event belongs to it by comparing the did on the event with the current value of its set_of_dids parameter. When the action sequence is begun in this way, the TPPM coordinator awaits for this SAO to finish processing, signalled by the aei!SAO_Unlock!did event, before looking for the TPPM unlock event as usual. A TPPM action sequence is also begun by an SAO passing a begin dialogue indication on the macf gate, as previously the SAO will not have been part of the TPPM.

```
process TPPMCoord[tpsu,caf,macf,aei,tlk](dids:set_of_dids):noexit:=
    tpsu ?did: dc_id ?sp: ServicePrim [IsReqRsp(sp)]; exit
    []
    tpsu ?sp: ServicePrim [IsReqRsp(sp)];exit
    []
    caf ?did: dc_id ?brid: branch_identifier ?sp: ServicePrim [IsInd(sp)];exit
    []
    macf ?did: dc_id ?addr: address ?rch: recovery_context_handle ?afsp: ServicePrim;
    exit (*begin dialogue indication*)
    []
    tlk !lock; exit                              (* Internal Event *)
    []
    aei !SAO_lock ?did: dc_id [did IsIn dids]; aei !SAO_Unlock !did; exit
>>
    tlk ?dids:set_of_dids; TPPMCoord[tpsu,caf,macf,aei,tlk](dids)
endproc
```

The CPMCoord process works in the same fashion as the TPPMCoord process

```
process CPMCoord[cpm,macf,aei,clk](dids:set_of_dids):noexit:=
    cpm ?sp:ServicePrim [IsReq(sp)]; exit
    []
    macf ?did: dc_id ?addr: address ?sp: ServicePrim; exit
    []
    clk !lock; exit                              (* Internal event *)
    []
    aei !sao_lock ?did: dc_id [did IsIn dids]; aei !sao_unlock !did; exit
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
>>
    clk ?dids: set_of_dids; CPMCoord[cpm,macf,aei,clk]( dids)
endproc
```

The SAOCoord contains a Relay process explained in the next section, which controls when the process exits.

```
process SAOCoord[macf,p,aei,sao,slk](did:dc_id):exit:=
    (   SAOAllow[macf]
        |[slk]|
        (   Relay[aei,sao,slk](did_absent) |[aei]| TrackToken[aei,sao,p](true) )
    )
    |||
    (   ActSeqCoord[macf,aei,sao,slk](did_absent) [> exit )
endproc
```

Locking events to the ActSeqCoord are:- input on the macf gate; aei!sao_lock - as explained above this is seen by the TPPMCoord, so the ActSeqCoord signals to the TPPM after it has seen an unlock an the slk gate; sao_lock - this can be issued by the concatenator before starting an action sequence if the SAO is not currently attached.

```
process ActSeqCoord[macf,aei,sao,slk](did:dc_id):noexit:=
    aei!sao_lock!did[not(is_absent(did))];
    (   slk?did:dc_id; exit(did)
        []
        macf!did!SAFAssociationLostInd;exit(did_absent)
        >> accept did:dc_id in aei!sao_unlock!did;
        ActSeqCoord[macf,aei,sao,slk](did)
    )
    []
(   sao!lock[is_absent(did)];exit [] sao!sacf_int_lock; exit
    []
    macf?did:dc_id?sp:ServicePrim[IsReqRsp(sp)];exit
    []
    macf?did:dc_id?acn:application_context_name?qosO:quality_of_service_Opt?
    addr:address? rch:recovery_context_handle_Opt?sp:ServicePrim;
    exit  (* BeginDialogue request *)
    []
    macf?did:dc_id?addr:address?sp:ServicePrim;
    exit  (*Begin Dialogue for channel *)
>>slk?did:dc_id; ActSeqCoord[macf,aei,sao,slk](did)
)
endproc
```

The relay process is used to determine which psap open/close events the concatenator should accept; relaying UASE aborts seen by the SACFRouter to the MACF when needed, choosing when to call a local error, and exiting accordingly, which will cause the SAO as a whole to exit. See Table 2 for more details of the coordination events used.

```
process Relay[aei,sao,slk](did:dc_id):exit:=
    (   aei!did!close_psap;exit(did)
        []
        aei!did!open_psap;exit(did)
        []
        slk?did:dc_id;exit(did)
        []
        aei!did!token?tok:bool;exit(did)
    >>  accept did:dc_id in
        Relay[aei,sao,slk](did)
    )
    []
    sao!sao_killed;exit
    []
    sao!UASE_abort;
    (   [not(is_absent(did))] -> aei!did!UASE_abort; exit
        []
        [is_absent(did)] -> exit
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    )
    []
    i;
    (   [not(is_absent(did))] -> aei!did!InternalError;exit
        []
        [is_absent(did)] -> exit
    )
endproc
process TrackToken[p,aei,sao](tok:bool):exit:=
(    aei?did:dc_id!token!tok;exit(tok)
    []
    sao!token!tok;exit(tok)
    []
    p?pad:p_address?psp:ServicePrim[Assigns_Token_Here(psp) (* PTokenGiveInd or PReSync/PInitiialize with the tokens
    parameter set appropriately *)]; exit(true)
    []
    p?pad:p_address?psp:ServicePrim[Assigns_Token_There(psp) (* token sent to reomte side *)];exit(false)
    []
    p?pad:p_address?psp:ServicePrim[not(Assigns_Token_Here(psp) or Assigns_Token_There(psp))]; exit(tok)
>> accept tok:bool in TrackToken[p,aei,sao](tok)
)
[]
exit
endproc
```

H.1.4.2 The TP_CInterface

This is a CPM to TPPM buffer paralleled with a process used for storing the log, and a process for transferring events form the TPPM to the CPM

```
process TP_CInterface[caf,cpm]:noexit:=
    TPtoC_Transfer[caf,cpm] ||| CtoTP_Buffer[caf,cpm] ||| LogStore[ caf, cpm]
endproc
process TPtoC_Transfer[caf,cpm]:noexit:=
    caf ?chid: dc_id ?sp:ServicePrim; cpm!chid!sp;exit
    []
    caf?sp:ServicePrim; cpm!sp;exit
>>  TPtoC_Transfer[caf,cpm]
endproc
process CtoTP_Buffer[ caf, cpm]: noexit :=
    (   hide end in Buffer[ caf, end, cpm] ) [> caf !tppm_crash; CtoTP_Buffer[ caf, cpm]
endproc
process Buffer[caf,mid,cpm]:noexit:=
    cpm ?chid: dc_id ?brid: branch_identifier ?sp:ServicePrim;
    (     hide end in mid; caf !chid !brid !sp; end; stop |[end]| Buffer[caf,end,cpm] )
endproc
```

The log is written by the TPPMProcedures on the caf gate, to enable this information to survive the node crash. This process will set the value of the log record following the caf !tppm_crash event.

```
process LogStore[ caf, cpm]: noexit :=
    EmptyLog[ caf, cpm] ||| BranchStore[ caf, cpm]
endproc
process EmptyLog[ caf, cpm]: noexit :=
    caf !write_log_record ?log: log_record;
    (   Log[ caf, cpm]( log)
        |||
        EmptyHeuristicLog[ caf]( get_aaid( log))
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        |||
        EmptyDamageLog[ caf, cpm]( get_aaid( log))
    )
    []
    caf !log_record_absent; EmptyLog[ caf, cpm]
endproc
process Log[ caf, cpm]( log: log_record): noexit :=
    [ is_ready( get_state( log)) ] -> caf !write_log_record ?log: log_record; Log[ caf, cpm]( log)
    []
    caf !read_log_record !log; Log[ caf, cpm]( log)
    []
    caf !forget_transaction; EmptyLog[ caf, cpm]
endproc
process EmptyHeuristicLog[ caf]( aaid: atomic_action_identifier): noexit :=
    caf !write_log_heuristic !aaid ?bds: bound_data_state; HeuristicLog[ caf]( aaid, bds)
    []
    caf !log_heuristic_absent !aaid; EmptyHeuristicLog[ caf]( aaid)
endproc
process HeuristicLog[ caf]( aaid: atomic_action_identifier, bds: bound_data_state): noexit :=
    caf !write_log_heuristic !aaid ?bds: bound_data_state;  HeuristicLog[ caf]( aaid, bds)
    []
    caf !remove_log_heuristic !aaid; EmptyHeuristicLog[ caf]( aaid)
    []
    caf !read_log_heuristic !aaid !bds; HeuristicLog[ caf]( aaid, bds)
endproc
process EmptyDamageLog[ caf, cpm]( aaid: atomic_action_identifier): noexit :=
    caf !write_log_damage !aaid ?hr: heuristic_report; DamageLog[ caf, cpm]( aaid, hr)
    []
    caf !log_damage_absent !aaid; EmptyDamageLog[ caf, cpm]( aaid)
    []
    cpm !log_damage_absent !aaid; EmptyDamageLog[ caf, cpm]( aaid)
endproc
process DamageLog[ caf, cpm]( aaid: atomic_action_identifier, hr: heuristic_report): noexit :=
    caf !write_log_damage !aaid !heuristic_hazard; DamageLog[ caf, cpm]( aaid, hr)
(*  ignore heuristic-hazard if damage-log already exists  *)
    []
    caf !write_log_damage !aaid !heuristic_mix; DamageLog[ caf, cpm]( aaid, heuristic_mix)
    []
    caf !remove_log_damage !aaid; EmptyDamageLog[ caf, cpm]( aaid)
    []
    caf !read_log_damage !aaid !hr; DamageLog[ caf, cpm]( aaid, hr)
    []
    cpm !read_log_damage !aaid !hr; DamageLog[ caf, cpm]( aaid, hr)
endproc
process BranchStore[ caf, cpm]: noexit :=
    caf !branch_list; BranchStore[ caf, cpm]
    []
    cpm !branch_list; BranchStore[ caf, cpm]
    []
    caf !branch_list ?aaid: atomic_action_identifier ?bl: branch_list;
    BranchList[ caf, cpm]( aaid, bl)
endproc
process BranchList[ caf, cpm]( aaid: atomic_action_identifier, bl: branch_list) : noexit :=
    caf !branch_list; BranchStore[ caf, cpm]
    []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    cpm !branch_list !aaid !bl; BranchList[ caf, cpm]( aaid, bl)
    []
    caf !branch_list ?aaid: atomic_action_identifier ?bl: branch_list;
    BranchList[ caf, cpm]( aaid, bl)
endproc
```

H.1.4.3 The Allow processes

These are generally to allow events that do not start an action sequence, and are therefore not of interest to the Coordinators, to occur freely.

```
process TPPMAllow[tpsu,caf,macf]:noexit:=
    tpsu?did:dc_id?sp:ServicePrim[IsIndCnf(sp)]; exit
    []
    caf?sp:ServicePrim[IsReq(sp)]; exit
    []
    caf?chid: dc_id ?sp:ServicePrim[IsReq(sp)]; exit
    []
    caf !tppm_crash; exit
    []
    LogAllow[ caf]
    []
    macf?did:dc_id?sp:ServicePrim; exit
    []
    macf !address_not_recognised ?addr: address; exit
    []
    macf ?did: dc_id ?acn:application_context_name ?qosO: quality_of_service_Opt ?addr: address ?rchO:
    recovery_context_handle_Opt ?sp: ServicePrim;
    exit
>>TPpmAllow[tpsu,caf,macf]
endproc
process CPMAllow[cpm,macf]:noexit:=
    cpm ?chid: dc_id ?brid: branch_identifier ?sp:ServicePrim[IsInd(sp)];exit
    []
    LogAllow[ cpm]
    []
    macf?did:dc_id?sp:ServicePrim; exit
    []
    macf?did:dc_id?addr:address?sp:ServicePrim;exit
>> CPMAllow[cpm,macf]
endproc
process LogAllow[ lg]: exit:=
    lg ?key: log_key; exit
    []
    lg ?key: log_key ?log: log_record; exit
    []
    lg ?key: log_key ?aaid: atomic_action_identifier; exit
    []
    lg ?key: log_key ?aaid: atomic_action_identifier ?hr: heuristic_report; exit
    []
    lg ?key: log_key ?aaid: atomic_action_identifier ?bd: bound_data_state; exit
    []
    lg ?key: log_key ?aaid: atomic_action_identifier ?bl: branch_list; exit
endproc
process SAOAllow[macf]:exit:=
    macf?did:dc_id?sp:ServicePrim[IsIndCnf(sp)and not(IsSAFAssociationLostInd(sp))]; SAOAllow[macf]
    []
    macf?did:dc_id?addr:address?rch:recovery_context_handle?sp:ServicePrim; SAOAllow[macf]
    []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    macf?did:dc_id?addr:address?sp:ServicePrim; SAOAllow[macf]
    []
    exit
endproc
process ConcAllow[aei]:exit:=
(   aei!sao_unlock?did:dc_id; exit
    []
    aei?did:dc_id!UASE_Abort; exit
    []
    aei?did:dc_id!InternalError;exit
>> ConcAllow[aei] )
[]
exit
endproc
process conformance_requirements[ tpsu, p]:noexit:=
    tpsu ?sp : ServicePrim; conformance_requirements[ tpsu, p]
    []
    p ?pad : p_address ?sp : ServicePrim; conformance_requirements[ tpsu, p]
endproc
```

H.1.5 Naming and Addressing

The standard does not always specify where various addressing parameters are to be obtained; however it does state that the provision of addressing information for use within the host system is a local matter.

TP service primitives relate to a particular dialogue but there is no mechanism for the TPSU to identify this dialogue. Service primitives and APDUs also need to identify which association they relate to: so that the TPPM can identify which SAO should deal with the given service primitive, and so that the SAO can send the APDUs to the correct remote address.

Within the MACFs we number the dialogue and channels with an internally generated number, unique to the MACF. This number, the dialogue-channel-identifier (of sort dc_id), is passed to the SAO which will support this dialogue or channel. The dialogue-channel-identifier appears with the service primitives on the gates used by the MACF to identify which dialogue or channel they relate to.

Connection with a remote TPSUI will be made on an association by an SAO, and the point where connection is made with the presentation layer is identified by the pair of presentation-service-access-point and presentation-service-end-identifier. The pair of presentation-service-access-point and presentation-service-end-identifier is termed a presentation-address, and this presentation-address will appear on all service primitives at the presentation gate to identify the association.

When the association manager sets up an association to add to the pool it sets up an SAO which issues an AAssociateRI (carried in the appropriate presentation service primitive) to the presentation layer. At this point the presentation layer tells the SAO the presentation-address it will be using. When an SAO is set up on the receipt of an AAssociateRI, it records the presentation-address carried with the AAssociateRI.

When the TPSU initiates a dialogue by sending a TPBeginDialogue request, it will be told the dialogue(-channel)-identifier which the TP-MACF will use to identify this dialogue. Dialogues are simply numbered sequentially as TPBeginDialogue requests are received on the TPSU gate.

When a dialogue is established by the receipt of a AFBeginDialogue indication, it is given a unique number which identifies it as the superior dialogue. This is possible because an individual TP-MACF will only ever receive one AFBeginDialogue indication. This dialogue(-channel)-identifier is passed to the TPSU with the TPBeginDialogue indication.

Channels are established by the issuing of an AFBeginDialogue request or the receipt of an AFBeginDialogue indication, and the C-MACF will number all channels consecutively, as they are established.

The dialogue/channel identifier will then be attached to any resulting (TP-, AF-, SAF-, CCR-, UASE-, ACSE-, or presentation-) service primitives which are issued on the **tpsu, tppm, caf** or **macf** gates for a particular dialogue/channel.

If a service primitive does not relate to a specific dialogue, there will be no application-identifier. The service primitive may result in the MACF issuing service primitives on some or all of the current dialogues.

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

On the **caf** and **cpm** gates, CAF Please requests are issued with no addressing information, other (AF-, CCR-, and CAF-) service primitives will be issued with dialogue-channel-identifier of the channel to which they relate.

The gates within an SAO (**sacf**, **ase**, and **cd**) are seen only by that SAO so no addressing information is needed.

When the SAO sends a presentation service primitive (possibly containing an individual APDU or a queue of APDUs), it is sent with the presentation-address which identify the association. Similarly when a presentation service primitive is issued to the SAOs, it has attached a presentation-address so it will be received by the right SAO.

The table below shows the addressing information carried on various gates in the LOTOS specification.

Table H.2 - Summary of service primitives / PDUs and addressing information on the gates

Gate	Service primitives and PDUs on the gate		Addressing information
tpsu, tppm	TP-service primitives	not multi-dialogue	dialogue-channel-identifier
		multi-dialogue[a]	
caf, cpm	CAF-Give Ind, CAF-Fail Ind, CAF-Recover Ind		dialogue-channel-identifier & branch-identifier
	CAF-Detach Req		dialogue-channel-identifier
	CAF-Please Req		
macf	SAF- , AF- , CCR- , UASE- , ACSE- service primitives, P-TokenGive Req/Ind, P-TokenPlease Req/Ind		dialogue-channel-identifier
sacf, sq	AF- , CCR- , UASE- , ACSE- service primitives P-TokenGive Req/Ind, P-TokenPlease Req/Ind		
ccr	CCR- , ACSE- service primitives		
tp, cd	TP- , UASE- , CCR- , ACSE- PDUs, P-service primitives		input_output (i.e. up / dn)
p	P-service primitives		presentation-address

a. Multi-dialogue TP-Service primitives are TP-Commit Req/Ind, TP-Rollback Req/Ind, TP-CommitComplete Ind, TP-RollbackComplete Ind and TP-Done Req

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

H.2 The SACF and the Association Manager

H.2.1 Association Management Procedures

Establishing an Association

This process is responsible for setting up the association for the SAO.

```
process EstablishAssociation[macf,sacf]:exit(bool,bool,AMassocPars) :=
    (   i; InitAssocEstab[sacf] >> accept pvs:protocol_version in
        ReceiveAssocEstabCnf[macf,sacf](pvs)
    )
    []
    (   ReceiveAssocEstabInd[sacf]  >> accept sp:ServicePrim, pdu:PDU, tp_aed: tp_assoc_estab_diagnostic_Opt in
        RespAssocEstab[macf,sacf](sp, pdu, tp_aed)
    )
endproc
process ReceiveAssocEstabInd[sacf]: exit(ServicePrim,PDU,tp_assoc_estab_diagnostic_Opt):=                    (* 8.5.5 *)
    sacf?sp:ServicePrim[IsAAssocInd(sp) and has_embedded_AssocEstabRI(sp)];
    (   let pdu:PDU = get_pdu(sp),
(* Establish which diagnostic values are to be set.  *)
            ccr_version_2_not_available : bool = HasCCR(get_acn(sp)) and ( not(HasCCR(get_pcdl(sp))) or
            not(HasCCRReqs(get_sr(sp))) ),
            tp_protocol_version_incompatibility : bool = not(my_version <= get_pv(get_pdu(sp)) ) in
(* Choose the Bid/CW diagnostics as a local matter ... *)
        choice  contention_winner_assignment_rejected : bool,  bid_mandatory_rejected : bool  []
        let no_reason_given: bool = not(tp_protocol_version_incompatibility or ccr_version_2_not_available or
        contention_winner_assignment_rejected or bid_mandatory_rejected) in
        exit( sp, pdu, TPAEdiag(tp_protocol_version_incompatibility,  ccr_version_2_not_available,
        contention_winner_assignment_rejected,  bid_mandatory_rejected, no_reason_given))
    )
endproc
process RespAssocEstab[macf,sacf](sp:ServicePrim, pdu:PDU,tp_aed: tp_assoc_estab_diagnostic_Opt):
exit(bool,bool,AMassocPars):=                                                                                (* 8.5.5a *)
    [is_absent(tp_aed)] -> i;
(* Must have all diagnostic parameters false as precondition for accepting *)
(* Choose to accept ... Note although no requirements are placed on the acn and QOS used on the AAssocRsp in subclauses
    8.5.5(a), the values of these on the response are the ones used on the association; therefore they need to be known to tell which
    dialogue/channels the association will be compatible with.*)
    (   choice qos_to_use:QOS_ISO8326, acn_to_use: application_context_name,
        rch:recovery_context_handle_Opt, resp_addr : address []
(* If iaot is acceptor chooses, set value here *)
        [is_acceptor_chooses(get_iaot(sp))] -> ( i; exit(acceptor_side) [] i; exit(requestor_side) )
        []
        [not(is_acceptor_chooses(get_iaot(sp)))] -> exit(get_iaot(sp))
    >>  accept iaot:initial_assignment_of_tokens in
        sacf !AAssocRsp( acn_to_use, resp_addr, tp_initialize_rc(my_version,rch,aed_absent), accepted,  qos_to_use,iaot)
        [ resp_addr <= my_addr];
        exit(get_cwa(pdu), get_bid(pdu), AMassocPars(get_cllng(sp), get_rchO(pdu), acn_to_use,  qos_to_use))
    )
    []
    i;                                          (* Reject because diag set or otherwise as local decision *)
    (   choice result : a_associate_result ,acn :application_context_name,  resp_addr: address []
        sacf!AAssocRsp(acn, resp_addr,  tp_initialize_rc(my_version,rch_absent,tp_aed), result , get_qos(sp), get_iaot( sp))
        [ (resp_addr <= my_addr) and (pvi_or_ccrv2(tp_aed)  implies is_rejected_perm(result)) and (cwr_or_bmr(tp_aed) implies
        is_rejected_trans(result))];
        EstablishAssociation[macf,sacf]
    )
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
process InitAssocEstab[sacf]:exit(protocol_version):= (*8.5.4 *)
    choice pvs : protocol_version, qos:QOS_ISO8326, called_addr, calling_addr : address,
    acn:application_context_name, rch: recovery_context_handle_Opt, aam: a_associate_mode,
    pcdl:presentation_context_definition_list, sr:session_requirements,
    cw:bool, bm : bool , iaot:initial_assignment_of_tokens []
    sacf! AAssocReq(aam, acn, calling_addr , called_addr,
    tp_initialize_ri(pvs ,cw, bm, rch),  pcdl, qos, sr, iaot)
    [ ( HasCCR(acn) implies  ( HasCCRreqs(sr) and HasCCR(pcdl)) )
    and  (calling_addr <= my_addr) and (pvs <= my_version)
    and (HasCCR(acn) implies ( (cw implies is_requestor_side(iaot)
                and (not(cw)  implies is_acceptor_side(iaot)))))];
(* my_addr and my_version are constants defined in the datatyping *)
    exit(pvs)
endproc
process  ReceiveAssocEstabCnf[macf,sacf]
(pvs : protocol_version):exit(bool, bool, AMassocPars) :=
    sacf?sp:ServicePrim[IsAAssocCnf(sp) and has_embedded_AssocEstabRC(sp)];
    (   let pdu:PDU = get_pdu(sp) in
        [is_accepted(get_aar(sp)) and (get_pv(pdu) <= pvs) ] ->
        (   [HasCCR(get_acn(sp)) implies  ( HasCCRreqs(get_sr(sp)) and
            HasCCR(get_pcdl(sp)) )] ->
                exit(get_cwa(pdu), get_bid(pdu), AMassocPars(get_cllng(sp), get_rchO(pdu), get_acn(sp),  get_qos(sp)))
            []
            [not(HasCCR(get_acn(sp)) implies  ( HasCCRreqs(get_sr(sp)) and
            HasCCR(get_pcdl(sp)) ))] ->
(* The PM will never be able to use this association, therefore it is just forgotten here *)
                EstablishAssociation[macf,sacf]
        )
        []
        [not(is_accepted(get_aar(sp))) and is_called_addr_not_recognised(get_aadO(sp))] ->
        AddressUnrecognised[macf](get_addr(sp)) (* Tells MACF this address unrecognised *)
        []
        [not(is_accepted(get_aar(sp))) and not(is_called_addr_not_recognised(get_aadO(sp)))] ->
        EstablishAssociation[macf,sacf] (* Reuse this SAO *)
        []
        [is_accepted(get_aar(sp)) and not(get_pv(pdu) <= pvs) ] ->
(* must release *)
            sacf!AReleaseReq; sacf!AReleaseCnf(affirmative);
            EstablishAssociation[macf,sacf]
    )
endproc
(* This process remembers the failed attempt to set up the association with this address - if the MACF has a begin dialogue request
    for a matching address, then it can reject it (if it chooses) with a recipient unknown diagnostic *)
process AddressUnrecognised[macf](addr:address):noexit :=
    macf!Address_not_recognised?addr1:address[addr1 <= addr];AddressUnrecognised[macf](addr)
endproc
(*The following process is responsible for the association once it has been set up. Note the SACF is responsible for ensuring it
    is in the FREE state when receiving an AFBeginDiagReq. Also, the SACF is responsible for issuing ARelease/Abort in the
    FREE state, as the AssocManager doesn't know which state the SACF is in. *)
process AssocMan[macf](amps:AMassocPars): noexit :=                                    (*8.5.2/3*)
    CheckDialogueCompat[macf](amps)
    |||
    CheckChannelCompat[macf](amps)
    |||
    QualifyBegDInd[macf](amps)
    |||
    AllowOtherMacfEvts[macf]
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
endproc
process CheckDialogueCompat[macf](amps :AMassocPars): noexit:=                    (* 8.5.2 *)
    let qosOffer:QOS_ISO8326 = get_qos(amps),
        addr:address= get_addr(amps),
        acnOffer: application_context_name = get_acn(amps),
        rchO:recovery_context_handle_Opt = get_rchO(amps)   in
        macf?did:dc_id?acn:application_context_name?qosO:quality_of_service_Opt!addr!rchO
        ?sp:ServicePrim [IsAFBeginDialogueReq(sp) and  (* Check compatibility for dialogues *)
(* a) address compatibility checked in the MACF *)
(* those that fail already thrown away*)
(* c) pcdl requirement as above, Check here if CCR in acn if needed *)
    (    has_commit(sp) implies HasCCR(acn)) and
(* d) Check quality of service compatible *)
    (    is_absent(qosO) or (get_qos(qosO) <= qosOffer) ) and
(* e)  Check acn meets requirements given by acn on the TPBegDreq *)
        (acn <=acnOffer) ]; CheckDialogueCompat[macf](amps)
endproc
process CheckChannelCompat[macf](amps :AMassocPars): noexit:=                     (* 8.5.2 *)
    let addr:address= get_addr(amps),
    acnOffer: application_context_name = get_acn(amps), rch:recovery_context_handle_Opt = get_rchO(amps)   in
        macf?did:dc_id!addr?sp:ServicePrim
        [IsAFBeginDialogueReq(sp)  and (* a) + b) as above, c) Check CCR in acn *) HasCCR(get_acn(sp))];
        CheckChannelCompat[macf](amps)
endproc
(* The following process is used to set various parameters needed by the MACF on a Begin dialogue indication *)
process QualifyBegDInd[macf](amps:AMassocPars):noexit:=
    macf?did:dc_id!get_addr(amps)!get_rchO(
    amps)?sp:ServicePrim[IsAFBeginDialogueInd(sp)]; QualifyBegDInd[macf](amps)
    []
    macf?did:dc_id!get_addr(amps)?sp:ServicePrim[IsAFBeginDialogueInd(
    sp)]; QualifyBegDInd[macf](amps)
endproc
process AllowOtherMacfEvts[macf]:noexit:=
    macf?did:dc_id?sp:ServicePrim[not(IsAFBeginDialogueReq(sp) or
    IsAFBeginDialogueInd(sp))];exit
    []
    macf?did:dc_id!SACF_PE;exit
>> AllowOtherMacfEvts[macf]
endproc
```

H.2.2 TP- ASE Procedure Processes

```
process TPProcedures[sacf,tp]: exit:=                                             (* 9.4 *)
    sacf ?afsp:ServicePrim [ IsAFBeginDialogueReq(afsp) and is_dialogue(afsp)];  (* 9.4.1  *)
    tp!dn  !tp_begin_dialogue_ri(get_ittO(afsp),get_rttO(afsp),get_sfu(afsp),   get_btO(afsp), get_bdc(afsp),nat(get_dccO(
    afsp)), get_lpiO(afsp), get_ud(afsp));
    exit
    []
    sacf ?afsp:ServicePrim [ IsAFBeginDialogueReq(afsp) and is_channel(afsp)];
    tp!dn  !tp_begin_dialogue_ri( get_sfu(afsp), nat( get_dccO( afsp)),  get_cu(afsp), get_lpiO(afsp));
    exit
    []                                                                            (* 9.4.2 *)
    tp!up ?tpdu:PDU [Istp_D_begin_dialogue_ri(tpdu) ];
    sacf !AFBeginDialogueInd(get_ittO(tpdu),get_rttO(tpdu),get_sfu(tpdu),  get_btO(tpdu), get_bdc(tpdu),
    correlator(get_dcc(tpdu)), get_lpiO(tpdu),get_ud(tpdu));
    exit
    []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
tp!up ?tpdu:PDU [Istp_C_begin_dialogue_ri(tpdu)];
sacf !AFBeginDialogueInd(get_sfu(tpdu), correlator(get_dcc(tpdu)),get_cu(tpdu),  get_lpiO(tpdu));
exit
[]                                                                    (* 9.4.3 *)
sacf ?afsp:ServicePrim [IsAFBeginDialogueRsp(afsp) and is_dialogue(afsp)];
(   let tpdu:PDU = tp_begin_dialogue_rc(get_fuO(afsp), get_bdr(afsp),
    get_bddO(afsp), nat(get_dccO( afsp)), get_ud(afsp)) in
    [is_map_dataRI(afsp)] -> tp!dn  !tpdu;exit
    []
    [is_map_rollbackRI(afsp)] -> tp!dn  !CRollbackReq(tpdu); exit
    []
    [is_map_rollbackRC(afsp)] -> tp!dn  !CRollbackRsp(tpdu); exit
)
[]
sacf ?afsp:ServicePrim [IsAFBeginDialogueRsp(afsp) and is_channel(afsp)];
(   let tpdu:PDU = tp_begin_dialogue_rc( get_bdr(afsp), get_bcdO(afsp),
    nat(get_dccO(afsp))) in [is_map_dataRI(afsp)] -> tp!dn  !tpdu;exit )
[]                                                                    (* 9.4.4 *)
tp!up ?tpdu:PDU [Istp_D_begin_dialogue_rc(tpdu)];
sacf !AFBeginDialogueCnf( get_fuO(tpdu), get_bdr(tpdu), get_bddO(tpdu),  correlator(get_dcc(tpdu)), dataRI,
get_ud(tpdu));
exit
[]
tp!up ?tpdu:PDU [Istp_C_begin_dialogue_rc(tpdu)];
sacf !AFBeginDialogueCnf(get_bdr(tpdu),get_bcdO(tpdu),  correlator(get_dcc(tpdu)), dataRI);
exit
[]                                                                    (* 9.4.5 *)
sacf ?afsp:ServicePrim [IsAFBidReq(afsp)];  tp!dn  !tp_bid_ri(get_ctr(afsp),get_lpiO(afsp)); exit
[]                                                                    (* 9.4.6 *)
tp!up ?tpdu:PDU [Istp_bid_ri(tpdu)];  sacf !AFBidInd(get_ctr(tpdu),get_lpiO(tpdu)); exit
[]                                                                    (* 9.4.7 *)
sacf ?afsp:ServicePrim [IsAFBidRsp(afsp)]; tp!dn  !tp_bid_rc(get_br(afsp)); exit
[]                                                                    (* 9.4.8 *)
tp!up ?tpdu:PDU [Istp_bid_rc(tpdu)];  sacf !AFBidCnf(get_br(tpdu)); exit
[]                                                                    (* 9.4.9*)
sacf ?afsp:ServicePrim [IsAFEndDialogueReq(afsp)];  tp!dn  !tp_end_dialogue_ri(get_edc(afsp)); exit
[]                                                                    (* 9.4.10 *)
tp!up ?tpdu:PDU [Istp_end_dialogue_ri(tpdu)];  sacf !AFEndDialogueInd(get_edc(tpdu)); exit
[]                                                                    (* 9.4.11 *)
sacf ?afsp:ServicePrim [IsAFEndDialogueRsp(afsp)];  tp!dn  !tp_end_dialogue_rc; exit
[]                                                                    (* 9.4.12 *)
tp!up ?tpdu:PDU [Istp_end_dialogue_rc(tpdu)];  sacf !AFEndDialogueCnf; exit
[]                                                                    (* 9.4.13 *)
sacf ?afsp:ServicePrim [IsAFUErrorReq(afsp)];  tp!dn  !tp_U_error_ri; exit
[]                                                                    (* 9.4.14 *)
tp!up ?tpdu:PDU [Istp_U_error_ri(tpdu)];  sacf !AFUErrorInd; exit
[]                                                                    (* 9.4.15 *)
sacf ?afsp:ServicePrim [IsAFUErrorRsp(afsp)];  tp!dn  !tp_U_error_rc; exit
[]                                                                    (* 9.4.16 *)
tp!up ?tpdu:PDU [Istp_U_error_rc(tpdu)];  sacf !AFUErrorCnf; exit
[]                                                                    (* 9.4.17 *)
sacf ?afsp:ServicePrim [IsAFAbortReq(afsp)];
(   let tpdu:PDU  = make_tp_abort_ri( afsp) in
    [is_map_dataRI(afsp)] -> tp!dn  !tpdu; exit
    []
    [is_map_abortRI(afsp)] -> tp!dn  !AAbortReq( tpdu); exit
    []
    [is_map_commitRI(afsp) ] -> tp!dn  !CCommitReq( tpdu); exit
    []
    [is_map_commitRC(afsp)] -> tp!dn  !CCommitRsp( tpdu); exit
    []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    [is_map_rollbackRI(afsp)] -> tp!dn  !CRollbackReq( tpdu); exit
    []
    [is_map_rollbackRC(afsp)] -> tp!dn  !CRollbackRsp( tpdu); exit
)
[]                                                              (* 9.4.18 *)
tp!up ?tpdu:PDU [Istp_abort_ri( tpdu)];  sacf !make_AFAbortInd(tpdu, dataRI); exit
[]                                                              (* 9.4.19*)
sacf ?afsp: ServicePrim [ IsAFGrantControlReq(afsp) ];  tp!dn  !tp_grant_control_ri; exit
[]                                                              (* 9.4.20 *)
tp!up ?tpdu: PDU [ Istp_grant_control_ri(tpdu) ];  sacf !AFGrantControlInd; exit
[]                                                              (* 9.4.21*)
sacf ?afsp: ServicePrim [ IsAFRequestControlReq(afsp) ];  tp!dn  !tp_request_control_ri; exit
[]                                                              (* 9.4.22 *)
tp!up ?tpdu: PDU [ Istp_request_control_ri(tpdu) ];  sacf !AFRequestControlInd; exit
[]                                                              (* 9.4.23 *)
sacf ?afsp: ServicePrim [ IsAFHandshakeReq( afsp) ];  tp!dn  !tp_handshake_ri( get_cnfuO( afsp)); exit
[]                                                              (* 9.4.24 *)
tp!up ?tpdu: PDU [ Istp_handshake_ri( tpdu) ];  sacf !AFHandshakeInd( get_cnfuO( tpdu)); exit
[]                                                              (* 9.4.25 *)
sacf ?afsp: ServicePrim [ IsAFHandshakeRsp( afsp) ];  tp!dn  !tp_handshake_rc; exit
[]                                                              (* 9.4.26 *)
tp!up ?tpdu: PDU [ Istp_handshake_rc( tpdu) ];  sacf !AFHandshakeCnf; exit
[]                                                              (* 9.4.27 *)
sacf ?afsp: ServicePrim [ IsAFHandshakeAndGrantControlReq( afsp) ];
tp!dn  !tp_handshake_and_grant_control_ri( get_cnfu( afsp));
exit
[]                                                              (* 9.4.28 *)
tp!up ?tpdu: PDU [ Istp_handshake_and_grant_control_ri( tpdu) ];
sacf !AFHandshakeAndGrantControlInd( get_cnfu( tpdu)); exit
[]                                                              (* 9.4.29 *)
sacf ?afsp: ServicePrim [ IsAFHandshakeAndGrantControlRsp( afsp) ];
tp!dn  !tp_handshake_and_grant_control_rc; exit
[]                                                              (* 9.4.30 *)
tp!up ?tpdu: PDU [ Istp_handshake_and_grant_control_rc( tpdu) ];
sacf !AFHandshakeAndGrantControlCnf;  exit
[]                                                              (* 9.4.31 *)
sacf ?afsp : ServicePrim [ IsAFDeferReq( afsp) ];
tp!dn  !tp_defer_ri( get_dt( afsp)); exit
[]                                                              (* 9.4.32 *)
tp!up ?tpdu : PDU [ Istp_defer_ri( tpdu) ];  sacf !AFDeferInd( get_dt( tpdu)); exit
[]                                                              (* 9.4.33 *)
sacf ?afsp : ServicePrim [ IsAFPrepareReq( afsp) ];
tp!dn  !CPrepareReq(  tp_prepare_ri(get_dpO(afsp))); exit
[]                                                              (* 9.4.34 *)
tp!dn  ?csp : ServicePrim [ IsCPrepareInd( csp) and has_embedded_Prepare( csp)];
sacf !AFPrepareInd(get_dpO(csp)); exit
[]                                                              (* 9.4.35 *)
sacf ?afsp:ServicePrim [IsAFHeuristicReportReq(afsp)];
(   let tpdu:PDU = tp_heuristic_report_ri(get_hr(afsp)) in
    [is_map_commitRC(afsp)] -> tp!dn  !CCommitRsp(tpdu); exit
    []
    [is_map_recover_doneRC(afsp)] ->
    tp!dn  !CRecoverRsp(done, get_aaid(afsp),get_brid(afsp), tpdu); exit
    []
    [is_map_rollbackRI(afsp)] -> tp!dn  !CRollbackReq(tpdu); exit
    []
    [is_map_rollbackRC(afsp)] -> tp!dn  !CRollbackRsp(tpdu); exit
)
[]                                                              (* 9.4.36 *)
sacf ?afsp:ServicePrim
[IsAFAbortAndHeuristicReportReq(afsp)];
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
(   let tpdus: PDUqueue = tp_heuristic_report_ri(get_hr(afsp)) +
    tp_abort_ri(user, get_ud( afsp)) in
    [is_map_rollbackRI(afsp)] -> tp!dn  !CRollbackReq( tpdus); exit
    []
    [is_map_rollbackRC(afsp)] -> tp!dn  !CRollbackRsp( tpdus);exit
    []
    [is_map_commitRC(afsp)] -> tp!dn  !CCommitRsp( tpdus);exit
)
[]                                                          (* 9.4.37 *)
sacf ?afsp:ServicePrim [ IsAFRecoverReq(afsp)];
tp!dn  !CRecoverReq(get_rsri(afsp), get_aaid(afsp), get_brid(afsp),  tp_recover_ri(get_rch(afsp))); exit
[]                                                          (* 9.4.38 *)
tp!up ?csp: ServicePrim [ IsCRecoverInd( csp) and has_embedded_Recover( csp)];
sacf !AFRecoverInd( make_map( get_rsri( csp)), get_rch(get_pdu( csp))); exit
[]                                                          (* 9.4.39 *)
tp!up ?asp:ServicePrim [ IsAAbortInd(asp) and has_embedded_Abort( asp)];
sacf !AFAbortInd(provider, abortRI, get_pad( get_pdu( asp))); exit
[]                                                          (* 9.4.40, 41, 42, 43, 44 *)
tp!up ?csp:ServicePrim [(IsCRollbackInd(csp) or IsCRollbackCnf(csp) or  IsCCommitCnf(csp))
and has_embedded_AbortAndHeuristicReport(csp)];
(   let tpdu:PDU = get_pdu(csp) in
    sacf !AFAbortAndHeuristicReportInd(make_map(csp), get_hr(tpdu),
    get_ud( tpdu)); exit )
[]
tp!up ?csp:ServicePrim [(IsCRollbackInd(csp) or IsCRollbackCnf(csp) or  IsCCommitInd(csp) or  IsCCommitCnf(csp) )
and  has_embedded_Abort(csp)];
(   let tpdu:PDU = get_pdu(csp) in
    sacf !make_AFAbortInd( tpdu, make_map(csp)); exit )
[]
tp!up ?csp:ServicePrim [(IsCRollbackInd(csp) or IsCRollbackCnf(csp) or  IsCCommitCnf(csp) or IsCrecoverCnf(csp))
and has_embedded_HeuristicReport(csp)];
(   let tpdu:PDU = get_pdu(csp) in  sacf
    !AFHeuristicReportInd(make_map(csp), get_hr(tpdu)); exit )
[]
tp!up ?csp:ServicePrim [(IsCRollbackInd(csp) or IsCRollbackCnf(csp)) and
has_embedded_BeginDialogueRC(csp)];
(   let tpdu:PDU = get_pdu(csp) in
    sacf !AFBeginDialogueCnf(get_fuO(tpdu), get_bdr(tpdu), get_bddO(tpdu),  correlator(get_dcc(tpdu)), make_map(csp),
    get_ud(tpdu));
    exit
)
[]                                                          (* 9.4.45 *)
tp!up ?pdu: PDU [ istp_token_give_ri( pdu) ];
sacf !AFTokenGiveInd( get_reason( pdu), get_dccO(pdu)); exit
[]                                                          (* 9.4.46 *)
sacf ?afsp:ServicePrim [ IsAFTokenGiveReq(afsp)];
tp!dn  !tp_token_give_ri( get_reason( afsp), get_dccO(afsp)); exit
[]                                                          (* 9.4.47 *)
tp!up ?pdu: PDU [ istp_token_please_ri( pdu) ];
sacf !AFTokenPleaseInd; exit
[]                                                          (* 9.4.48 *)
sacf ?afsp:ServicePrim [ IsAFTokenPleaseReq(afsp)];
tp!dn !tp_token_please_ri; exit
endproc
```

H.2.3 SACF Queuing

The following process is used to model the requirements for queuing in the SACF filter.

```
process SACFQueuer[sq, sacf, slk]:exit:=
    ((  Transfer[sq,sacf, slk] |[ slk ]| NotQueuing[sq, sacf ,slk](absentSP)) |[sq]| Monitor[ sq]( absentSP)) [> exit
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
endproc
process Transfer[sq, sacf, slk]:noexit:=
    sacf?sp:ServicePrim [ IsIndCnf( sp) ]; sq!sp ; Transfer[sq, sacf, slk]
    []
    sq !pass_to_UASE ?sp : ServicePrim; sacf !pass_to_UASE !sp ; Transfer[sq, sacf, slk]
    []
    slk ?did : dc_id; Transfer[sq, sacf, slk]
endproc
(* this process monitors the last service primitive sent or received by the SACF. This is so it can tell the SACF process this
   information when  interrogated with the REM event on the sq gate.*)
process Monitor[ sq]( last_sp: ServicePrim): noexit :=
    sq ?sp: ServicePrim; Monitor[ sq]( sp)
    []
    sq !REM ?last_SP_sent: ServicePrim !last_sp; Monitor[ sq]( last_sp)
    []
    sq !pass_to_UASE ?sp : ServicePrim; Monitor[ sq]( last_sp)
    []
    sq ?flk: flush_key; Monitor[ sq]( last_sp)
endproc
(* Note: the following process keeps track of the last service primitive it has received. This is so it can tell the SACF this
   information when  interrogated with the REM event on the sq gate. The last service primitive is not remembered whilst
   queuing, since the SACF never needs to know the last sp when there is a queue.*)
process NotQueuing[sq, sacf,slk](last_SP_sent:ServicePrim):noexit:=
    slk?did:dc_id;NotQueuing[sq, sacf ,slk](last_SP_sent)
    []
    sq?sp:ServicePrim; sacf!sp; NotQueuing[sq, sacf ,slk](sp)
    []
    sq!REM!last_SP_sent ?last_sp: ServicePrim; NotQueuing[sq, sacf ,slk](last_SP_sent)
    []
    sq!queue; ( Queuing[sq, sacf, slk](emptySP) >> NotQueuing[sq, sacf, slk](absentSP) )
    []
    sq!discard;NotQueuing[sq, sacf,slk](last_SP_sent) (* Ignore *)
endproc
process  Queuing[sq, sacf,slk](spq:ServicePrimQueue): exit :=
    sq?sp:ServicePrim[not(IsAFAbortReq(sp) and is_map_abortRI(sp))]; Queuing[sq, sacf, slk](spq + sp)
    []
    sq?sp:ServicePrim[IsAFAbortReq(sp) and is_map_abortRI(sp)]; sacf!sp; exit
    []
    slk?did:dc_id; Queuing[sq, sacf ,slk](spq)
    []
    sq!queue; Queuing[sq, sacf,slk](spq)      (* Ignore if already queuing *)
    []
    sq?flk:flush_key;
    (   [IsDiscard(flk)] -> exit
        []
        [IsFlush(flk)] -> FlushQueue[sacf](spq)
        []
        [IsToCBegin(flk)] -> ( FlushtoCbeg[sacf](spq) >>
        accept spq:ServicePrimQueue in  Queuing[sq,sacf,slk]( spq) )
    )
endproc
process FlushQueue[sacf](spq: ServicePrimQueue) : exit :=
    [IsEmpty(spq)] -> exit
    []
    [not(IsEmpty(spq))] -> sacf! head(spq); FlushQueue[sacf](tail(spq))
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
process FlushtoCbeg[sacf](spq: ServicePrimQueue) : exit(ServicePrimQueue) :=
    [IsCBeginReq(head(spq))] -> exit(spq)
    []
    [not(IsCBeginReq(head(spq)))] ->
    sacf! head(spq); FlushtoCbeg[sacf](tail(spq))
endproc
process SACFRouter[sacf,tp,ccr,cd,sao,slk]: noexit:=
    slk?did:dc_id; exit
    []
    cd !up?sp:ServicePrim[ IsPTokenGiveInd( sp) or IsPTokenPleaseInd( sp) ]; sacf!sp; exit
    []
    cd !up?sp:ServicePrim[IsOtherPresentationIndCnf(sp)]; tp!up!sp; exit          (* presentation service primitive for UASE *)
    []
    cd!up?pdu:PDU; tp!up!pdu; exit
    []
    sacf?sp:ServicePrim[IsPTokenGiveReq(sp) or IsPTokenPleaseReq(sp)]; cd !dn !sp; exit
    []
    sacf !pass_to_UASE ?sp : ServicePrim; tp !up !sp; exit                        (* Passed up to UASE *)
    []
    tp!dn?pdu:PDU; cd!dn!pdu; exit
    []
(* Check if Abort/Release is issued to/from ACSE, and exit if it is - this will cause the whole of the SAO to exit after all the parts
    have finished dealing with this sp or resultant ADUs.*)
    (   ccr?sp:ServicePrim[has_embedded_pdu(sp) and IsIndCnf( sp) ]; tp !up !sp; exit(sp)                    (* 6.1.6 *)
        []
        ccr?sp:ServicePrim[not(has_embedded_pdu(sp)) and IsIndCnf( sp) ]; sacf!sp ; exit(sp)
        []
        tp !dn ?sp:ServicePrim[IsPsp(sp)]; cd !dn !sp; exit(sp)
        []
        tp !dn ?sp:ServicePrim[not(IsPsp(sp))]; ccr!sp; exit(sp)
    >> accept sp: ServicePrim in
        [(IsAAbortInd(sp) and has_embedded_Abort(sp)) or IsAReleaseCnf(sp)] ->
        sao!sao_killed;exit
        []
        [IsAAbortInd(sp) and not(has_embedded_Abort(sp))] ->
        sao!UASE_abort; exit
        []
        [not(IsAAbortInd(sp) or IsAReleaseCnf(sp))] -> exit
        []
        [(IsAAbortReq(sp) and has_embedded_Abort(sp)) or IsAReleaseRsp(sp)] ->
        sao!sao_killed;exit
        []
        [IsAAbortReq(sp) and not(has_embedded_Abort(sp))] ->
        sao!UASE_abort; exit
        []
        [not(IsAAbortReq(sp) or IsAReleaseRsp(sp))] -> exit
    )
>> SACFRouter[sacf,tp,ccr,cd,sao, slk]
endproc
```

H.2.4 SACF Filter Procedures

```
(* The SACF uses the did given to it by its macf to qualify which input belongs to it. When it is not attached to any MACF, the
    value "did_absent" is used. *)
process  SACFProcedures[macf,sq,sao]  (info: sf_info): exit(sf_info) :=          (* 10.4 10.5, 10.6 *)
    SF_SafDetachReq[macf, sq, sao](info)
    []
    SF_AFBgnDgReq[macf, sq](info)
    []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    SF_AFBegDgInd[macf, sq, sao](info)
    []
    SF_AFBgnDgRsp[macf, sq](info)
    []
    SF_AFBgnDgCnf[macf, sq](info)
    []
    SF_AFBidInd[macf, sq, sao](info)
    []
    SF_AFBidCnf[macf, sq](info)
    []
    SF_AbortInd[macf,sq](info)
    []
    SF_AFTokGiveInd[macf,sq](info)
    []
    SF_AFTokGiveReq[macf,sq,sao](info)
    []
    SF_CBgnReq[macf, sq, sao](info)
    []
    SF_CBgnInd[macf, sq, sao](info)
    []
    SF_CRbkInd[macf, sq, sao](info)
    []
    SF_AFAbrtInd_Rbk[macf, sq, sao](info)
    []
    SF_CRbkCnf[macf, sq, sao](info)
    []
    SF_PTokGiveInd[macf, sq](info)
    []
    SF_TokPleaseInd[macf, sq](info)
    []
    sao!sacf_int_lock;SF_UnsolBidRej[sq](info)
    []
    SF_DefaultOutgoing[macf, sq](info)
    []
    SF_DefaultIncoming[macf, sq](info)
    []
    TerminateAssoc[sq](info)
    []
    [IsFREE(info)] ->
        i; ReturntoPool[macf](info)          (* SAO becomes attached to some other MACF *)
endproc
process SF_SafDetachReq[macf, sq, sao](info: sf_info): exit(sf_info) :=                                    (*10.4*)
    macf!get_did(info)?sp:ServicePrim[IsSAFDetachAssociationReq(sp)];
    sao!token?tok:bool;
    (   [ IsFREE(get_state(sp)) and IsQing(info)] ->                    (* both free and begin-fear have state FREE *)
            exit( set_did(did_absent, set_CFear(IsBeginFear(get_status(sp)),info)))
        []
        [ IsFREE(get_state(sp)) and not(IsQing(info))] ->                (* both free and begin-fear have state FREE *)
        (   [IsLoser(info) and tok] ->
                sq!AFTokenGiveReq(regular, dcc_absent);
                exit(change_state(FREE, set_did(did_absent, set_CFear(IsBeginFear(get_status(sp)),info))))
            []
            [not(IsLoser(info) and tok)] ->
                exit(change_state(FREE, set_did(did_absent, set_CFear(IsBeginFear(get_status(sp)),info))))
        )
        []
        [ ((IsCL_RbkCnf_EX(get_state(sp)) or IsCL_RbkInd_EX(get_state(sp))) and Not( IsQing(info))) or
        IsCL_BgnInd_EX(get_state(sp)) ] ->
            exit(change_state(get_state(sp), set_did(did_absent, info)))
        []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        [ (IsCL_RbkCnf_EX(get_state(sp)) or IsCL_RbkInd_EX(get_state(sp))) and IsQing(info) ] ->
            sq!discard;
            (   [IsLoser(info) and tok] ->
                    sq!AFTokenGiveReq(regular, dcc_absent);
                    exit(change_state(FREE, set_did(did_absent, info)))
                []
                [not(IsLoser(info) and tok)] ->
                    exit(change_state(FREE, set_did(did_absent, info)))
            )
    )
endproc
process  SF_AFBgnDgReq[macf, sq](info: sf_info): exit(sf_info) :=                                    (* 10.5.1 *)
    macf?did:dc_id?acn:application_context_name?qosO:quality_of_service_Opt?
    addr:address?rch:recovery_context_handle_Opt
    ?sp:ServicePrim[ IsAFBeginDialogueReq(sp) and IsFREE(info) ] ;
    exit(sp,did)
    []
    macf?did:dc_id?addr:address?sp:ServicePrim[IsAFBeginDialogueReq(sp) and IsFREE(info)];
    exit(sp,did)
>> accept sp:ServicePrim, did:dc_id in
    (   let  bm: bool = get_bm(info), ctr:bool = has_commit(sp) or has_recovery(sp) in
        choice bid: bool []                                                  (* Local Decision to bid *)
        [IsLoser(info) and (bm or IsCFear(info) or bid ) ] ->
            sq!AFBidReq(ctr, get_lbdrec(info));
            sq!Queue; exit( Qing(change_state(BIDDING,set_did( did,info))))
        []
        [IsWinner(info) and  IsCFear(info)] ->
            sq!Queue; exit( Qing(change_state(STRAY,set_did(did,info))))
        []
        [not((IsCFear(info) and IsWinner(info)) or (IsLoser(info) and (bm or
        IsCFear(info) or bid )))] ->
            exit(change_state(STRAY,info))
    >> accept info: sf_info in
        [not(IsWinner(info) or IsBIDDING(info))] ->
            sq!set_dccO(inc_dcc(get_lbdsent(info)),set_lpiO(get_lbdrec(info),sp));
(* issues AFBeginDialogue Req *)
            exit(inc_lbdsent(set_did(did, info)))
        []
        [IsWinner(info) or IsBIDDING(info)] ->                                (* Omit lpi *)
            sq!set_dccO(inc_dcc(get_lbdsent(info)),set_lpiO(dcc_absent,sp));
(* issues AFBeginDialogue Req *)
            exit(inc_lbdsent(set_did(did, info)))
    ) (* Queuer will queue sp if appropriate *)
endproc
process SF_AFBegDgInd[macf, sq, sao](info: sf_info): exit(sf_info) :=                               (* 10.5.2 *)
    sq?afsp:ServicePrim[IsAFBeginDialogueInd(afsp)];
    sao!token?tok:bool;
    (   choice reserved:bool []
        let cw:bool =  IsWinner(info), bm:bool= get_bm(info), valid:bool = get_lpiO(afsp) eq get_lbdsent(info) in
        let create:bool = (cw and (IsFREE(info) and valid and not(bm) and not(reserved)) or (IsBIDIndRec(info) and
        WasBidRspSent(info))) or (not(cw) and IsFREE(info)) in
        [cw and not(bm) and (IsSTRAY(info) or (IsFREE(info) and not(valid))) ] -> exit(info)
        []
        [not(cw) and (IsSTRAY(info) or IsBIDDING(info) )] ->
        (   [not(is_absent(get_did(info)))] ->
                macf!get_did(info)!SAFAssociationLostInd; exit(set_CFear( false, set_did(did_absent,info)))
            []
            [is_absent(get_did(info))] -> exit(info)
        >> accept info:sf_info in
            sq!discard; ReturntoPool[macf](info)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        )
(* The SAO attaches to some other TPPM; therefore the issuing of the AFBegDInd to this TPPM is not part of the actions of
   'our' TPPM, and are therefore not modelled. Instead, the internal event models the eventual return of this SAO to the pool.
   Note after the SAFAssocLost is issued, the SAO will no longer be part of the TPPM's action sequences *)
        []
        [create ] ->
        (   let keeptok:bool = cw and IsFREE(info) and not(bm) and not(reserved)
            and (has_recovery(afsp) or has_commit(afsp)) in
            [keeptok and tok] ->
                sq!AFTokenGiveReq( keep, get_dccO(afsp)); exit(info)
            []
            [keeptok and not(tok)] -> exit(set_skt(true,info)) (* Flag the KeepTok to be sent later *)
            []
            [not(keeptok)] -> exit(info)
            >> accept info: sf_info in
            macf?did:dc_id?addr:address?rch:recovery_context_handle!afsp; exit(did)
            []
            macf?did:dc_id?addr:address!afsp; exit(did)
            >> accept did:dc_id in exit(change_state(BUSY, change_lbdrec(get_dccO(afsp),set_did(did, info))))
        )
        []
        [cw and IsFREE(info) and valid and not(bm) and reserved] ->
            sq!AFBeginDialogueRsp( Opt(my_functional_units), rejected_provider, association_reserved, get_dccO(afsp),
            dataRI, ud_absent);
            exit(change_state(FREE,change_lbdrec(get_dccO(afsp),info)))
        []
        [not(create or IsFREE(info) or (IsSTRAY(info) and ((not(bm) and cw) or not(cw)))
        or (not(cw) and IsBIDDING(info)))] ->
(* Protocol error case *)
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_AFBgnDgRsp[macf, sq](info: sf_info): exit(sf_info) := (* 10.5.3 *)
    macf!get_did(info)?sp:ServicePrim[IsAFBeginDialogueRsp(sp)];
    sq!set_dccO(get_lbdrec(info), sp) ;exit(info)
endproc
process SF_AFBgnDgCnf[macf, sq](info: sf_info): exit(sf_info) := (* 10.5.4 *)
    sq?sp:ServicePrim[IsAFBeginDialogueCnf(sp)];
    (   [(IsFREE(info) or IsBIDDING(info)) and is_map_dataRI(sp)] -> exit(info)
        []
        [IsSTRAY(info) and is_map_dataRI(sp) and not(get_dccO(sp) eq get_lbdsent(info))] -> exit(info)
        []
        [(IsSTRAY(info) or IsBIDCnfRec(info)) and (get_dccO(sp) eq get_lbdsent(info))
        and not(is_reserved_association(sp))] ->
            macf!get_did(info)!sp; exit(change_state(BUSY,info) )
        []
        [IsLoser(info) and IsSTRAY(info) and (get_dccO(sp) eq get_lbdsent(info))
        and is_reserved_association(sp) and is_map_dataRI(sp)] ->
            macf!get_did(info)!sp; exit(change_state(BUSY,info) )
        []
        [not( ((IsFREE(info) or IsBIDDING(info)) and is_map_dataRI(sp))
        or (IsSTRAY(info) and is_map_dataRI(sp) and (not(get_dccO(sp) eq get_lbdsent(info))
        or ((get_dccO(sp) eq get_lbdsent(info)) and IsLoser(info) and is_reserved_association(sp))))
        or ( (IsSTRAY(info) or IsBIDCnfRec(info)) and (get_dccO(sp) eq get_lbdsent(info))
        and is_reserved_association(sp) ) ) ] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_AFBidInd[ macf,sq, sao](info: sf_info): exit(sf_info) :=                                    (* 10.5.5 *)
```


IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    sq!get_did(info)?sp:ServicePrim[IsAFBidInd(sp)];
    sao!token?tok:bool;
    (   let valid : bool = get_lpiO(sp) eq  get_lbdsent(info) in
        [IsWinner(info)] ->
        (   [ (IsFREE(info) and not(valid)) or IsSTRAY(info) or IsCL_RbkInd_EX(info) ] -> exit(info)
            []
            [ IsFREE(info) and valid] ->
                (   [get_ctr(sp) and not(tok)] ->
                        i; exit(NotTokReq(change_state(BIDIndRec,info)))
(* Local decision to accept, and to wait for the token *)
                []
                (   choice result: tp_bid_result []
(* Local decision to wait not taken *)
                    sq!AFBidRsp(result);
                    (   [is_rejected(result)] -> exit(info)
                        []
                        [is_accepted(result)] ->
                        (   [get_ctr(sp) and tok] -> sq!AFTokenGiveReq(regular, dcc_absent);
                                exit(TokReq(BidRspSent(change_state(BIDIndRec, info))))
                            []
                            [get_ctr(sp) and not(tok)] ->
                                exit(TokOwed(TokReq(BidRspSent(change_state(BIDIndRec,  info)))))
                            []
                            [not(get_ctr(sp))] ->
                                exit(NotTokReq(BidRspSent(change_state(BIDIndRec, info))))
                        )
                    )
                )
            )
        )
        []
        [IsLoser(info) or (IsWinner(info) and not(IsFREE(info) or IsSTRAY(info) or  IsCL_RbkInd_EX(info) ))] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_AFBidCnf[macf,sq](info: sf_info): exit(sf_info) :=                                   (* 10.5.6 *)
    sq?sp:ServicePrim[IsAFbidCnf(sp)];
    (   [IsBIDDING(info) ] ->
        (   [is_accepted(get_br(sp)) ] -> sq!flush_to_cbegin;
            (   [is_absent(get_did(info))] -> exit(change_state(FREE, info))
                []
                [not(is_absent(get_did(info)))] ->
                exit(change_state(BIDCnfRec, info))
            )
            []
            [is_rejected(get_br(sp)) ] ->
            (   [is_absent(get_did(info))] ->
                sq!discard;exit(NoQ( change_state(FREE, info)))
                []
                [not(is_absent(get_did(info)))] ->
                    macf!SAFAssociationLostInd; sq!discard;
                    exit(set_CFear( false, NoQ( change_state(FREE,set_did(did_absent,  info)))))
            )
        )
    )
    []
    [not(IsBIDDING(info))] -> SF_ProtocolError[macf, sq](info)
endproc
process SF_AbortInd[macf,sq](info:sf_info): exit(sf_info) :=                                    (* 10.5.14 *)
    sq?afsp:ServicePrim[IsAFAbortInd(afsp) and is_map_abortRI(afsp) and is_provider(afsp)] ;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    (   [IsFREE(info) or IsBIDIndRec( info) or IsCL_RbkInd_EX(info) or IsCL_BgnInd_EX(info) ] -> exit(info)
        []
        [IsBIDDING(info) or IsSTRAY(info) or IsBIDCnfRec( info) or IsBUSY(info)] ->
        (   [not(is_absent(get_did(info)))] -> macf!get_did(info)!afsp; sq!discard;exit(info)
            []
            [is_absent(get_did(info))] ->sq!discard;exit(info)
        )
        []
        [not(IsFREE(info) or IsBIDIndRec( info) or IsBIDCnfRec( info) or IsCL_RbkInd_EX(info)
        or IsCL_BgnInd_EX(info) or IsBIDDING(info) or  IsSTRAY(info) or IsBUSY(info))] ->
            SF_ProtocolError[macf, sq](info)
    )
    []
    sq?afsp:ServicePrim[IsAFAbortInd(afsp) and (is_map_commitRI(afsp) or   is_map_commitRC(afsp)) and is_user(afsp)] ;
                                                                        (* 10.5.38 *)
    (   [IsBUSY(info)] ->  macf!get_did(info)!afsp; exit(info)
        []
        [not(IsBUSY(info))] -> SF_ProtocolError[macf, sq](info)
    )
    []                                                                  (* 10.5.15 *)
    sq?afsp:ServicePrim[IsAFAbortInd(afsp) and is_map_dataRI(afsp)] ;
    (   [IsFREE(info) or IsBIDDING(info) or IsSTRAY(info) or
        IsCL_RbkInd_EX(info) ] -> exit(info)
        []
        [IsCL_BgnInd_EX(info)] -> exit(change_state(FREE, info))
        []
        [IsBUSY(info)] -> macf!get_did(info)!afsp; exit(info)
        []
        [ not(IsFREE(info) or IsBIDDING(info) or IsSTRAY(info) or IsCL_RbkInd_EX(info)
    or IsCL_BgnInd_EX(info) or IsBUSY(info))] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_AAbortInd[macf, sq](info: sf_info): exit(sf_info):=                          (* 10.5.17 *)
    sq?sp:ServicePrim[IsAAbortInd(sp) or IsAPAbortInd(sp) or IsAReleaseCnf(sp)];
    (   [IsBIDDING(info) or IsSTRAY(info) or IsBIDCnfRec(info) or IsBUSY(info)] ->
        (   [not(is_absent(get_did(info)))] -> macf!get_did(info)!sp; exit(info)
            []
            [is_absent(get_did(info))] -> exit(info)
        )
        []
        [not(IsBIDDING(info) or IsSTRAY(info) or IsBIDCnfRec(info) or IsBUSY(info))] -> exit(info)
    )
endproc
process SF_CBgnReq[macf, sq, sao](info: sf_info): exit(sf_info):=                       (* 10.5.33 *)
    macf!get_did(info)?sp:ServicePrim[IsCBeginReq(sp)] ;
    sao!token?tok:bool;
    (   [not(tok) and (IsSTRAY(info) or IsBIDCnfRec(info))] ->   sq!queue; sq!sp ; exit(Qing(info))
        []
        [not(tok) and IsBIDDING(info)] -> sq!sp ; exit(info)
        []
        [tok] -> sq!sp; exit(info)
    )
endproc
process  SF_CBgnInd[macf, sq, sao](info: sf_info): exit(sf_info) :=                     (* 10.5.34 *)
    sq?sp:ServicePrim[IsCBeginInd(sp)];
    sao!token?tok:bool;
    (   [(IsFREE(info) and IsCFear(info)) or IsCL_BgnInd_EX(info)] ->
            sq!CRollbackReq; exit(change_state(CL_RollbackCnf_EX,info))
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        []
        [IsBUSY(info)] -> macf!get_did(info)!sp; exit(info)
        []
        [(IsBIDDING(info) or IsSTRAY(info)) and IsCFear(info) ] ->
        (   [not(is_absent(get_did(info)))] ->
                macf!get_did(info)!SAFAssociationLostInd; exit(set_CFear( false, set_did(did_absent, info)))
            []
            [is_absent(get_did(info))] -> exit(info)
        >> accept info:sf_info in
            sq!discard; sq!CRollbackReq; exit(NoQ( change_state(CL_RollbackCnf_EX,set_did(did_absent,info))))
        )
        []
        [not(IsCL_BgnInd_EX(info) or IsBUSY(info) or  ((IsFREE(info)
        or IsBIDDING(info) or IsSTRAY(info))  and IsCFear(info) ))] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_CRbkInd[macf, sq, sao](info: sf_info): exit(sf_info) :=                                    (* 10.5.42 *)
    sq?sp:ServicePrim [IsCRollbackInd(sp)];
    sao!token?tok:bool;
    (   [IsBUSY(info) or IsSTRAY(info) or IsBIDCnfRec(info) ] ->
            macf!get_did(info)!sp; exit(change_state(BUSY,info))
        []
        [IsCL_RbkCnf_EX(info)] ->
            sq!REM?last_SP_sent:ServicePrim ?last_sp: ServicePrim;
(*  Interrogate the queuer as to the  last AF service prim issued *)
            (   let abortORbegD :bool = (IsAFAbortReq(last_SP_sent) and  is_map_rollbackRI(last_SP_sent)
                and is_Provider(last_SP_sent) and is_begin_trans_rej(get_pad(last_SP_sent)) )
                or  (IsAFBeginDialogueRsp(last_SP_sent) and is_rejected(get_bdr(last_SP_sent))
                and is_map_rollbackRI(sp) and is_User(last_SP_sent)) in
(*  Assuming parameters in the BegD case are to be set to the values on the first response *)
                [abortORbegD] ->
                    sq!set_map(rollbackRC, last_SP_sent); exit(change_state(FREE, info))
                []
                [not(abortORbegD)] ->  sq!CRollbackRsp; exit(change_state(FREE,info))
            >> accept info:sf_info in
                [IsLoser(info) and tok] -> sq!AFTokenGiveReq(regular, dcc_absent); exit(info)
                []
                [not(IsLoser(info) and tok)] -> exit(info)
            )
        []
        [IsCL_RbkInd_EX(info)] ->
            sq!CRollbackRsp;
            (   [IsLoser(info) and tok] -> sq!AFTokenGiveReq(regular, dcc_absent);  exit(change_state(FREE,info))
                []
                [not(IsLoser(info) and tok)] -> exit(change_state(FREE,info))
            )
        []
        [not(IsBUSY(info) or  IsSTRAY(info) or IsBIDCnfRec(info) or
        IsCL_RbkCnf_EX(info) or IsCL_RbkInd_EX(info))] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_AFAbrtInd_Rbk[macf, sq, sao](info: sf_info): exit(sf_info) :=                              (* 10.5.43 *)
    sq?sp:ServicePrim[(IsAFAbortInd(sp) or IsAFAbortAndHeuristicReportInd(sp) or IsAFHeuristicReportInd(sp) )
    and is_map_rollbackRI(sp)];
    sao!token?tok:bool;
    (   [IsBUSY(info) or IsSTRAY(info) or IsBIDCnfRec(info)] ->
            macf!get_did(info)!sp; exit(change_state( BUSY, info))
        []
```

IECNORM.COM: Click to view the full PDF of ISO/IEC 10026-3:1992

```
        [IsCL_RbkCnf_EX(info) and IsAFAbortInd(sp)] ->
            sq!CRollbackRsp;
            (   [ IsLoser( info) and tok ] -> sq !AFTokenGiveReq( regular, dcc_absent); exit(change_state(FREE, info))
                []
                [ not( IsLoser( info) and tok) ] -> exit(change_state(FREE, info))
            )
        []
        [not( IsBUSY(info) or IsSTRAY(info) or IsBIDCnfRec(info) or  (IsCL_RbkCnf_EX(info) and IsAFAbortInd(sp))) ] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_CRbkCnf[macf, sq, sao](info: sf_info): exit(sf_info) :=                                  (* 10.5.44 *)
    sq?sp:ServicePrim[IsCRollbackCnf(sp) or (  is_map_rollbackRC(sp) and
    (IsAFHeuristicReportInd(sp) or IsAFAbortInd(sp) or
    IsAFAbortAndHeuristicReportInd(sp))  )];
    sao!token?tok:bool;
    (   [ IsCL_RbkCnf_EX(info)] ->
        (   [IsLoser(info) and tok] -> sq!AFTokenGiveReq(regular, dcc_absent); exit(change_state(FREE,info))
            []
            [not(IsLoser(info) and tok)] -> exit(change_state(FREE,info))
        )
        []
        [IsBUSY(info) or IsSTRAY(info) or IsBIDCnfREC(info)] ->
            macf!get_did(info)!sp; exit(change_state(BUSY, info))
        []
        [not(IsCL_RbkCnf_EX(info) or  IsBUSY(info) or IsSTRAY(info) or
        IsBIDCnfREC(info))] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_RecoverReq[macf, sq, sao](info:sf_info): exit(sf_info):=                                 (* 10.5.46 *)
    sq?sp:ServicePrim[IsCRecoverReq(sp) or IsAFRecoverReq(sp)];
    sao!token?tok:bool;
    sq !REM ?last_SP_sent: ServicePrim ?last_sp: ServicePrim;  (*  interogate as to last service primitive at the SACF  *)
    (   [IsBUSY(info) and (tok or (IsCRecoverReq( sp) and is_commit( get_rsri( sp)) and (IsCRecoverInd( last_sp)
        or IsAFRecoverInd( last_sp)) and is_ready( get_rsri( last_sp))))] -> macf!get_did(info)!sp; exit(info)
        []
        [IsBUSY(info) and not(tok or (IsCRecoverReq( sp) and is_commit( get_rsri( sp)) and (IsCRecoverInd( last_sp)
        or IsAFRecoverInd( last_sp)) and is_ready( get_rsri( last_sp))))] -> sq!queue; sq!sp; exit(Qing(info))
    )
endproc
process SF_AFTokGiveInd[macf, sq](info: sf_info): exit(sf_info) :=                                  (* 10.5.51 *)
    sq?sp:ServicePrim[IsAFTokenGiveInd(sp) and is_regular(get_reason(sp))];
    (   [IsLoser(info) and  (IsFREE(info) or IsSTRAY(info) or IsBIDDING(info)  )] ->
            sq!AFTokenGiveReq(regular, dcc_absent); exit(info)
        []
        [(IsWinner(info) and IsSTRAY(info)) or IsBIDCnfRec(info)] ->
            sq!flush;
            (   [is_absent(get_did(info)) and IsSTRAY(info)] -> exit(NoQ(change_state(FREE,info)))
                []
                [not(is_absent(get_did(info)) and IsSTRAY(info))] -> exit(NoQ(info))
            )
        []
        [ IsWinner(info) and (IsFREE(info) or (IsBIDIndRec(info)  and  not(WasTokReq(info))))] -> exit(info )
        []
        [ IsWinner(info) and IsBIDIndRec(info) and WasTokReq(info)] ->
        (   [not(WasBidRspSent(info)) ] ->
                sq!AFBidRsp(bid_accepted); sq!AFTokenGiveReq(regular, dcc_absent);
                exit(BidRspSent(info))
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
            []
            [WasBidRspSent(info) and IsTokOwed(info)] ->
                sq!AFTokenGiveReq(regular, dcc_absent); exit(NotTokOwed(info))
            []
            [WasBidRspSent(info) and not(IsTokOwed(info))] -> exit(info)
        )
        []
        [IsBUSY(info) and SendKeepTok(info)] ->
            sq!AFTokenGiveReq( keep, get_lbdrec(info)); exit(set_skt(false,info))
        []
        [not(SendKeepTok(info)) and IsBUSY(info) ] ->
            sq !pass_to_UASE !sp; exit(info)            (* Routed  to UASE *)
        []
        [IsCL_RbkInd_EX(info) or IsCL_BgnInd_EX(info) or  IsCL_RbkCnf_EX(info)] ->
            SF_ProtocolError[macf, sq](info)
    )
    []                                                                  (* 10.5.52 *)
    sq?sp:ServicePrim[IsAFTokenGiveInd(sp) and is_keep(get_reason(sp))];
    (   let valid:bool =  get_dccO(sp) eq get_lbdsent(info) in
        [IsLoser(info) and (IsSTRAY(info) or IsBUSY(info)) and valid] ->
            sq!flush; sq !pass_to_UASE!sp; exit(NoQ(info))    (* Routed  to UASE *)
        []
        [IsLoser(info) and (IsFREE(info) or ((IsSTRAY(info) or IsBIDDING(info)) and not(valid)))] ->
            sq!AFTokenGiveReq(regular, dcc_absent); exit(info)
        []
        [IsWinner(info) or (IsLoser(info) and not(IsSTRAY(info) or
        (IsFREE(info) and not(valid)) or (IsBUSY(info) and valid)))] ->
            SF_ProtocolError[macf, sq](info)
    )
    []                                                                  (* 10.5.54 *)
    sq?sp:ServicePrim[IsAFTokenGiveInd(sp) and is_two_way_recovery(get_reason(sp))];
    (   [is_chan(get_did(info)) and IsBUSY(info)] ->
        (   [ not( IsQing( info)) ] -> macf!get_did(info)!sp; exit(info)
            []
            [ IsQing( info) ] -> sq !flush; exit( NoQ( info))
        )
        []
        [ not(is_chan(get_did(info)) and IsBUSY(info))] -> SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_AFTokGiveReq[macf, sq, sao](info:sf_info): exit(sf_info) :=                  (* 10.5.53 *)
    macf!get_did(info)?sp:ServicePrim[IsAFTokenGiveReq(sp) and is_two_way_recovery(get_reason(sp))];
    sao!token?tok:bool;
    (   [IsBUSY(info) and tok] -> sq!sp; exit(info)
        []
        [IsBUSY(info) and not(tok)] -> sq!queue; sq!sp; exit(Qing(info))
        []
        [not(IsBUSY(info))] ->  SF_ProtocolError[macf, sq](info)
    )
endproc
process SF_PTokGiveInd[macf, sq](info: sf_info): exit(sf_info) := (* 10.5.55 *)
    sq?sp:ServicePrim[IsPTokenGiveInd(sp)];
    (   [IsLoser(info) and (IsFREE(info) or IsSTRAY(info) or IsBIDDING(info)) ] ->
            sq!AFTokenGiveReq(regular, dcc_absent); exit(info)
        []
        [IsBUSY(info) and is_dial(get_did(info))] ->
            sq !pass_to_UASE !sp; exit(info)            (* routed to UASE *)
        []
        [IsWinner(info) and IsSTRAY(info) ] ->
            sq!flush;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
        (   [is_absent(get_did(info))] -> exit(NoQ(change_state(FREE,info)))
            []
            [not(is_absent(get_did(info)))] -> exit(NoQ(info))
        )
    []
    [IsWinner(info) and IsFREE(info)] -> exit(info)
    []
    [not((IsBUSY(info) and is_dial(get_did(info))) or IsSTRAY(info) or IsFREE(info)
    or (IsLoser(info) and IsBIDDING(info)))] ->
        SF_ProtocolError[macf, sq](info)
)
endproc
process SF_TokPleaseInd[macf, sq](info: sf_info): exit(sf_info) :=                    (* 10.5.58 *)
    sq?sp:ServicePrim[IsPTokenPleaseInd(sp)];
    (   [IsBUSY(info) and is_dial(get_did(info))] -> sq !pass_to_UASE !sp; exit(info) (* Will be passed to UASE*)
        []
        [not(IsBUSY(info) and is_dial(get_did(info)))] -> exit(info)
    )
endproc
process SF_ProtocolError[macf, sq](info:sf_info) :exit(sf_info) :=                    (* 10.5.59 *)
    [ IsBUSY(info) or IsBIDDING(info) or IsSTRAY(info) or IsBIDCnfRec(info)] ->
        macf!get_did(info)!sacf_PE; exit(info)
    []
    [not(IsBUSY(info) or IsBIDDING(info) or IsSTRAY(info) or IsBIDCnfRec(info))] ->
        sq!AFAbortReq(provider, abortRI, protocol_error); exit(info)
(* text has A-ABORT instead of AF-ABORT at this point  *)
endproc
(* This covers all the remaining  procedures in 10.5. For the remaining request/response sps not in any of the proceeding
   processes, the default is to pass through or queue. *)
process SF_DefaultOutgoing[macf, sq](info:sf_info) : exit(sf_info) :=
    [ IsBUSY( info) ] ->
        macf!get_did(info)?sp:ServicePrim
        [not(IsSAFDetachAssociationReq(sp) or IsAFBeginDialogueReq(sp) or IsAFBeginDialogueRsp(sp) or
        IsCBeginReq(sp) or IsCRecoverReq(sp) or IsAFRecoverReq(sp) or IsAFTokenGiveReq(sp))] ;
        sq!sp ;
        exit(info)
endproc
process SF_UnsolBidRej[sq](info:sf_info) :exit(sf_info) :=                            (* 10.6.1 *)
    [ IsWinner(info) and IsBIDIndRec(info) and not(WasBidRspSent(info))] ->
        i; sq!AFBidRsp(bid_rejected) ; exit(change_state(FREE, info))
    []
    exit(info)
endproc
(* For incoming events  not in any of the preceding processes, if in the BUSY state the sp is always passed through. Most are also
   accepted and discarded whilst in the FREE, BIDDING or STRAY states. *)
process SF_DefaultIncoming[macf, sq](info:sf_info) :exit(sf_info) :=
    sq!get_did(info)?sp:ServicePrim[IsAFEndDialogueInd(sp) or  IsAFEndDialogueCnf(sp)  or IsAFUErrorInd(sp)
    or IsAFUerrorCnf(sp) or  IsAFGrantControlInd(sp) or IsAFRequestControlInd(sp) or  IsAFHandshakeInd(sp)
    or IsAFHandshakeCnf( sp)  or  IsAFHandshakeAndGrantControlInd( sp) or IsAFHandshakeAndGrantControlCnf(  sp)
    or IsAFTokenPleaseInd(sp) or IsUASEInd(sp) ] ;
    (   [IsBUSY(info)] -> sq!sp ; exit(info)
        []
        [IsFREE(info) or IsSTRAY(info) or IsBIDDING(info) ] -> exit(info)
        []
        [IsCL_RbkInd_EX(info) and IsAFUErrorInd(sp)] -> exit(info)
        []
        [not(IsFREE(info) or IsSTRAY(info) or IsBIDDING(info) or  (IsCL_RbkInd_EX(info) and IsAFUErrorInd(sp)))] ->
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
            SF_ProtocolError[macf, sq](info)
    )
    []
    sq!get_did(info)?sp:ServicePrim[ IsAFDeferInd( sp) or IsAFPrepareInd( sp)
    or IsCBeginCnf(sp) or IsCReadyInd(sp) or  IsCCommit_CBeginInd( sp) or
    IsAFRecoverInd( sp)  or IsCCommitCnf( sp) or IsCRecoverCnf(sp) or
    IsCRecoverInd(sp) or IsCCommitInd(sp) or  (IsAFAbortAndHeuristicReportInd(
    sp) and is_map_commitRC(sp) ) or   (IsAFHeuristicReportInd(sp) and (
    is_map_recover_doneRC(sp) or  is_map_commitRC(sp)) )];
    (   [IsBUSY(info)] -> macf!get_did(info)!sp; exit(info)
        []
        [IsCL_RbkCnf_EX(info) and IsAFPrepareInd( sp)] -> exit(info)
        []
        [not(IsBUSY(info) or(IsCL_RbkCnf_EX(info) and IsAFPrepareInd( sp)))] ->
            SF_ProtocolError[macf, sq](info)
    )
endproc
(* Whilst in the FREE state, the Association manager may choose to release . *)
process TerminateAssoc[sq](info:sf_info):exit(sf_info):=
    [IsFREE(info)] ->
        i ;                                          (* SAO to be terminated *)
        sq!AReleaseReq;
        sq!AReleaseCnf(affirmative);
        exit(info)
endproc
(* Also whilst in the FREE state, the SAO may be attached to a different MACF. Subsequent to this it may return to the pool with
    some parameters changed:*)
process ReturntoPool[macf](info:sf_info):exit(sf_info):=
    i; (* Becomes available again*)
    (   choice lbdrec,lbdsent:correlator_Opt, CFear : bool []
        [(not(is_absent(get_lbdrec(info))) implies (nat(get_lbdrec(info)) lt  nat(lbdrec))
          and (nat(get_lbdsent(info)) lt nat(lbdsent)))] ->                    (* return to pool with new param values *)
            exit( change_lbdsent(lbdsent, change_lbdrec(lbdrec, set_CFear( CFear, change_state(FREE, info)))))
    )
endproc
process SACFConcatenator[ cd, p, aei, sao]: exit:=
    cd !dn ?apdu: PDU [ IsAARQ( apdu) ];
    p ?pad: p_address !presentation_embedding( apdu);
    concat_separator[  cd, p, aei,sao]( pad)
    []
    p ?pad: p_address ?psp: ServicePrim
    [ IsPWithEmbeddedAPDUIndCnf( psp) and has_embedded_AARQ( psp) ];
    sao !lock; cd !up !get_PDU( psp); concat_separator[  cd, p, aei, sao]( pad)
endproc
process concat_separator[  cd, p, aei, sao]( pad: p_address): exit :=
    (   hide hs in concatenate[  cd, p, hs]( pad, emptyPDU, cnfu_absent) ||| separator[  cd, p, aei, sao, hs]( pad) )
    |[p]|
    (   ( Outputs[p]( pad)  ||| PSAPQueuer[ p, aei](pad) ) [> exit )
endproc
process PSAPQueuer[ p, aei]( pad: p_address): noexit:=
    p !pad ?psp: ServicePrim; PSAPQueuer[ p,aei](pad)
    []
    aei ?did:dc_id !open_psap;  PSAPQueuer[ p,aei](pad)
    []
    aei?did:dc_id !close_psap; aei ?did:dc_id !open_psap;
    PSAPQueuer[ p,aei](pad)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
endproc
process Outputs[p](pad: P_address):noexit:=
    p!pad?sp:ServicePrim[IsReqRsp(sp)]; Outputs[p](pad)
endproc
process concatenate[ cd, p, hs] ( pad: p_address, q: PDUqueue, cnfuO: confirmation_urgency_Opt) : exit :=
    [ not( IsEmpty( q)) ] ->
        i; p !pad !presentation_embedding( q);
        concatenate[ cd, p, hs]( pad, emptyPDU, cnfuO)
    []
    cd !dn ?psp: ServicePrim;
    p !pad !presentation_embedding( q);
    p !pad !psp;
    concatenate[ cd, p, hs]( pad, emptyPDU, cnfuO)
    []
    cd !dn ?apdu: PDU;
(* test whether a pdu can be concatenated to a queue, using allowed_concat which is defined in the datatyping to match the
    allowed concatenation sequences in subclause 10.7 *)
    (   [ IsEmpty( q) ] -> concat_pdu[ cd, p, hs]( pad, apdu, cnfuO)
        []
        [ not( IsEmpty( q)) and allowed_concat( q + apdu) ] ->
            concat_queue[ cd, p, hs]( pad, emptyPDU + apdu, apdu, cnfuO)
        []
        [ not( IsEmpty( q)) and not( allowed_concat( q + apdu)) ] ->
            p !pad !presentation_embedding( q);
            concat_pdu[ cd, p, hs]( pad, apdu, cnfuO)
(* We send the current queue (embedded in the appropriate presentation service primitive) and then try to start a new queue of
    PDUs by with the PDU on its own. *)
    )
    []
    cd !dn ?pdu: PDU ?pdu1: PDU [ Isc_commit_ri( pdu) and Isc_begin_ri( pdu1) ];
    (   [ IsEmpty( q) ] ->
            concat_queue[ cd, p, hs]( pad, emptyPDU + pdu + pdu1, pdu1, cnfuO)
        []
        [ not( IsEmpty( q)) and allowed_concat( q + pdu + pdu1) ] ->
            concat_queue[ cd, p, hs]( pad, q + pdu + pdu1, pdu1, cnfuO)
        []
        [ not( IsEmpty( q)) and not( allowed_concat( q + pdu + pdu1)) ] ->
            p !pad !presentation_embedding( q);
            concat_queue[ cd, p, hs]( pad, emptyPDU + pdu + pdu1, pdu1, cnfuO)
    )
    []
    hs ?cnfuO: confirmation_urgency_Opt;
    concatenate[ cd, p, hs]( pad, q, cnfuO)
    []
    exit
endproc
process concat_pdu[ cd, p, hs] ( pad: p_address, apdu: PDU, cnfuO: confirmation_urgency_Opt): exit :=
    [ can_concat( apdu) ] -> concatenate[ cd, p, hs]( pad, queue( apdu), cnfuO)
    []
    [ not( can_concat( apdu)) ] ->
        p !pad !presentation_embedding( apdu);
        concatenate[ cd, p, hs]( pad, emptyPDU,  cnfuO)
endproc
process concat_queue[ cd, p, hs]( pad: p_address, q: PDUqueue,
last_apdu: PDU, cnfuO: confirmation_urgency_Opt): exit :=
    [ end_concat( last_apdu, cnfuO) ] ->
        p !pad !presentation_embedding( q);
        concatenate[ cd, p, hs]( pad, emptyPDU, cnfuO)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
(* If the pdu always ends a concatenation sequence then we send the resulting queue of PDUs in the appropriate presentation
   service primitive, otherwise we continue with the new queue. *)
    []
    [ not( end_concat( last_apdu, cnfuO)) ] ->
        concatenate[  cd, p, hs]( pad, q, cnfuO)
endproc
process separator[  cd, p, aei, sao, hs ]( pad: p_address): exit :=
    p !pad ?psp: ServicePrim;
    (   [ has_embedded_PDUs( psp) ] ->
        (   separate[ cd, aei, sao, hs]( get_PDUqueue( psp)) >>
            separator[  cd, p, sao, aei, hs]( pad))
        []
        [ not( has_embedded_PDUs( psp)) ] ->
        (   aei !lock ?did:dc_id; exit [] sao!lock;exit >>
            cd !up !psp; separator[  cd, p, sao, aei, hs]( pad) )
    )
    []
    exit
endproc
process separate[ cd, aei, sao, hs]( q: PDUqueue): exit :=
    [ IsEmpty( q) ] -> exit
    []
    [ not( IsEmpty( q)) ] ->
    (   let pdu: PDU = head( q), q: PDUqueue = tail( q) in
        [ Isc_commit_ri( pdu) and not( IsEmpty( q)) and Isc_begin_ri( head( q)) ] ->
        (   aei !lock ?did:dc_id; exit [] sao!lock;exit >>
            cd !up !pdu !head(q); separate[ cd, aei, sao, hs]( tail( q)) )
        []
        [ not( Isc_commit_ri( pdu) and not( IsEmpty( q)) and Isc_begin_ri( head( q)) ) ] ->
        (   aei !lock ?did:dc_id; exit [] sao!lock;exit >> cd !up !pdu;
(*  The concatenator-separator operates asynchronously from the action sequences, so there is no pressure to issue PDUs from
    the separator. In particular, a C-BEGIN-RI may be held in the separator until the current transaction has completed, if the
    implementation chooses to accept the CBegin indication then, as described in the note in 11.3.37*)
            (   [ is_handshake_ri( pdu) ] ->
                hs !get_cnfuO( pdu); separate[ cd, aei, sao, hs]( q)
                []
                [ not( is_handshake_ri( pdu)) ] ->
                    separate[ cd, aei, sao, hs]( q)
            )
        )
    )
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

H.2.5 Other ASEs

```
process UASE[sacf,tp,slk]: noexit :=
    sacf ?sp: ServicePrim [ IsUASEsp( sp) ]; UASE[sacf,tp,slk]
    []
    tp ?io: input_output ?pdu: PDU [ Isupdu( pdu) ]; UASE[sacf,tp,slk]
    []
    tp ?io: input_output ?psp: ServicePrim [ IsPsp( psp) ]; UASE[sacf,tp,slk]
    []
    slk ?did: dc_id; UASE[sacf,tp,slk]
endproc
process ACSE[ccr,cd,slk]: noexit :=
    ccr ?sp: ServicePrim [ IsACSEsp( sp) ]; ACSE[ccr,cd,slk]
    []
    cd ?io: input_output ?pdu: PDU [ IsACSEpdu( pdu) ]; ACSE[ccr,cd,slk]
    []
    cd ?io: input_output ?psp: ServicePrim [ IsPsp( psp) ]; ACSE[ccr,cd,slk]
    []
    slk ?did: dc_id; ACSE[ccr,cd,slk]
endproc
process CCR[ccr,cd,slk]: noexit :=
    ccr ?sp: ServicePrim [ IsCCRsp( sp) ]; CCR[ccr,cd,slk]
    []
    cd ?io: input_output ?pdu: PDU [ IsCCRpdu( pdu) ]; CCR[ccr,cd,slk]
    []
    slk ?did: dc_id; CCR[ccr,cd,slk]
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

H.3 The PM Procedures

```
process TPPMProcedures[tppm, caf, macf, aei, tlk] ( Assocs:set_of_assocs,tppm:tppm_info): exit(set_of_assocs,tppm_info):=
    TPPM_Procedures[tppm, caf, macf, aei](Assocs,tppm)
    []
    tlk!lock; MACFinitiated_events[ tppm, caf, macf, aei]( Assocs,tppm)
endproc
process CPMProcedures[cpm, macf, aei, clk] ( Assocs:set_of_assocs): exit(set_of_assocs):=
    CPM_Procedures[cpm, macf, aei](Assocs)
    []
    clk!lock; CMACFinitiated_events[ cpm, macf, aei]( Assocs)
endproc
```

H.3.1 TPPM Procedures Processes

```
process TPPM_Procedures[tppm, caf, macf, aei]
(Assocs:set_of_assocs,tppm:tppm_info): exit(set_of_assocs,tppm_info):=
    TPBeginDialogueReq[tppm,caf,macf](Assocs,tppm)
    []
    AFBeginDialogueInd[tppm,macf,aei](Assocs,tppm)
    []
    TPBeginDialogueRsp[tppm,macf](Assocs,tppm)
    []
    AFBeginDialogueCnf_accepted[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFBeginDialogueCnf_dataRI[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFBeginDialogueCnf_rollbackRI[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFBeginDialogueCnf_rollbackRC[tppm,caf,macf,aei](Assocs,tppm)
    []
    SAFAssociationLostInd[tppm,caf,macf,aei](Assocs,tppm)
    []
    TPEndDialogueReq[tppm,macf](Assocs,tppm)
    []
    AFEndDialogueInd[tppm,caf,macf,aei](Assocs,tppm)
    []
    TPEndDialogueRsp[tppm,macf](Assocs,tppm)
    []
    AFEndDialogueCnf[tppm,caf,macf,aei](Assocs,tppm)
    []
    TPUErrorReq[tppm,macf](Assocs,tppm)
    []
    AFUErrorInd[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFUErrorCnf[tppm,caf,macf,aei](Assocs,tppm)
    []
    TPUAbortReq[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFAbortInd_DataRI[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFAbortInd_U_CmmtRI[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFAbortInd_U_CmmtRC[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFAbortInd_RbkRI[tppm,caf,macf,aei](Assocs,tppm)
    []
    AFAbortInd_RbkRC[tppm,caf,macf,aei](Assocs,tppm)
    []
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

AFAbortInd_DataRI[tppm,caf,macf,aei](Assocs,tppm)
[]
AFAbortInd_U_CmmtRI[tppm,caf,macf,aei](Assocs,tppm)
[]
AFAbortInd_U_CmmtRC[tppm,caf,macf,aei](Assocs,tppm)
[]
AFAbortInd_RbkRI[tppm,caf,macf,aei](Assocs,tppm)
[]
AFAbortInd_RbkRC[tppm,caf,macf,aei](Assocs,tppm)
[]
APAbortInd_Dialogue[tppm,caf,macf,aei](Assocs,tppm)
[]
APAbortInd_Channel[caf,macf](Assocs,tppm)
[]
TPGrantCtrlReq[tppm,macf](Assocs,tppm)
[]
AFGrantCtrlInd[tppm,caf,macf,aei](Assocs,tppm)
[]
TPRequestCtrlReq[tppm,macf](Assocs,tppm)
[]
AFRequestCtrlInd[tppm,macf](Assocs,tppm)
[]
TPHandshakeReq[tppm,macf](Assocs,tppm)
[]
AFHandshakeInd[tppm,caf,macf](Assocs,tppm)
[]
TPHandshakeRsp[tppm,macf](Assocs,tppm)
[]
AFHandshakeCnf[tppm,caf,macf,aei](Assocs,tppm)
[]
TPHandshakeAndGrantCtrlReq[tppm,macf](Assocs,tppm)
[]
AFHandshakeAndGrantCtrlInd[tppm,macf,caf,aei](Assocs,tppm)
[]
TPHandshakeAndGrantCtrlRsp[tppm,macf](Assocs,tppm)
[]
AFHandshakeAndGrantCtrlCnf[tppm,caf,macf,aei](Assocs,tppm)
[]
TPBeginTransReq[tppm,caf,macf](Assocs,tppm)
[]
CBeginInd[tppm,caf,macf,aei](Assocs,tppm)
[]
CBeginCnf[macf](Assocs,tppm)
[]
TPDataReq[tppm,macf](Assocs,tppm)
[]
UASEInd[tppm,caf,macf,aei](Assocs,tppm)
[]
TPDfrredEndDialogueReq[tppm,macf](Assocs,tppm)
[]
TPDfrredGrantControlReq[tppm,macf](Assocs,tppm)
[]
AFDfrInd[tppm,caf,macf,aei](Assocs,tppm)
[]
TPPrepReq[tppm,macf](Assocs,tppm)
[]
TPCmtReq[tppm,caf,macf,aei](Assocs,tppm)
[]
AFPrepInd[tppm,caf,macf,aei](Assocs,tppm)
[]
CReadyInd[tppm,caf,macf,aei](Assocs,tppm)

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    []
    CCommitInd[ tppm,caf,macf,aei](Assocs,tppm)
    []
    TPDoneReq[tppm,macf,caf,aei](Assocs,tppm)
    []
    CCommitCnf[tppm,caf,macf,aei](Assocs,tppm)
    []
    TPRollbackReq[tppm,caf,macf](Assocs,tppm)
    []
    CRollbackInd[tppm,caf,macf,aei](Assocs,tppm)
    []
    CRollbackCnf[tppm,caf,macf,aei](Assocs,tppm)
    []
    CAFRecoverReadyInd[tppm,caf,macf](Assocs,tppm)
    []
    CAFRecoverCommitInd[tppm,caf,macf,aei](Assocs,tppm)
    []
    CRecoverCommitInd[tppm,caf,macf,aei](Assocs,tppm)
    []
    CRecoverDoneCnf[tppm,caf,macf,aei](Assocs,tppm)
    []
    CRecoverUnknownCnf[tppm,caf,macf](Assocs,tppm)
    []
    CRecoverRetryCnf[caf,macf](Assocs,tppm)
    []
    AFTokenGiveInd[caf,macf](Assocs,tppm)
    []
    AFTokenPleaseInd[caf,macf](Assocs,tppm)
    []
    CAFGiveInd[caf,macf](Assocs,tppm)
    []
    CAFFailInd[caf](Assocs,tppm)
    []
    TPPMErrors[tppm,caf,macf,aei](Assocs,tppm)
endproc
process CPM_Procedures[cpm, macf, aei]
(Assocs:set_of_assocs): exit(set_of_assocs) :=
    AFBeginDialogueInd_CPM[macf,aei](Assocs)
    []
    AFBeginDialogueCnf_CPM[cpm,macf](Assocs)
    []
    SAFAssociationLostInd_CPM[cpm,macf](Assocs)
    []
    AFEndDialogueInd_CPM[macf](Assocs)
    []
    CRecoverReadyInd_CPM[cpm,macf](Assocs)
    []
    CRecoverCommitInd_CPM[ cpm,macf,aei](Assocs)
    []
    AFTokenGiveInd_CPM[cpm,macf](Assocs)
    []
    AFTokenPleaseInd_CPM[cpm,macf](Assocs)
    []
    CAFDetachReq[cpm](Assocs)
    []
    CAFPleaseReq[cpm,macf,aei](Assocs)
    []
    APAbortInd_CPM[cpm,macf](Assocs)
    []
    CPMErrors[aei](Assocs)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
endproc
process MACFinitiated_events[ tppm, caf, macf, aei]
(Assocs:set_of_assocs,tppm:tppm_info): exit(set_of_assocs,tppm_info):=
    DelayRecovery[caf,macf](Assocs,tppm)
    []
    HeuristicDamageCompensation[ caf]( Assocs, tppm)
    []
    RetryRecovery[caf](Assocs,tppm)
    []
    HeuristicDecision[caf](Assocs,tppm)
    []
    TPPMInitiatedRollback[tppm,caf,macf](Assocs,tppm)
(*  TPPMAfterNodeCrash[tppm,caf](Assocs,tppm) called directly after disabling *)
endproc
process CMACFinitiated_events[ cpm, macf, aei] ( Assocs:set_of_assocs): exit(set_of_assocs):=
(*  CPMRestartAfterNodeCrash - dealt with by logstore *)
    ChannelTerm[ macf, aei](Assocs)
endproc
```

H.3.2 PM MACF Procedures

H.3.2.1 Main Procedures

```
process TPBeginDialogueReq[tppm,caf,macf]
(Assocs:set_of_assocs,tppm:tppm_info): exit(set_of_assocs,tppm_info):=                    (* 11.3.1 *)
    tppm?did:dc_id?tpsp:ServicePrim [IsTPBeginDialogueReq(tpsp)];
(* The occurrence of the following event will cause the receiving SAO to be assigned. Alternatively, the internal event may
    happen first, representing a local decision to reject *)
    (    macf!did!get_acn(tpsp)!get_qosO(tpsp)?addr:address?rchO:recovery_context_handle_Opt
        !AFBeginDialogueReq(get_ittO(tpsp),get_rttO(tpsp),get_sfu(tpsp),
        get_btO(tpsp),get_bdc(tpsp),dcc_absent,dcc_absent,get_ud(tpsp))
        [(get_addr(tpsp) <= addr) and (has_commit( tpsp) implies has_aet( addr))];
        (    let Assocs:set_of_Assocs =
            add(dialogue_entry(did,addr,rchO,tpsp),Assocs)
            in
(* The application context name and the quality of service parameters are included on this event  to communicate to the SAO
    requirements for this dialogue that are not included on the AF  sp. The address and rch parameters are used to communicate
    the values from the SAO being attached to, for later inclusion in the log record. The address is also used to check compatibility
    with the value on the TP service primitive. The dcc and lpi are set in the SACF. *)
            [dial_coord_commitment(tpsp)] ->
                InitTransactionBranch[caf,macf](did,Assocs,tppm)
            []
            [not(dial_coord_commitment(tpsp))] ->exit(Assocs,tppm)
        )
        []
        i; tppm!did!TPBeginDialogueCnf(fu_absent,rejected_provider, no_reason_given,ud_absent,false);
        exit(Assocs,tppm)
        []
        macf!address_not_recognised!get_addr(tpsp);                    (* tried to set up but unsuccessful *)
        tppm!did!TPBeginDialogueCnf(fu_absent,rejected_provider, recipient_unknown,ud_absent,false);
        exit(Assocs,tppm)
    )
endproc
(*  11.3.2 is split into two in this LOTOS description for ease of reading. *)
process AFBeginDialogueInd[tppm,macf,aei]
(Assocs:set_of_assocs,tppm:tppm_info): exit(set_of_assocs,tppm_info):=                    (* 11.3.2 TPPM *)
    macf!sup_dial?addr:address?rchO:recovery_context_handle_Opt?afsp:ServicePrim
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    [ IsAFBeginDialogueInd(afsp) and is_dialogue(afsp)
    and (has_commit( afsp) implies has_aet( addr))];
    (   choice tp_addr: address []                    (* Choose what to put on the TPBegDInd -  tp_addr <= addr *)
        let apt:AP_title_Opt=get_apt(tp_addr),apii:API_id_Opt= get_apii(tp_addr),
            aeq:AE_qual_Opt=get_aeq(tp_addr),aeii:AEI_id_Opt= get_aeii(tp_addr),
            Assocs:set_of_assocs = add(dialogue_entry(sup_dial,addr,rchO,afsp), Assocs)  in
        [tp_addr <= addr] ->
        (   [not(dial_coord_commitment(afsp))] ->
            (   i;
                (   let tpsp: ServicePrim = TPBeginDialogueInd( apt, apii, aeq, aeii, get_ittO(afsp), get_sfu(afsp), get_btO(afsp),
                    get_bdc(afsp), get_ud(afsp)) in
(* issuing this creates the TPSUI *)
                    tppm!sup_dial!tpsp; exit(add_Pending(tpsp,sup_dial,Assocs),tppm)
                )
                []
                i;
                (   choice bddO:tp_begin_dialogue_diagnostic_Opt []
                    [is_functional_unit_not_supported(bddO)] -> exit(Opt(my_functional_units))
                    []
                    [not(is_functional_unit_not_supported(bddO))] -> exit(fu_absent)
                >>  accept fuO:functional_units_Opt in
                    macf!sup_dial!AFBeginDialogueRsp(fuO,rejected_provider,bddO,dcc_absent,dataRI,ud_absent);
                    (   [not(unchained_selected(get_sfu(afsp)))] ->
                            macf!sup_dial!SAFDetachAssociationReq(begin_indication_expected);
                            exit(remove(sup_dial,Assocs),tppm)
                        []
                        [unchained_selected(get_sfu(afsp))] ->
                            macf!sup_dial!SAFDetachAssociationReq(begin_fear); exit(remove(sup_dial,Assocs),tppm)
                    )
                )
            )
            []
            [dial_coord_commitment(afsp)] ->
            (   i;exit(add_Pending(TPBeginDialogueInd(apt,apii,aeq,aeii,
                get_ittO(afsp),get_sfu(afsp),                get_btO(afsp),get_bdc(afsp),
                get_ud(afsp)),sup_dial,add_Pending(AFBeginDialogueRsp(fu_absent,
                accepted,bdd_absent,dcc_absent,dataRI,ud_absent),sup_dial,
                Assocs)),tppm)
                []
                i;
                (   choice bddO:tp_begin_dialogue_diagnostic_Opt []
                    [is_functional_unit_not_supported(bddO)] ->
                        macf!sup_dial!AFBeginDialogueRsp(Opt(my_functional_units),
                        rejected_provider,bddO,dcc_absent,dataRI,ud_absent);
                        macf!sup_dial!SAFDetachAssociationReq(begin_indication_expected);
                        exit(remove(sup_dial,Assocs),tppm)
                    []
                    [not(is_functional_unit_not_supported(bddO))] ->
                        macf!sup_dial!AFBeginDialogueRsp(get_fuO(afsp),
                        rejected_provider,bddO,dcc_absent,dataRI,ud_absent);
                        macf!sup_dial!SAFDetachAssociationReq(begin_indication_expected);
                        exit(remove(sup_dial,Assocs),tppm)
                )
            )
        )
    )
endproc
process AFBeginDialogueInd_CPM[macf,aei]( Assocs:set_of_assocs): exit(set_of_assocs):=                (* 11.3.2 CPM *)
    let chid: dc_id = next_chid(Assocs) in
    macf!chid?addr:address?afsp:ServicePrim
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992

```
    [ IsAFBeginDialogueInd(afsp) and is_channel(afsp) and has_aet( addr)];
    (   i;                                              (* channel accepted  *)
        (   let Assocs:set_of_assocs = add(channel_entry(chid,addr,afsp),Assocs) in
            macf!chid!AFBeginDialogueRsp(accepted,bcd_absent, dcc_absent,dataRI);exit(Assocs)
        )
        []
        i;                                              (* channel rejected *)
        (   choice bcdO:tp_begin_channel_diagnostic_Opt []
            macf !chid !AFBeginDialogueRsp(rejected_provider,bcdO, dcc_absent, dataRI);
            macf !chid !SAFDetachAssociationReq(FREE);
            exit(Assocs)
        )
    )
endproc
process TPBeginDialogueRsp[tppm,macf](Assocs:set_of_assocs,tppm:tppm_info):
    exit(set_of_assocs,tppm_info):=                                         (*  11.3.3 *)
    tppm?did:dc_id?tpsp:ServicePrim [IsTPBeginDialogueRsp(tpsp)];
    (   [is_coord_none(did,Assocs)] ->
        (   [is_accepted(get_bdr(tpsp))] ->
                FirstReqRsp[macf](did,add_Pending(AFBeginDialogueRsp(fu_absent,
                accepted,bdd_absent,dcc_absent,dataRI,get_ud(tpsp)),did,Assocs),tppm)
            []
            [not(is_accepted(get_bdr(tpsp)))] ->
                macf!did!AFBeginDialogueRsp(fu_absent,rejected_user,bdd_absent,
            dcc_absent,DataRI,get_ud(tpsp));
            (   [not(unchained_selected(get_sfu(did,Assocs)))] ->
                    macf!did!SAFDetachAssociationReq(FREE);
                    exit(remove(did,Assocs),tppm)
                []
                [unchained_selected(get_sfu(did,Assocs))] ->
                    macf!did!SAFDetachAssociationReq(begin_fear);
                    exit(remove(did,Assocs),tppm)
            )
        )
        []
        [is_coord_commitment(did,Assocs)] ->
        (   [is_accepted(get_bdr(tpsp)) and is_active(get_state(tppm))] ->
                FirstReqRsp[macf](did,add_Pending(AFBeginDialogueRsp(fu_absent,
                accepted,bdd_absent,dcc_absent,dataRI,get_ud(tpsp)),did,Assocs),tppm)
            []
            [is_accepted(get_bdr(tpsp)) and is_dec_rbk(get_state(tppm))] ->
                exit(Assocs,tppm)
            []
            [not(is_accepted(get_bdr(tpsp)))] ->
            (   [is_active(get_state(tppm))] ->
                    macf!did!AFBeginDialogueRsp(fu_absent,rejected_user,bdd_absent ,
                    dcc_absent,RollbackRI,get_ud(tpsp) );
                    macf!did!SAFDetachAssociationReq(rollback_confirm_expected);
                    exit(remove(did,Assocs),empty_tppm_info)
                []
                [is_dec_rbk(get_state(tppm))] ->
                    macf!did!AFBeginDialogueRsp(fu_absent,rejected_user,
                    bdd_absent,
                    dcc_absent,rollbackRC,get_ud(tpsp) );
                    macf!did!SAFDetachAssociationReq(FREE);
                    exit(remove(did,Assocs),empty_tppm_info)
            )
        )
    )
endproc
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 10026-3:1992