INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

X.860

(09/92)

DATA COMMUNICATION NETWORKS

OPEN SYSTEMS INTERCONNECTION – DISTRIBUTED TRANSACTION PROCESSING: MODEL

Recommendation X.860
Superseded by a more recent version

FOREWORD

The CCITT (the International Telegraph and Telephone Consultative Committee) is a permanent organ of the International Telecommunication Union (ITU). CCITT is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The Plenary Assembly of CCITT which meets every four years, establishes the topics for study and approves Recommendations prepared by its Study Groups. The approval of Recommendations by the members of CCITT between Plenary Assemblies is covered by the procedure laid down in CCITT Resolution No. 2 (Melbourne, 1988).

Recommendation X.860 was prepared by Study Group VII and was approved under the Resolution No. 2 procedure on the 10th of September 1992.

CCITT NOTE

In this Recommendation, the expression “Administration” is used for conciseness to indicate both a telecommunication administration and a recognized private operating agency.

CONTENTS

Superseded by a more recent version

INTRODUCTION

Recommendation X.860, 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 Recommendations in the set as defined by the Reference Model for Open Systems Interconnection (see Recommendation X.200). 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 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.

Recommendations X.860, X.861 and X.862 define an OSI TP Model, an OSI TP Service and specify 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.

Recommendations X.860, X.861 and X.862 provide sufficient facilities to support transaction processing, and establish a framework for coordination across multiple OSI TP resources in separate open systems.

Recommendations X.860, X.861 and X.862 do not specify the interface to local resources nor do they specify an application programming interface within the local system. However, future enhancement of these Recommendations may deal with these issues.

OPEN SYSTEMS INTERCONNECTION – DISTRIBUTED TRANSACTION PROCESSING: MODEL[1)]

(1992)

1 Scope

This Recommendation:

a) provides a general introduction to the concepts and mechanisms defined in this Recommendation;

b) defines a model of transaction processing;

c) defines the requirements to be met by the OSI TP Service; and

d) takes into consideration the need to coexist with other Application Service Elements, e.g RDA (Remote Database Access), ROSE (Remote Operations Service Element), and non-ROSE based applications.

This Recommendation makes sufficient provisions to allow the specification of transaction-mode communications services and protocols that meet the properties of atomicity, consistency, isolation, and durability (the ACID properties), as defined in Recommendation X.851.

This Recommendation does not specify individual implementations or products, nor does it constrain the implementation of entities or interfaces within a computer system.

2 Normative references

The following CCITT Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations | International Standards are subject to revision, and parties to agreements based on this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the Recommendations | International Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The CCITT Secretariat maintains a list of the currently valid CCITT Recommendations.

– CCITT Recommendation X.200, *Reference Model of Open Systems Interconnection for CCITT applications*. (See also ISO 7498-1.)

– CCITT Recommendation X.208, *Specification of Abstract Syntax Notation One (ASN.1)*. (See also ISO/IEC 8824.)

– CCITT Recommendation X.209, *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*. (See also ISO/IEC 8825.)

– CCITT Recommendation X.210, *OSI layer service conventions*. (See also ISO 8509.)

– CCITT Recommendation X.215, *Session service definition for Open Systems Interconnection for CCITT applications*. (See also ISO 8326 and ISO 8326 Addendum 2.)

– CCITT Recommendation X.216, *Presentation service definition for Open Systems Interconnection for CCITT applications*. (See also ISO/IEC 8822.)

– CCITT Recommendation X.217, *Information Technology – Open Systems Interconnection – Service definition for the Association Control Service Element*.

– CCITT Recommendation X.219, *Remote Operations: Model, Notation, and Service definition*. (See also ISO/IEC 9072-1.)

---

1) Recommendation X.860 and ISO/IEC 10026-1 (Information Technology – Open Systems Interconnection – Distributed Transaction Processing – Part 1:Model) were developed in close collaboration and are technically aligned.

- CCITT Recommendation X.227, *Information Technology – Open Systems Interconnection – Connection-mode protocol specification for the Association Control Service Element*.
- CCITT Recommendation X.229, *Remote Operations: Protocol Specification*. (See also ISO/IEC 9072-2.)
- CCITT Recommendation X.290, *OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT applications – General concepts*. (See also ISO/IEC 9646-1.)
- CCITT Recommendation X.291, *OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT applications – Abstract Test Suite Specification*. (See also ISO/IEC 9646-2.)
- CCITT Recommendation X.501, *Open Systems Interconnection – The Directory – Models*. (See also ISO/IEC 9594-2.)
- CCITT Recommendation X.520, *Open Systems Interconnection – The Directory – Selected Attribute Types*. (See also ISO/IEC 9594-6.)
- CCITT Recommendation X.650, *Open Systems Interconnection – Basic Reference Model – Part 3: Naming and Addressing*.
- CCITT Recommendation X.800, *Security Architecture for Open Systems Interconnection for CCITT applications*. (See also ISO/IEC 7498-2.)
- CCITT Recommendation X.851[2)] | ISO/IEC 9804[2)], *Information Technology – Open Systems Interconnection – Service definition for the Commitment, Concurrency and Recovery Service Element*.
- CCITT Recommendation X.852[2)] | ISO/IEC 9805[2)], *Information Technology – Open Systems Interconnection – Protocol specification for the Commitment, Concurrency and Recovery Service Element*.
- CCITT Recommendation X.861, *Open Systems Interconnection – Distributed Transaction Processing: Service definition*.
- CCITT Recommendation X.862, *Open Systems Interconnection – Distributed Transaction Processing: Protocol specification*.

ISO 8326/AM4, *Information technology – Open Systems Interconnection – Basic connection oriented session service definition, Amendment 4: Additional synchronization functionality*.

ISO 8822/AM5, *Information technology – Open Systems Interconnection – Connection oriented presentation service definition, Amendment 5: Additional synchronization functionality to the presentation service user*.

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

ISO/IEC 9579-1, *Information technology – Open Systems Interconnection – Remote Database Access – Part 1: Generic Model, Service, and Protocol*.

ISO/IEC 9579-2, *Information technology – Open Systems Interconnection – Remote Database Access – Part 2: SQL Specialization*.

3 Definitions

For the purposes of this Recommendation, the following definitions apply:

3.1 *Terms Defined in other Recommendations | International Standards*

3.1.1 This Recommendation makes use of the following terms defined in Recommendation X.200:

a) application-entity;

b) application-process;

---

2) Presently at the stage of draft.

c) application-protocol-data-unit;

d) concatenation;

e) open system;

f) presentation-service;

g) presentation-service-access-point;

h) presentation-service-data-unit;

i) real open system; and

j) separation.

3.1.2 This Recommendation makes use of the following terms defined in Recommendation X.800:

a) access control;

b) audit;

c) authentication;

d) confidentiality;

e) integrity; and

f) non-repudiation.

3.1.3 This Recommendation makes use of the following terms defined in Recommendation X.650:

a) application-process-invocation-identifier;

b) application-process-title;

c) application-entity-invocation-identifier;

d) application-entity-qualifier; and

e) application-entity-title.

3.1.4 This Recommendation makes use of the following term defined in Recommendation X.215:

quality-of-service

3.1.5 This Recommendation makes use of the following terms defined in Recommendation X.210:

a) request;

b) indication;

c) response;

d) confirm;

e) service primitive; primitive;

f) service-provider; and

g) service-user.

3.1.6 This Recommendation makes use of the following terms defined in ISO/IEC 9545:

a) application-association; association;

b) application-context;

c) application-context-name;

d) application-entity-invocation;

e) application-process-invocation;

f) application-service-element;

g) association control service element;

h) multiple association control function;

i) single association control function; and

j) single association object.

3.1.7 This Recommendation makes use of the following terms defined in Recommendation X.501:

a) Directory information tree;

b) Directory entry; entry;

c) distinguished name;

d) object class; and

e) relative distinguished name.

3.1.8 This Recommendation makes use of the following terms defined in Recommendation X.851:

a) atomic action data;

b) atomicity;

c) bound data;

d) consistency;

e) durability;

f) final state;

g) heuristic decision;

h) initial state; and

i) isolation.

3.2 *Terms defined in this Recommendation*

3.2.1 **application supported distributed transaction**

A transaction, where the user of the OSI TP Service is responsible for the maintenance of the ACID properties.

3.2.2 **chained sequence**

A sequence of related contiguous (provider supported) transaction branches, on the same dialogue, that are aimed at achieving a common goal.

3.2.3 **Channel Protocol Machine (CPM)**

The part of an AEI involved in OSI TP that establishes and terminates TP channels.

3.2.4 **commitment coordinator**

A TPSU invocation (TPSUI) involved in a distributed transaction together with its Transaction Processing Protocol Machine (TPPM) that arbitrates the final outcome of the transaction.

3.2.5 **control**

The permission, on a particular dialogue, for a TPSUI to communicate with its partner.

3.2.6 **coordination level**

An agreement between two TPSUIs on what mechanism will be used to guarantee the four properties of a transaction.

3.2.7 **dialogue**

The relationship between two TPSUIs that communicate with each other.

3.2.8 **dialogue recovery**

Action taken after a failure to resume use of the dialogue.

3.2.9 **distributed transaction**

A transaction, parts of which may be carried out in more than one open system.

3.2.10 **heuristic-hazard**

The condition that arises when, as a result of communication failure with a subordinate, the bound data of the subordinate's subtree is in an unknown state.

3.2.11 **heuristic-mix**

The condition that arises when, as a result of one or more heuristic decisions having been taken, the bound data of the transaction is in an inconsistent state.

3.2.12 **local resource**

A resource that is resident on the same real open system as the requestor of the resource, or a resource that is managed by an entity residing in the same real open system as the requestor of the resource.

3.2.13 **log-commit record**

A record written to the recovery log that reflects the transaction's decision to commit.

3.2.14 **log-damage record**

A record written to the recovery log that reflects the current inconsistent state of bound data in the subtree.

3.2.15 **log-heuristic record**

A record written to the recovery log that reflects the node's heuristic decision.

3.2.16 **log-ready record**

A record written to the recovery log that reflects the subtree's ability to commit or rollback the transaction.

3.2.17 **node**

Either a TPSUI, or a TPPM, or a TPSUI together with its TPPM. Which case applies can be derived from the specific context where the term is used.

3.2.18 **polarized control mode**

A mode of communication over a dialogue where only one TPSUI involved in the dialogue is allowed to have control at a time.

3.2.19 **Protocol Machine (PM)**

A generic term to denote either a Transaction Processing Protocol Machine or a Channel Protocol Machine.

3.2.20 **provider supported distributed transaction**

A transaction where the provider of the OSI TP Service is responsible for the maintenance of the ACID properties.

3.2.21 **ready-to-commit state**

A state of bound data in which, until the transaction has been terminated by commitment or rollback, the bound data can be released in either their initial or their final state.

3.2.22 **recovery**

Action taken after a failure to remove undesired consequences of the failure.

3.2.23 **recovery log**

A repository in secure storage used to record data and state information for the purposes of restart and recovery.

3.2.24 **remote resource**

A resource that is resident on a different real open system than the real open system making the request for resources.

3.2.25 **resource**

Data and processing capabilities necessary for a TPSUI to carry out the part of a transaction for which it is responsible.

3.2.26 **secure storage**

A reliable non-volatile place where stored information survives any type of recoverable failure within the real open system.

3.2.27 **shared control mode**

A mode of communication over a dialogue where both TPSUIs involved in the dialogue have control.

3.2.28 **subordinate subtree**

The subtree of a subordinate node.

3.2.29 **subtree**

A subset of a tree. The subtree of a particular node contains:

a) the node itself, called the root node of the subtree; and

b) the subtrees of each subordinate node of the root node of the subtree, recursively.

A leaf node is its own subtree.

3.2.30 **transaction**

A set of related operations characterized by four properties: atomicity, consistency, isolation, and durability. A transaction is uniquely identified by a transaction identifier.

Note – For reasons of brevity, the term “transaction” is used as a synonym of the term “provider supported distributed transaction”, from 7.6 onwards.

3.2.31 **transaction branch**

The portion of a distributed transaction performed by a pair of TPSUIs sharing a dialogue.

Note – For reasons of brevity, the term “transaction branch” is used as a synonym of the phrase “branch of provider supported distributed transaction”, from 7.6 onwards.

3.2.32 **transaction branch identifier**

An unambiguous identifier for a specific branch of a specific transaction.

3.2.33 **transaction commitment; commitment**[3)]

Completion of a transaction with the release of bound data in the final state.

3.2.34 **transaction identifier**

A globally unambiguous identifier for a specific transaction.

3.2.35 **transaction logging**

The recording of node state information and data in a recovery log.

3.2.36 **Transaction Processing Application Service Element (TPASE)**

That part of a Transaction Processing Protocol Machine (TPPM) which handles the OSI TP Protocol on a single application-association.

3.2.37 **Transaction Processing channel; channel**

A relationship over an association between two AEIs to facilitate Transaction Processing Service Provider (TPSP) recovery activity. Channels are not visible to the TPSUIs.

3.2.38 **Transaction Processing Protocol Machine (TPPM)**

The provider of the OSI TP Service for exactly one TPSUI. A TPPM handles the TP Protocol on all associations that have been used for its TPSUI's activity.

3.2.39 **Transaction Processing Service Provider (TPSP)**

The provider of the OSI TP Service. The TPSP provides the OSI TP Service to all the TPSUIs involved in a particular dialogue tree. The TPSP spans several application-process-invocations (APIs) and is the conceptual view of the OSI TP Service as a whole.

3.2.40 **Transaction Processing Service User (TPSU)**

A user of the OSI TP Service: it refers to a specific set of processing capabilities within an application-process.

3.2.41 **TPSU Invocation (TPSUI)**

A particular instance of a TPSU performing functions for a specific occasion of information processing.

3.2.42 **transaction recovery**

Action taken after a failure in order to put all the bound data of that transaction into a consistent state.

3.2.43 **transaction rollback; rollback**[3)]

Completion of a transaction with the release of bound data in the initial state.

---

3) The terms "commitment" and "rollback" have a different scope from that defined in Recommendation X.851 (ISO/IEC 9804). Recommendations X.860, X.861 and X.862 (ISO/IEC 10026) are concerned with the commitment and rollback of a complete transaction, whereas Recommendation X.851 (ISO/IEC 9804) refers to the commitment and rollback of a single atomic action branch.

3.2.44 **TPSU-title**

A name, unambiguous within the scope of the application-process containing the TPSU, which denotes a particular TPSU. The TPSU-title implies the type of processing (capabilities) of this TPSU.

3.2.45 **tree**

An acyclic graph with nodes arranged in a hierarchical structure.

3.2.46 **unchained sequence**

A sequence of non-contiguous (provider supported) transaction branches, on the same dialogue, that are aimed at achieving a common goal.

3.2.47 **user-ASE**

An application-specific ASE.

4 Abbreviations

For the purposes of this Recommendation, the following definitions apply:

ACID Atomicity, Consistency, Isolation, and Durability

ACSE Association Control Service Element

AE Application-Entity

AEI Application-Entity Invocation

ALS Application Layer Structure

AP Application-Process

APDU Application-Protocol-Data-Unit

API Application-Process Invocation

ASE Application Service Element

CCR Commitment, Concurrency, and Recovery

CPM Channel Protocol Machine

DIT Directory Information Tree

MACF Multiple Association Control Function

OSI Open Systems Interconnection

OSIE Open Systems Interconnection Environment

PM Protocol Machine (either a TPPM or a CPM)

PICS Protocol Implementation Conformance Statement

PSAP Presentation Service Access Point

PSDU Presentation-Service-Data-Unit

RDA Remote Database Access

ROSE Remote Operations Service Element

SACF Single Association Control Function

SAO Single Association Object

TP Transaction Processing

TPASE Transaction Processing Application Service Element

TPPM Transaction Processing Protocol Machine

TPSP Transaction Processing Service Provider

TPSU Transaction Processing Service User

TPSUI Transaction Processing Service User Invocation

U-ASE User-Application Service Element

5 Conventions

This Recommendation is guided by the conventions discussed in Recommendation X.210 as they apply to the OSI TP Service.

6 Requirements

6.1 *Introduction*

This clause summarizes the requirements for OSI TP. It includes both requirements which are addressed by Recommendations X.860, X.861 and X.862, and also requirements which are not addressed and which require further study; these additional requirements are candidates for further standardization as new editions of this Recommendation and/or additional Recommendations.

6.2 *User requirements*

In order to satisfy user needs, the OSI TP standards

a) define procedures which support distributed transactions, as discussed in 7.2. These procedures

   1) allow a distributed transaction to be organized into a transaction tree;

   2) provide multi-party coordination (part of which is multi-party commitment), including local resources;

   3) allow restoration to a consistent state, following failure, of the state/context of a distributed transaction and of bound data;

   4) allow the detection of a distributed transaction's failure to achieve ACID properties;

   5) allow a distributed transaction to be restarted following successful state restoration; and

   6) indicate the completion status of a transaction;

b) provide for the delimitation of a sequence of logically related transactions;

c) allow the grouping of TPSUs within an application-process;

d) allow for one, or more, of the following security requirements:

   Note – The provision for security is for further standardization as an amendment.

   1) *access control:* It must be possible to support multiple access control policies. At least those types described in Recommendation X.800 (Administration imposed and dynamically selectable, rule-based and identity-based) should be included;

   2) *access control granularity:* It should be possible to classify OSI TP objects into groups in order to simplify the specification of access control and allow for distribution of the authorisation database. Such classification should be for optimisation, not a substitute for individual auditing;

3) authentication among

   – corresponding TPSUIs;

   – TPPMs;

   – AEIs; and

   – TPSUIs and TPPMs. However, this is considered to be a local matter;

4) *non-repudiation:* To prevent denial of having participated in a specific transaction or dialogue;

5) *confidentiality:* To prevent unauthorized reception of part, or all of the information exchanged within a dialogue tree;

6) *integrity:* To detect unauthorized changes to part, or all of the information exchanged within a dialogue tree; and

7) *audit:* To record significant security events occurring within a dialogue tree;

e) allow conformance testing of the protocol defined by the *protocol* Specification and delineate clearly the static conformance requirements (through a PICS).

Note – A PICS proforma for the OSI TP Protocol is for further study.

6.3 *Modelling requirements*

The OSI TP Model provides a model of transaction processing and the communications mechanisms to support it which are consistent with the OSI architecture defined in Recommendation X.200 and ISO/IEC 9545, and which addresses the following requirements:

a) definition of mechanisms for partitioning into transactions the interactions between application-processes of two or more open systems. In particular, these mechanisms provide for

   1) indication of the completion status of a transaction;

   2) support of transactions which do not require the full distributed commitment mechanisms to ensure the ACID properties: the application is responsible for ensuring the ACID properties; and

   3) flexibility in order to match the choice of data transfer method to the semantics of the transaction;

b) specification of mechanisms to use the services of the Presentation Layer;

c) procedures that have acceptable performance and efficiency; and

d) procedures that cover a wide variety of needs (short or long, simple or complex transactions).

Note – Some of these procedures are candidates for further standardization.

6.4 *OSI TP Service and Protocol requirements*

The OSI TP Service and Protocol provide for

a) flexibility to handle changing load conditions;

b) efficient support of operations under high, low or burst conditions;

c) efficient handling of short APDUs;

d) acceptable response time for users;

e) resilience from failure, including the means to recover and restart processing after faults have been corrected or circumvented;

f) optimal resource usage; and

g) minimization of the dependence of local resource control upon communications.

In order to meet these requirements, the OSI TP Protocol

a) optimizes the use of the Presentation Layer Service;

b) minimizes the communication overhead required for each transaction – in particular, the OSI TP Protocol limits the number of round trips required by the communication protocols to be no greater than the number of round trips required by the semantics of the application;

c) optimizes operations to the needs of high volume transaction processing; and

d) optimizes operations to the needs of the normal case rather than to those of exception cases.

7 Concepts of distributed TP

7.1 *Transaction*

A transaction is a set of related operations characterized by four properties: atomicity, consistency, isolation, and durability.

7.2 *Distributed transaction*

A transaction that may span more than one open system is called a distributed transaction.

A distributed transaction is composed of at least as many parts as there are open systems involved in this distributed transaction. Within each open system, a part of the distributed transaction relates to an entity called a TP Service User (TPSU).

The TPSU is the user of the OSI TP Service. It refers to a specific set of processing capabilities within an application-process. There may be zero, one, or more TPSUs within any given application-process.

Note – A TPSU may in turn be distributed within an application-process. This Recommendation does not preclude such a refinement, but does not discuss it, since distribution within an open system lies beyond the scope of OSI.

A TPSU invocation (TPSUI) models, from the perspective of the OSIE, a particular instance of a TPSU, within an application-process-invocation, performing functions for a specific occasion of information processing.

To maintain the four properties of transactions, coordination is required among the TPSUIs performing a distributed transaction. Such coordination requires communication among TPSUIs.

7.3 *Dialogue*

TPSUIs communicate among themselves in a peer-to-peer relationship; this peer-to-peer relationship between two TPSUIs is called a dialogue.

In a dialogue, TPSUIs may communicate for the following purposes:

a) transfer of data;

b) error notification;

c) initiation, commitment, or rollback of a transaction;

d) orderly or abrupt termination of their dialogue; and

e) handshake activities.

Dialogues may be controlled in two modes:

a) polarized control, when only one TPSUI has control of the dialogue at a time; and

b) shared control, when both TPSUIs have control of the dialogue simultaneously.

In polarized control mode, a TPSUI needs have control of the dialogue to initiate a request other than

a) error notification;

b) rollback of a transaction;

c) abrupt termination of the dialogue; and

d) request control.

7.4 *Dialogue tree*

A dialogue tree is a tree with TPSUIs as nodes, and dialogues as arcs between the nodes.

Within the dialogue tree, the TPSUI that establishes the dialogue is referred to as the direct superior of the TPSUI with which the dialogue is established. The TPSUI with which the dialogue is established is referred to as the direct subordinate of the adjacent superior TPSUI.

The TPSUI in the dialogue tree that has no superior is called the root TPSUI. A TPSUI that has no subordinate is called a leaf TPSUI. A TPSUI that has both a superior and at least one subordinate is called an intermediate TPSUI.

7.5 *Transaction branches*

When requested, the TPSP provides the TPSUIs with a commitment service for use on a given dialogue. The value of the coordination level determines whether the commitment service is used, on that dialogue, by the TPSUIs:

a) “commitment” when the service is used by the TPSUIs; or

b) “none”, otherwise.

The portion of a distributed transaction performed by a pair of TPSUIs sharing a dialogue is called a transaction branch.

There are two basic kinds of transaction branches with respect to the division of responsibility between the TPSP and the TPSUIs:

a) application supported transaction branches: transaction branches operating on a dialogue with coordination level equal to “none”.

   For application supported transaction branches, the TPSUI is responsible for maintenance of the ACID properties, recovery, and delineation of transaction branches.

   Note – The mechanisms, if any, required to maintain the ACID properties for application supported transaction branches are beyond the scope of this Recommendation.

   The TPSP provides only access to data transfer, error notification and dialogue control services, and is not aware of the beginning or completion of the application supported transaction branches; and

b) provider supported transaction branches: transaction branches operating on a dialogue with coordination level equal to “commitment”.

   For provider supported transaction branches, the TPSP is responsible for coordinating the maintenance of the ACID properties (therefore making use of globally unambiguous transaction identifiers, commitment, etc.), recovery, and delineation of transaction branches, as well as providing access to the remaining services.

Hereafter, for reasons of brevity, the term “provider supported transaction branch” is referred to by the short term “transaction branch”. When needed, the term “application supported transaction branch” is used explicitly.

7.6 *Transaction tree*

A transaction tree is a tree with TPSUIs as nodes, and transaction branches as arcs between the nodes.

Within a transaction tree, the TPSUI that initiates the transaction branch is referred to as the direct superior of the TPSUI with which the transaction branch is being established. The TPSUI with which the transaction branch is being established is referred to as the direct subordinate of the adjacent superior TPSUI.

The TPSUI in the transaction tree that has no superior is called the root TPSUI. A TPSUI that has no subordinate is called a leaf TPSUI. A TPSUI that has both a superior and at least one subordinate is called an intermediate TPSUI.

The root TPSUI together with its TPPM forms the commitment coordinator for the transaction.

7.7 *Channels*

During recovery there is a requirement for the AEIs to communicate directly with each other, without the involvement of any TPSUIs. This requirement is realized by channels.

A channel exists between two AEIs. Channels are established and terminated by a Channel Protocol Machine (CPM). The CPMs in two peer systems may establish one or more channels between them for the purpose of recovery.

A channel has the following properties:

a) it is not directly visible to the TPSUIs. There are, therefore, no OSI TP primitives referring to channels in the OSI TP Service; and

b) a channel is assigned by a CPM to a TPPM for the purpose of recovery.

For the purpose of recovery, channels are modelled as being used to recover one transaction branch at a time.

7.8 *Handshake*

TPSUIs may have to synchronize their activities, in order to reach a mutually agreed processing point. The semantics of such a processing point are application dependent.

When requested, the TPSP provides the TPSUIs with a handshake service, available for the duration of the dialogue, as a tool for application structuring, independently from the mode in which dialogues may be controlled.

8 Model of the OSI TP Service

8.1 *Nature of the OSI TP Service*

The term OSI TP Service pertains to the service provided by the TPSP and used by the TPSUIs.

The following functions are associated with the OSI TP Service:

a) establishment, maintenance, and termination of the dialogue between two TPSUIs. The OSI TP Service

   1) provides for the selection of a TPSU from a set of TPSUs. The TPSU-title serves that purpose;

   2) ensures that the attributes requested by the initiating TPSUI are compatible with those of the recipient TPSUI. If so, the dialogue is established between a **new** invocation of the requested TPSU and the initiating TPSUI; and

      Note – From the OSIE perspective, a “new invocation” means a TPSU invocation which is not currently in the OSIE. It is a local matter as to whether the “new invocation” is mapped, in a real open system, to a new instance of the TPSU, or to an old instance that is being reused.

   3) provides means to both TPSUIs to interact, access remote resources, and possibly include them in a transaction.

b) according to the selected coordination level, overall coordination of **all** resources, in a reliable fashion, to either successfully or unsuccessfully terminate a transaction, achieving consistent state of **all** resources, except possibly when heuristic decisions are taken: the ACID properties apply to the whole transaction, in particular to both remote and local resources.

In order to allow control and management of local resources by either the TPSP, the TPSUI, or both, the coordination of **all** resources may be fully located within the TPSP, or may be shared between the TPSP and the TPSUI. In the latter case, the TPSUI gathers the related information from part or all of its local resources and controls the subsequent commitment or rollback of these local resources upon decision of the TPSP.

The OSI TP Service

1) includes the necessary provisions to coordinate all remote resources in order to ensure the application of the ACID properties: at termination of a transaction, the TPSP is responsible for coordinating the correct commitment or rollback of the entire set of remote resources; and

2) provides the ability to include local resources in the termination of the transaction. Depending on the sharing between the TPSP and the TPSUIs,

   i) the TPSP includes the local resources together with the remote resources in the termination of the transaction; or

   ii) the TPSP provides all the information required by the TPSUIs to correctly include (other) local resources such that the ACID rules can be applied to **all** resources.

   The TPSP guarantees, by the execution of the appropriate protocol, that all resources obey the ACID properties. In particular, the TPSP includes appropriate recovery mechanisms to re-establish a consistent state of all resources after failure and to resume transaction processing after re-establishment of a consistent state of all resources, when possible.

8.2 *Rules on dialogue trees*

8.2.1 *Growth of dialogue trees*

A TPSUI may activate remote TPSUIs in order to execute parts of a distributed transaction; this is done by having the remote open system invoke a **new** TPSUI and then establishing a dialogue with it (see also 8.2.3 and 8.4.1). It is in this way that an arc of the dialogue tree is added.

Note – From the OSIE perspective, a "new invocation" means a TPSU invocation which is not currently in the OSIE. It is a local matter as to whether the "new invocation" is mapped, in a real open system, to a new instance of the TPSU, or to an old instance that is being reused.

Attributes of the dialogue indicating the type of transaction processing to be performed are specified at establishment of the dialogue. These attributes determine the subset of communication facilities to be selected on that dialogue. These may include

a) the polarized control mode or the shared control mode;

b) the handshake service; and

c) the commitment service.

A dialogue with an initial coordination level of "none" may be added to a dialogue tree at any time. A dialogue with a coordination level of "commitment" may only be added when it is permitted to start a transaction, or to add a transaction branch to the current transaction.

A TPSUI may establish dialogues with one or more subordinate TPSUIs. However, two TPSUIs share at most a single dialogue. Communication may take place on some or on all dialogues of a TPSUI at the same time. All the dialogues of a TPSUI belong to the same dialogue tree.

8.2.2 *Pruning of dialogue trees*

Two TPSUIs that no longer need to communicate with each other terminate their dialogue. They may do so at any time, provided that they ensure that the four ACID properties are still maintained.

A dialogue can terminate normally if and only if there is no transaction branch in progress on that dialogue. Dialogue termination is possible when

a) the coordination level is "none"; or

b) the current transaction branch is terminated, and the next one has not yet been started.

Dialogue termination may also occur upon communication failure or node crash. In this event, the corresponding transaction branch is terminated with the dialogue.

When a dialogue between two TPSUIs is terminated, the dialogues in the subtree of the subordinate TPSUI do not necessarily need to be terminated. Hence, a new dialogue tree, previously part of an already established dialogue tree may be created. The new dialogue tree is independent from the dialogue tree from which it originated. The intermediate node, for which the dialogue with the superior has been terminated, becomes the root of the new dialogue tree.

As dialogues are established and terminated, the dialogue tree changes.

8.2.3 *Support of dialogue trees*

A dialogue between two TPSUIs is supported by a single application-association at a time.

When a dialogue is related to an application-association, there is a one-to-one correspondence between them at any given time. However, the lifetime of a dialogue and that of an application-association may be distinguished in that:

– the lifetime of an application-association may span the lifetime of one or more dialogues; and

– the lifetime of a dialogue may span the lifetime of one or more application-associations. This means that a dialogue may be resilient to application-association failures.

Note – The provision for dialogue recovery is for further standardization as an amendment.

The OSI TP Service does not constrain the establishment or existence of application-associations. In particular, they are not constrained to a tree or other topological structure between AEIs. Hence, they are considered to form a graph of interconnected open systems.

To be able to support a dialogue, an application-association must have been established

a) between the AEIs supporting the communications requirements of the TPSUs related to the requested dialogue;

b) with an application context that supports the communication requirements of the TPSUs related to the requested dialogue;

c) with Presentation and Session Service support compatible with the requirements of the requested dialogue; and

d) with quality-of-service compatible with the requirements of the requested dialogue.

8.3 *Rules on transaction trees*

8.3.1 *Growth of transaction trees*

A new transaction branch may only be added to a transaction tree prior to the commencement of the transaction termination procedures (see 8.6).

There are two ways to grow a transaction tree:

a) a new transaction branch is added to the transaction tree, as perceived by the TPSP, by establishing a new dialogue with coordination level of “commitment”; and

b) where the coordination level is permitted to change dynamically (see also 8.3.3), a new transaction branch is added to the transaction tree when the coordination level changes from “none” to “commitment”. Only a superior node of the dialogue tree is allowed to modify the coordination level.

8.3.2 *Lifetime of transactions trees*

A transaction tree lasts only for the duration of a single transaction.

Where the coordination level is permitted to change dynamically, the coordination level can only change to “none” at completion of a transaction branch. Only a superior node of the dialogue tree is allowed to modify the coordination level.

The growth and termination of a transaction tree are not immediate. Both actions require multiple elementary exchanges that must be propagated throughout the transaction tree.

8.3.3 *Support of transaction trees*

The existence of a dialogue between two TPSUIs is a prerequisite for a transaction branch to exist between the two TPSUIs.

Where the coordination level is “commitment”, there is a one-to-one correspondence between a dialogue and a transaction branch at any given time. The TPSP is aware of the relationship between dialogues in a dialogue tree and the branches in the corresponding transaction tree(s), and coordinates their combined operations, for example, to achieve consistent commitment semantics across all of the open systems involved in a transaction.

The root of a transaction tree is not necessarily placed at the root of the dialogue tree. Within the bounds of the transaction tree, and with respect to the superior to subordinate relationships, there is a one-to-one correspondence between the nodes of the transaction tree and nodes of its supporting dialogue tree. The transaction tree and its supporting dialogue tree have the same orientation.

A dialogue whose coordination level is “none” does not support a transaction branch of a transaction tree.

The same dialogue tree may be used to support a sequence of distinct transactions. The relationship between dialogues in the dialogue tree persists across these distinct transactions. Within the bounds of a dialogue, a sequence of one or more transaction branches may take place. Two types of sequences are permitted:

a) chained sequences: these are sequences of transaction branches on the same dialogue that operate at the same coordination level (“commitment”). Each transaction branch is initiated directly by the superior TPPM; and

b) unchained sequences: these are sequences of transaction branches on the same dialogue such that there is a dialogue coordination level transition to “none” between each transaction branch. At dialogue establishment time as well as at dialogue termination time, the coordination level may be either “none” or “commitment”. Each transaction branch is initiated by the superior TPSUI.

If a part of a dialogue tree exists which has no transaction in progress (i.e. the dialogues have a coordination level of “none”), then a TPSUI in this part of the dialogue tree may initiate a new transaction. This can lead to there being zero, one or more transaction trees in the one dialogue tree, at the same time.

At any given time, transaction trees are disjoint among themselves. Between two transaction trees, disjunction shall be guaranteed by at least one dialogue with coordination level of “none”.

After dialogue termination between an intermediate node and its superior, the intermediate node becomes the root of a new dialogue tree, and may become the root of a transaction tree.

Figure 1/X.860 shows the correspondence, over time, between transactions branches, dialogues and associations. This set of correspondences is depicted between two adjacent open systems.

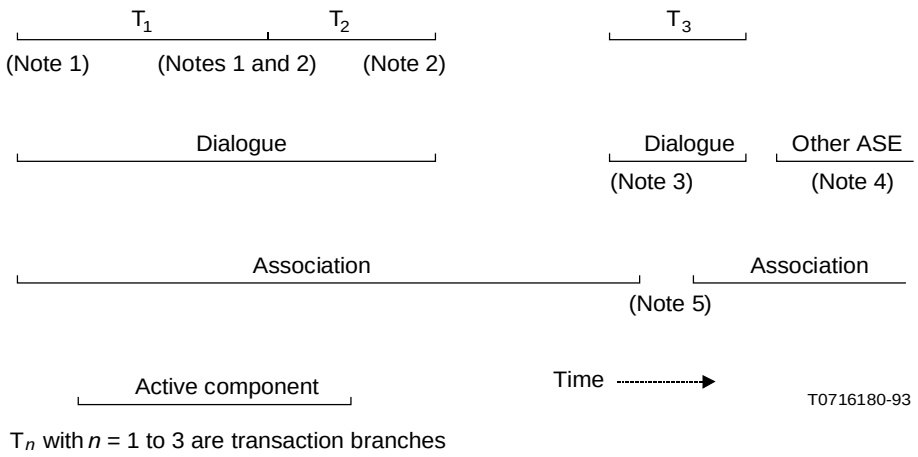
8.4 *Naming*

In addition to the naming facilities already established for OSI in ISO 7498-3, OSI TP requires titles for TPSUs, and identifiers for transactions and transaction branches. Definitions for these names and identifiers are given in clause 3.

8.4.1 *TPSU-title*

The TPSU-title is used during dialogue establishment to select a TPSU within a designated application-process with which the dialogue is to be established. The dialogue is established between the initiating TPSUI and a recipient TPSUI of the TPSU specified by the TPSU-title. The dialogue is established over an application-association (either pre-existing or newly established) between the two application-entity-invocations supporting the respective TPSUIs.

By denoting the target TPSU for dialogue establishment, the TPSU-title indicates the processing capabilities of the TPSU.



Note 1 – The beginning of a transaction branch occurs either at the beginning of a dialogue or during a dialogue.

Note 2 – The end of a dialogue implies the end of the current transaction branch. The end of a transaction branch occurs either during a dialogue or at the end of a dialogue.

Note 3 – A dialogue may follow another dialogue within the bounds of the same application-association.

Note 4 – Another ASE can use the application-association when the dialogue terminates.

Note 5 – A dialogue may survive an association failure. However, the necessary recovery mechanisms are not covered by X.861 and X.862.

FIGURE 1/X.860

Transaction branches, dialogues and associations

In the case where the dialogue is established over a pre-existing application-association, the TPSU-title may be used to derive information necessary to enable the initiating TPPM to select a suitable application-association from among those which may be available. An example of this information is the application-context.

In the case where no pre-existing association is available, the TPSU-title may be used to derive information necessary to enable the TPPM to establish the required association.

The TPSU-title is unambiguous within the scope of an application-process.

8.4.2 *Transaction identifier*

A transaction is denoted unambiguously within the OSIE by a transaction identifier. The transaction identifier consists of

a) the application-entity-title of the application-entity that supports the root node of the transaction; and

b) the transaction suffix, the value of which is unambiguous within the scope of the application-entity that supports the root node of the transaction. For example, the transaction suffix may be an integer which is incremented by one for each new transaction instantiated.

Note – The transaction identifier should also be globally unambiguous over time, with respect to recovery and business auditing requirements.

8.4.3 *Transaction branch identifier*

A transaction branch is denoted unambiguously within the scope of a transaction by a transaction branch identifier. The transaction branch identifier consists of

a) the application-entity-title of the application-entity that supports the node that initiates the transaction branch; and

b) the transaction branch suffix, the value of which is unambiguous within the scope of the application-entity that supports the node that initiates the transaction branch.

8.5 *Data transfer*

8.5.1 *Requirements and objectives*

To meet the requirements of TPSUIs involved in a distributed transaction to exchange data, the OSI TP Service allows data transfer to meet the following objectives:

a) the OSI TP Service allows the TPSUI to convey data according to its own semantics;

b) data transfer always relates to a single dialogue;

c) the TPSUI is free to organize the style of its semantic exchange using one or more specific user-ASEs. In particular, its semantic exchanges may be based on different disciplines; and

d) the definition of user-ASEs may be the same whether they work with or without OSI TP.

8.5.2 *Coordination of data transfer*

User-ASEs generate the data transfer APDUs which are mapped onto underlying services, coordinated by the SACF.

The TPPM handles the protocol for dialogue management; it does not itself directly generate data transfer APDUs. The TPPM determines the temporal ordering of the use of the underlying application-association for Transaction Processing.

Hence, within OSI TP, data transfer

a) may occur only within the bounds of a dialogue;

b) is subject to control modes. In particular, in the polarized control mode, data may only be sent if the TPSUI has control of the dialogue. The selection of the control mode depends on the particular requirements expressed by the user-ASEs; and

c) is subject to the states of the TPPM.

The TPPM ensures that data transfer is coordinated with the commitment phases during the termination of the transaction.

8.6 *Coordination of resources*

8.6.1 *Commitment*

The termination phase of a distributed transaction is entered upon request from the root TPSUI of the transaction tree. Within the transaction tree, the TPSP coordinates the termination phase among the TPSUIs to ensure that the transaction's bound data will be released in a consistent state. Coordination of the termination phase occurs in two steps:

a) commitment phase 1; and

b) commitment phase 2.

After completion of these two steps, commitment is complete. A new transaction may or may not begin.

8.6.1.1 *Commitment phase 1*

Each node is informed by its superior that the termination phase was entered; in particular, no more data will be received from the superior, and bound data will be released either in their initial state or in their final state.

If the node agrees to proceed, it attempts to place the bound data, within its subtree, in the ready-to-commit state. Bound data are in the ready-to-commit state if, until the transaction has been terminated by commitment or rollback, they can be released in either their initial or their final state. The node attempts to place its local bound data in the ready-to-commit state. For remote resources, it informs its subordinates; and so on, recursively.

Whenever, within its subtree, all its bound data are in the ready-to-commit state, the node notifies its superior, and waits for the final outcome of the transaction; and so on, recursively.

If the node is unable to place the bound data in the ready-to-commit state, it initiates rollback of the transaction.

8.6.1.2 *Commitment phase 2*

Within the transaction tree, the transaction may be committed whenever

a) all the bound data are in the ready-to-commit state; and

b) there are no ongoing operations changing the transaction tree, e.g. establishment of new transaction branches, or affecting communication between its nodes.

Each node is ordered by its superior to release the bound data within its subtree into the final state. The node commits its local bound data. For remote resources, it orders its subordinates to commit; and so on, recursively.

Whenever, within its subtree, all its bound data have been released in the final state, the node informs its superior; and so on, recursively. The transaction is complete.

8.6.2 *Rollback*

Rollback of a transaction may be initiated by any node of the transaction tree. Rollback returns the transaction's bound data to their initial state.

Issuing rollback does not, by itself, make the underlying dialogue terminate. If a TPSUI desires to terminate the dialogue, it may abort it. If a dialogue is aborted before the commencement of the transaction termination procedures, the transaction is rolled back.

After rollback has been completed, a new transaction may (but need not necessarily) be initiated.

8.6.3 *Heuristic decisions*

After it has entered commitment phase 1, a subordinate node may decide to release part or all of its bound data in the final state or initial state even though it has not been notified by its superior of the final outcome of the transaction. Such a decision is called a heuristic decision.

Heuristic decisions may be taken by individual nodes as the result of a communication failure, or as a result of system specific local conditions. The decision whether or not to take one or more heuristic decisions and what decisions to take is a local matter. Within the scope of OSI TP, whenever a node takes a heuristic decision, no propagation of the decision occurs to other nodes.

A node that has taken a heuristic decision is required to record that decision, using a log-heuristic record, in secure storage. If the state of the node's bound data and the outcome of the transaction prove to be consistent, then the log-heuristic record is erased, and normal termination of the transaction proceeds.

8.6.4 *Detection of heuristic inconsistency*

A node that has taken a heuristic decision determines that a heuristic inconsistency exists if the state of its local bound data is inconsistent with respect to the outcome of the transaction. The node can make this determination as soon as it is informed of the final outcome of the transaction by its direct superior. If the state of the node's bound data is inconsistent with the outcome of the transaction, then the ACID properties have been violated. This is a heuristic-mix condition.

A heuristic-hazard condition exists when a node is unable to determine the exact state of the bound data for its subordinate nodes within their subtree. This would result if communication were lost with one or more subordinates. If the final outcome of the transaction was to rollback, the state of the bound data of the subtree cannot be reported to the direct superior. This is due to presumed rollback (see 8.7.2 and Annex C). This is a heuristic-hazard condition, as the state of the bound data within the subtree is potentially a heuristic-mix.

A heuristic-hazard condition also exists if a TPSUI is unable to determine whether the state of the local bound data is consistent with the outcome of the transaction. This would come about as a result of a local loss of communications.

8.6.5 *Reporting*

Each node acquires knowledge of the state of bound data within its subtree. In particular, the root node acquires knowledge of the state of bound data within the whole transaction tree.

Within the TPSP, each TPPM collects reports on the state of the bound data within its subtree, as a result of

a) the state of the node's local bound data compared to the final outcome of the transaction; and

b) the report, from each subordinate, on the state of the bound data in the subordinate's subtree.

If the node determines that the state of the bound data in its subtree is consistent with the final outcome of the transaction, the TPPM reports to its superior that all bound data in its subtree is in a consistent state.

If the node determines that the state of the bound data within its subtree is inconsistent with the final outcome of the transaction, and is unable to compensate for the inconsistency, then the TPPM

a) as reports are collected, retains knowledge of bound data inconsistency within the node's subtree by means of the log-damage record. Refer to Table 1/X.860 for resulting values of the log-damage record according to reported inconsistency;

b) reports the inconsistency to its TPSUI;

c) reports to the superior node, if any, whenever a complete report of the state of the bound data within its subtree is available; and

d) reports the inconsistency to some local entity, e.g. a system operator.

The log-damage record is kept after the propagation of the final outcome of the transaction, until assurance has been received that its superior has received appropriate reporting.

Note 1 – This does not imply that the information about the inconsistency may not be kept until damage has been repaired.

Note 2 – The mechanism by which the subordinate node is assured that the superior is aware of the heuristic damage is outside the scope of OSI TP.

Note 3 – The provision for a more comprehensive reporting scheme is a candidate for further standardization as an amendment.

TABLE 1/X.860

Update of log-damage record

Previous state of log-damage record	Reported inconsistency		
	No inconsistency	Heuristic-hazard	Heuristic-mix
No log-damage record	No report	Heuristic-hazard	Heuristic-mix
Heuristic-hazard	Heuristic-hazard	Heuristic-hazard	Heuristic-mix
Heuristic-mix	Heuristic-mix	Heuristic-mix	Heuristic-mix

8.7 *Recovery*

8.7.1 *Types of failure*

8.7.1.1 *Introduction*

Table 2/X.860 identifies the potential causes of failures, the types of failures which may occur during a transaction, and the actions which should be taken to return the transaction to a manageable state.

TABLE 2/X.860

Types of failures

Possible causes	Failure type	Action by TPPM
Application error	Locally recoverable	None
Transaction abort, or dialogue abort	Recoverable before node is ready to commit	Rollback
Dialogue abort	Recoverable after node is ready to commit	Recovery procedures
Node crash; TPPM failure, or AEI failure	Atomic action data unavailable	Recovery of atomic action data; dialogue abort; possibly, association abort, and recovery procedures
Storage media failure	Atomic action data destruction	Beyond the scope of X.860, X.861 and X.862

8.7.1.2 *Locally recoverable failures*

If a failure occurs, the TPSUI may be able to recover by its own means such that the transaction can continue to commit. If the TPSUI or TPPM does so, there is no external manifestation (other than possibly delays) of the incident. This case is a local matter.

8.7.1.3 *Failures recoverable before the node is ready to commit*

Any failure occurring before the node is ready to commit causes a rollback. These failures may originate from either:

a) transaction abort, due to the following:

   1) the inability of the TPSUI to operate for the current transaction such that a rollback is explicitly requested;

   2) a distributed deadlock, where a transaction is part of a waiting cycle with other transactions;

   3) a storage media failure where the current value of the bound data is no longer accessible, but the bound data in their initial state are available; or

   4) a storage media failure where the bound data are destroyed, but the state of the transaction is known and local intervention is required to re-construct the bound data;

b) dialogue abort, due to the following:

   1) a failure in the application-association supporting the dialogue. This could occur as a result of a failure in ACSE, the Presentation service, or a supporting service (e.g. the Session service);

2) an application protocol error in any of the following protocols on the dialogue:

   – user-ASEs;

   – OSI TP; or

   – CCR;

3) a user-ASE failure; or

4) a failure of the TPSUI and/or the TPPM such that they are unable to continue communication on the dialogue (e.g. node crash).

8.7.1.4 *Failures recoverable after the node is ready to commit*

Recoverable failures occurring after the node is ready to commit are those failures that cause dialogue aborts. Upon failure, a recovery procedure is initiated to complete the transaction. Refer to 8.7.1.3, b) for possible causes of dialogue abort.

8.7.1.5 *Atomic action data unavailability*

A failure has occurred on an open system such that the working copy of the atomic action data for the current transaction is unavailable. The working copy of the atomic action data may have become unavailable (i.e. lost) due to failures, for example a TPPM failure, an AEI failure, or a node crash.

The recovery log must be read to restore the working copy of the atomic action data for the transaction. All dialogues and/or underlying associations that pertain to the current transaction are aborted.

8.7.1.6 *Atomic action data destruction*

The atomic action data have been lost due possibly to a storage media failure. Recovery from this type of failure is beyond the scope of this Recommendation.

8.7.2 *Support for recovery*

The TPSP includes facilities to recover after communication failure or node crash. Recovery support is two-fold:

a) *Recovery of transactions*

   Recovery of a transaction means that, after occurrence of a failure, all bound data that have been involved in the transaction will be re-instated to their final state or their initial state. It is the responsibility of the TPSP to ensure that all resources are re-instated to the same consistent state, i.e. either the final or the initial state.

   Transaction recovery is achieved within the bounds of the transaction tree by the TPSP. Outside of the transaction tree, recovery is the responsibility of the TPSUIs.

   Provision for transaction recovery requires that key steps in the progress of transaction branches (atomic action data) have been appropriately logged in every open system involved in the transaction tree. The presumed rollback two-phase commit protocol is used.

   The following information must not be lost, and thus must be saved in the recovery log:

   1) *Atomic action data:* These data are not subject to the commit/rollback procedure, but are used during the recovery process.

   2) *Bound data:* The data of the objects on which the transaction operates. These data are subject to the commit/rollback procedure, are invisible from outside of the transaction during the execution of the transaction, and will only be available to any other transaction after the transaction has terminated.

b) *Recovery of dialogues*

   Recovery of a dialogue means that the TPSUIs can initiate the next transaction as if no failure had occurred. Dialogue recovery may apply to any dialogue within the dialogue tree.

   Note – The provision for dialogue recovery is a candidate for further study.

8.7.3 *Node states*

When a failure occurs, the transaction may either be active or in the termination process. In the latter case, timing is an important factor in the recovery mechanism. The termination of a transaction is not immediate since several steps and exchanges are needed in the whole transaction tree between the moment where a TPSUI asks for the termination and the moment it is informed of the completion of its request (see Figure 2/X.860).

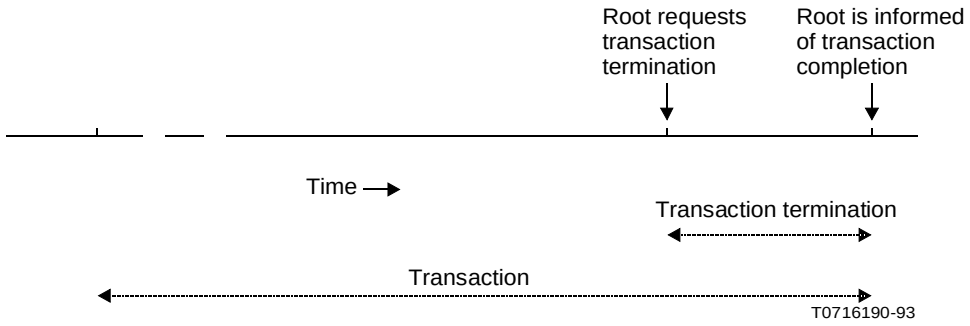


FIGURE 2/X.860

Termination of a transaction

When a failure occurs during transaction termination, the branches of the transaction may be in different states. Therefore recovery may require different types of action depending on the states of the nodes within the transaction tree.

The recovery actions of a node are dependent upon:

a) the role played by the node in the termination of the transaction, as to whether it is the commitment coordinator or not; and

b) the state of the node (with respect to recovery) for the transaction.

If the node is the commitment coordinator (the commitment coordinator is always the root of the transaction tree), its states can be one of the following:

a) ACTIVE: The processing of the transaction is going on. The node can choose to order rollback of the transaction and release bound data under its responsibility in their initial state without threatening their consistency.

   According to the presumed rollback approach, the node is not required to log in the recovery log the creation of any transaction branch; or

b) DECIDED: The node has decided, after consultation with its subordinates, in which state the bound data are to be left upon completion of the transaction.

   If the decision was to commit, the node must write a log-commit record in the recovery log before it propagates the decision to any of its subordinates. The log-commit record includes the transaction identifier, the commit decision, and the list of subordinates. After having written this log-commit record, the node is in the DECIDED state.

   If the decision was to rollback, no logging is required.

   After the node has received a complete report on the state of the bound data within the transaction tree, the node may remove from the recovery log all information concerning this transaction.

If the node is not the commitment coordinator, its states can be one of the following:

a) ACTIVE: The processing of the transaction is going on. The node can choose to order rollback of the transaction and release bound data under its responsibility without threatening their consistency.

According to the presumed rollback approach, the node is not required to log in the recovery log the creation of any transaction branch;

After it has been invited by the commitment coordinator to enter commitment phase 1, the node may decide to heuristically commit or rollback its bound data. The node taking a heuristic decision writes a log-heuristic record in the recovery log. The log-heuristic record includes the transaction identifier and the decision (commit or rollback). Although the node made a heuristic decision, it still remains in the ACTIVE state; or

b) READY: The node has been invited by the commitment coordinator to indicate whether the bound data of its complete subtree, i.e. the bound data under its own responsibility and under the responsibility of its subordinates, can be put into their final state (committed) or into their initial state (rolled back).

Before indicating that the complete subtree can be committed, the node shall write a log-ready record in the recovery log. The log-ready record includes the transaction identifier, the ready vote, the identification of the superior, and the list of subordinates. After having written this log-ready record, the node is in the READY state.

A node in the READY state may decide to heuristically commit or rollback its bound data. Although the node made a heuristic decision, it still remains in the READY state; or

c) DECIDED: The node has been ordered by its superior to put its bound data into either their final or their initial state.

If the decision was to commit, the node optionally can write a log-commit record in the recovery log. The log-commit record includes the transaction identifier, the commit decision, and the list of subordinates.

Note – There is no need for an intermediate or a leaf node to log this decision in the recovery log; however, in doing so, it may gain performance in the recovery procedure.

The node then propagates the decision to its subordinates. After all subordinates have reported the state of their bound data within their subtree, the node may remove all information concerning this transaction from recovery log, except for the log-damage record. The node then reports the state of the bound data within its subtree to the superior.

8.7.4 *Phases of recovery*

8.7.4.1 *Overview*

Recovery in OSI TP can be divided into three distinct steps called recovery phases:

a) failure detection and containment;

b) transaction recovery; and

c) dialogue recovery.

The objective of each phase is independent of the state of the node, but the type of recovery action taken within each recovery phase is dependent on the state of the node. The sequence of recovery phases is illustrated by Figure 3/X.860.

Phase 1 is entered upon failure detection; recovery is initiated. From the TPSP perspective, communication on some branch of the transaction tree is impossible. This phase attempts to limit the cost of failure, i.e. the time scarce resources are tied up unproductively.

Phase 2 is entered to recover or restart failed components of the transaction. Once the failed components are restarted, activities are initiated to determine whether bound data are in a consistent state and, if not, restore them to a consistent state; the type of recovery activity initiated is dependent on the state of the node.

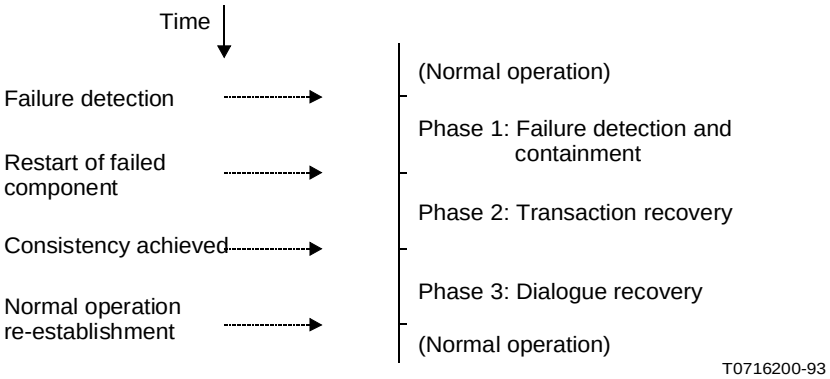


FIGURE 3/X.860

Phases of recovery

Phase 3 may only be entered after either:

a) recovery phase 2 has been completed successfully; or

b) a communication failure while the node was in the ACTIVE state. Presumed rollback ensures that the state of the bound data is consistent.

The dialogue that existed at the time of failure is re-established with the TPSUIs, provided that the TPSUIs still exist. Phase 3's goal is to permit resumption of normal operations.

Note – The provision for dialogue recovery is a candidate for further standardization as an amendment.

Tables 3/X.860 and 4/X.860 summarize the recovery actions taken by a superior node or a subordinate node during recovery phases 1 and 2.

8.7.4.2 *Phase 1: Failure detection and containment*

This phase is entered as a result of the failures which types are listed in Table 2/X.860.

For all failure types, except for atomic action data unavailability, the current state of the node determines what recovery action, if any, should be taken.

If atomic action data becomes unavailable, the state of the node is recovered from the log records; then, the state of the node determines what recovery action, if any, should be taken.

The state of the node which crashed is restored as follows:

a) READY if a log-ready record is available for this transaction; or

b) DECIDED if a log-commit record is available for this transaction. In this case, the outcome of the transaction is to commit; or

c) transaction forgotten if no log record is found. Table 5/X.860 summarizes the restoration of the node state after atomic action data has become unavailable.

Note – Presence of a log-heuristic record does not affect restoration of the node state.

Recovery actions:

a) ACTIVE state: The node brings its bound data to the initial state and propagates rollback to all other nodes with which it is in communication, if any.

When rollback is complete, the node forgets the transaction, and recovery phase 1 terminates. Then, either recovery terminates, or recovery phase 3 is entered.

b) READY state: The node may have taken a heuristic decision before the failure occurs. In this case, there is no particular action to take: a log-heuristic record has already been written.

Alternatively, the node may take a heuristic decision, in which case, the node writes a log-heuristic record.

Recovery phase 2 is entered.

c) DECIDED state: For the part of the transaction tree that has not been affected by the failure, the node behaves normally, as discussed in 8.6.

For the part of the transaction tree that has been affected by the failure, recovery phase 1 terminates, and recovery phase 2 is entered.

TABLE 3/X.860

Recovery actions by a superior node

Recovery phases	NODE STATES			Transaction forgotten (Note)
	ACTIVE	READY	DECIDED	
Phase 1	a) rollback remaining tree; and b) forget transaction.	The node may have previously taken a heuristic decision: no recovery action If the node takes a heuristic decision, a log-heuristic record is written	No recovery action	No recovery action
Phase 2	No recovery action	On a channel: a) receive inquiry from subordinate; b) either postpone reply or invite to retry later.	On a channel: a) propagate final decision within subtree; b) get heuristic information, if any; report to superior, if any; and c) forget transaction.	On a channel: a) receive inquiry from subordinate; b) reply unknown transaction.

Note – "Transaction forgotten" is not a node state. "Transaction forgotten" relates to actions taken by the CPM when information regarding the transaction no longer exists, except possibly a log-heuristic record or log-damage record.

8.7.4.3 *Phase 2: Transaction recovery*

This phase is entered after recovery phase 1 has terminated. Recovery phase 2 is entered when communication with an adjacent node has been disrupted, and the final outcome of the transaction needs to be communicated. There are only two situations that cause the node to re-establish communication for recovery purposes:

a) with the superior node if the node is in the READY state; or

b) with a subordinate node if the node is in the DECIDED state, the outcome of the transaction was to commit, and communication was disrupted with the subordinate before reporting of the state of the bound data within the subordinate's subtree is complete.

Communication is re-established by means of a channel.

TABLE 4/X.860

Recovery actions by a subordinate node

Recovery phases	NODE STATES			Transaction forgotten (Note)
	ACTIVE	READY	DECIDED	
Phase 1	a) rollback subtree; and b) forget transaction.	The node may have previously taken a heuristic decision: no recovery action If the node takes a heuristic decision, a log-heuristic record is written	No recovery action	No recovery action
Phase 2	No recovery action	On a channel: a) get final decision from superior; b) propagate final decision within subtree; c) get heuristic information, if any; report to superior, if any; and d) forget transaction.	On a channel: a) receive inquiry from superior; and b) report to the superior.	On a channel: a) receive inquiry from superior; and b) reply that the transaction was committed

Note – "Transaction forgotten" is not a node state. "Transaction forgotten" relates to actions taken by the CPM when information regarding the transaction no longer exists, except possibly a log-heuristic record or log-damage record.

TABLE 5/X.860

Restoration of node state after atomic action data unavailability

Type of log record	No log record	Log-ready record	Log-commit record
Node state	Transaction forgotten	READY	DECIDED

The current state of the node determines what recovery action, if any, should be taken.

Recovery actions:

a) ACTIVE state: The node never enters recovery phase 2 while in the ACTIVE state, according to the presumed rollback paradigm, there is no need to communicate the outcome of the transaction.

b) READY state: If communication failed with a subordinate node, no recovery action is required. If communication failed with the superior node, the node

   1) re-establishes communication with the superior node; and

2) issues a question to the superior node about the outcome of the transaction. Upon receipt of the superior's response, carrying the final outcome of the transaction, the node enters the DECIDED state, and

– according to presumed rollback, if the superior indicates that it has no knowledge of the transaction, then the node releases its bound data in the initial state, unless a heuristic decision was taken. The node also propagates rollback to all its subordinates the node is in communication with, if any.

The node forgets the transaction, and recovery phase 2 terminates. Then, either recovery terminates, or recovery phase 3 is entered;

– if the superior replied that the outcome of the transaction was to commit, the node releases its bound data in the final state, unless a heuristic decision was taken. The node also propagates commit to all its subordinates the node is in communication with, if any. For subordinate nodes with which communication was disrupted, the node behaves as subsequently discussed by item c), DECIDED state.

The node forgets the transaction, and recovery phase 2 terminates when the node has reported to its superior the state of the bound data within its subtree. Then, either recovery terminates, or recovery phase 3 is entered;

c) DECIDED state: If communication failed with a superior node, no recovery action is required. If communication failed with a subordinate node, and the outcome of the transaction was to commit, the node

1) re-establishes communication with the subordinate node; and

2) propagates commit to the subordinate node.

Recovery phase 2 terminates when the node has received a report on the state of the bound data within the subordinate's subtree. Then, either recovery terminates, or recovery phase 3 is entered; or

d) if the node has forgotten the transaction, no recovery action is required by the node. However, the node enters recovery phase 2 upon request from either a subordinate or a superior node, if any.

If the request originates from the superior node, the node reports (to the superior node) the state of the bound data within its subtree.

Note – The node takes into account the presence of any log-damage record when reporting the state of bound data within its subtree.

If the request originates from a subordinate node, the node replies that the outcome of the transaction was to rollback.

8.7.4.4 *Phase 3: Dialogue recovery*

Note – The provision of dialogue recovery is for further study.

8.8 *Concurrency control and deadlock*

The concurrency control mechanisms implemented on different open systems are outside the scope of the OSI TP specifications. Taking into account the fact that there are concurrency control mechanisms which do not exclude the occurrence of deadlocks involving resources on multiple open systems (global deadlocks), it is assumed that such deadlocks are avoided or detected.

Note – One means is the use of timers which are associated with the distributed transactions: if a transaction does not obtain a requested lock within the timeout interval, the transaction will be rolled back because deadlock is presumed.

8.9 *Security*

Note – The provision for security is for further study.

ANNEX A

(to Recommendation X.860)

Relationship of the OSI TP Model to the Application Layer Structure

(This annex forms an integral part of this Recommendation)

A.1 *Introduction*

This clause describes the basic structure of an application-process-invocation (API) with an application-entity-invocation (AEI) supporting a TPSUI.

A.2 *Application-processes within OSI TP*

TPSUs are part of application-processes. One or more TPSUs may exist within an application-process.

A TPSU makes use of the OSI TP Service to achieve OSI TP communication. It performs two distinct application-specific tasks:

a) processing, to achieve part of the transaction; and

b) communication to interact with partner TPSUs.

The OSI TP Service is available within the AEI.

Within an application-process, TPSU application-specific communication requirements are met by one or more User-ASEs within one or more SAOs.

Semantics for selecting TPSUs within an application-process are conveyed through the OSIE by OSI TP Protocol, using Application Layer addressing together with specific titles and identifiers defined by this Recommendation.

A.3 *Application entities with OSI TP*

A.3.1 *AE invocations with OSI TP*

A communicating AEI involved in OSI TP includes one or more SAOs. That is, a TPSUI may be simultaneously utilizing several SAOs. This does not imply a many-to-many relationship between SAOs and TPSUIs: a given SAO can only be used by one TPSUI at any one time.

A.3.2 *The MACF within OSI TP*

An application-entity supporting OSI TP always includes a MACF.

Within OSI TP, the MACF is the component of the TPPM which coordinates the interactions over multiple associations within an AEI in order to provide the OSI TP Service. Its functions include

a) allocation of application-associations for use and re-use by dialogues and/or channels;

b) establishment and termination of the binding between a TPSUI and one or more appropriate SAOs; and

c) coordination of activities over multiple application-associations to ensure the ACID properties of transactions. In particular, the MACF must

   1) cause the generation of protocol over one or more individual associations, as required in support of the OSI TP Service;

   2) have logged in secure storage the atomic action data required in support of recovery, and the information required to coordinate recovery of the user bound data associated with the transactions; and

   3) coordinate recovery mechanisms after an application or communications failure for each SAO within the AEI on behalf of a TPSUI.

A.3.3 *The SAOs within OSI TP*

Each SAO includes one or more single association User-ASEs that support TPSUI application-specific communication, and the single association aspects of the TPPM.

In the context of OSI TP, CCR is included only when the coordination level of "commitment" is required. If included, CCR is invoked by the TPPM only.

Within the bounds of a dialogue, the application-association supporting the dialogue between two TPSUIs is shared by the TPASE, ACSE, one or more User-ASEs, and optionally CCR. User-ASE semantics may only be exchanged in certain states of the TPPM; the OSI TP environment constrains User-ASEs so that the OSI TP rules are observed at both the OSI TP Service and OSI TP Protocol levels.

In each SAO, the SACF models the following functions:

a) the necessary coordination of interactions between single association aspects of the TPPM and other ASEs contained in the SAO, as specified in the application context definition for the association;

b) coordination of the use of the Presentation Service by the individual components of the SAO; and

c) concatenation and separation of APDUs, as appropriate.

An application context definition describes how component specifications are incorporated in the TP environment.

A.4 *OSI TP Service boundary*

The overall functionality of the OSI TP Service is provided via the MACF (for example, in the translation of the commitment service (see Recommendation X.861) to individual CCR service primitives in multiple SAOs).

The OSI TP Service is such that the integration of local resources into transaction commitment, as well as the coordination of the OSI TP Service with other application layer services, may be performed by the TPSUI.

The TPSP includes a TPPM for each TPSUI and one CPM per AEI. Each TPPM includes:

a) MACF functionality for OSI TP; and

b) on each association, an SAO comprising:

   1) SACF functionality for OSI TP;

   2) ACSE;

   3) TPASE;

   4) one or more user-ASEs; and

   5) CCR, if commitment processing is required.

A CPM includes:

a) MACF functionality for OSI TP and

b) on each association, an SAO comprising:

   1) SACF functionality for OSI TP;

   2) ACSE;

   3) TPASE; and

   4) CCR.

ANNEX B

(to Recommendation X.860)

Tutorial on concurrency and deadlock control in OSI TP

(This annex does not form an integral part of this Recommendation)

A distributed transaction is defined to be an atomic unit of work that either completely succeeds or fails as a whole. However, it is also desirable for transactions to be executed concurrently within a system and therefore concurrency control mechanisms must be provided to control access to shared resources.

In a distributed system, it is important that the various local resource managers, participating in a transaction, support compatible levels of concurrency control (e.g. both supporting “updates allowed, repeatable reads”).

The mechanisms to assure compatibility of concurrency control schemes are outside the scope of OSI TP (e.g. may be an application matter).

It is a well known problem that when using locking for concurrency control, deadlock situations can occur. This is not a problem for the OSI environment when it occurs within a single system. The system can use whatever local technique it chooses to resolve the deadlock. However, in the OSI environment, a deadlock situation could occur between transactions that are accessing resources on multiple open systems.

There are two basic approaches to deadlock control which are appropriate for OSI TP systems: deadlock detection, and deadlock avoidance.

In deadlock detection, the system waits until a deadlock exists. Deadlock detection algorithms typically use a wait-for graph. A wait-for graph is a directed graph which indicates which transactions are waiting for which other transactions. In the distributed processing environment, the local wait-for graphs must be combined to form a global wait-for graph.

The main disadvantage of deadlock detection is the additional overhead incurred in maintaining the wait-for graphs and detecting cycles in the wait-for graphs.

In deadlock avoidance, deadlock is avoided by aborting transactions in situations that can lead to a deadlock. Transactions are allowed to proceed unless a requested resource is unavailable. If a resource is not available, either the holding transaction or the requesting transaction may be aborted. The victim selection criteria depend on the avoidance scheme used.

One scheme is deadlock timeout. Associated with each transaction is a maximum time that the transaction will wait to obtain a resource lock.

Suppose that a transaction T1 requires a lock on resource R1. The deadlock timeout algorithm is implemented as follows:

Deadlock timeout

If transaction T1 obtains the lock on R1 within the deadlock timeout interval, then transaction T1 continues processing.

If transaction T1 does not obtain the lock on R1 within the deadlock timeout interval, then transaction T1 aborts.

Deadlock avoidance may be better in some systems for at least two reasons. First, if deadlocks are very rare, the extra cost of detection (in terms of system code, etc.) might not be justified. Another possibility is that detection could cause system bottle-necks (such as traffic locks) to become overly congested during the time it takes to detect and resolve a deadlock.

There are many deadlock detection and deadlock avoidance algorithms. Some of these can be used in distributed environments. There are two classes of these algorithms:

a) precise: Those that always detect real deadlocks and only real deadlocks;

b) imprecise: Those that always detect real deadlocks, but sometimes also report correct operations as deadlocks.

To date, there are no well understood “precise” deadlock algorithms that work in heterogeneous environments with good communication efficiency. As a consequence, local deadlock detection via timers, an “imprecise” mechanism, is assumed.

ANNEX C

(to Recommendation X.860)

Tutorial on the presumed ROLLBACK two-phase COMMIT protocol

(This annex does not form an integral part of this Recommendation)

A distributed transaction has the characteristics that there are several participants in the transaction that need to perform work in committing the transaction. This work has to be performed in an orderly fashion. It is therefore grouped into two phases.

In phase I, the participants record changes to protected resources in secure storage, to guarantee the “A”,“C” and “D” of the ACID properties. If all participants are successful, the second phase is entered by recording that the transaction has committed. In the second phase, all resources held by the transaction are released. This is to guarantee the “I” of the ACID properties.

There is a commitment coordinator that is responsible for determining that phase I has successfully completed. The commitment coordinator does this by collecting “ready” votes from the participants. If a READY consensus is reached, the transaction completes as COMMITTED. In a transaction tree, a node collects the votes from its subordinates, and votes for itself and its subordinates to its superior.

In presumed rollback, during recovery after a failure, the commitment coordinator is responsible for informing subordinates of the final outcome of the transaction, only if it COMMITTED. If a subordinate has a transaction in READY state after a failure, the subordinate is responsible for asking its superior about the final outcome of the transaction. It then presumes that the transaction should ROLLBACK if the superior has no knowledge of the transaction. The response from the superior could also be COMMIT or retry later. In the case of no failure, the superior is responsible for informing its subordinates whatever the outcome is.

The presumed rollback protocol only requires a commitment coordinator to record its subordinates when the subordinates have voted “ready” and the transaction can COMMIT. This is done at the commitment coordinator node in the log-commit record. An intermediate node may also write a log-commit record, when commanded to commit by its superior, but it does not have to. Keeping such a record is an overhead on each transaction, but speeds recovery in that the final decision can be propagated within the subtree from the node following failure even if communications with its superior are still disrupted. All nodes, except for the commitment coordinator, must record their (immediate) superior in a log-ready record when they have received “ready” votes from all subordinates and are ready themselves, and before voting “ready” to the superior.