INTERNATIONAL STANDARD

ISO/IEC 9804

Third edition
1998-12-01

Information technology — Open Systems Interconnection — Service definition for the Commitment, Concurrency and Recovery service element

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — Définition du service pour l'élément d'engagement, de concomitance et de rétablissement



Reference number
ISO/IEC 9804:1998(E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Contents

Page

1 Scope 1
2 Normative references 1
2.1 Identical Recommendations | International Standards 1
2.2 Paired Recommendations | International Standards equivalent in technical content 2
3 Definitions 2
3.1 Reference Model definitions 2
3.2 Service conventions definitions 3
3.3 Presentation service definitions 3
3.4 ACSE service definitions 3
3.5 Application Layer Structure definitions 3
3.6 CCR service definitions 4
4 Abbreviations 7
5 Conventions 8
6 Concepts 8
6.1 Use of CCR in a distributed application environment 8
6.2 CCR facilities 14
6.3 Heuristic decisions 16
7 Service definition 16
7.1 C-INITIALIZE service 17
7.2 C-BEGIN service 18
7.3 C-PREPARE service 19
7.4 C-READY service 20
7.5 C-COMMIT service 20
7.6 C-ROLLBACK service 21
7.7 C-NOCHANGE service 22
7.8 C-CANCEL service 23
7.9 C-RECOVER service 23
7.10 C-P-ERROR service 25
8 Sequencing information 25
8.1 General 25
8.2 Events 32
8.3 States 32
8.4 Predicates 32
8.5 Interpretation of the state table 32

© ISO/IEC 1998

All rights reserved. Unless otherwise specified, 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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.6 Completing the branch 32
8.7 Collisions and disruptive services 33
9 Using CCR 33
9.1 General 33
9.2 Use of CCR with non-reference mapping 33
9.3 Use of session synchronization and resynchronization services 33
9.4 Use of CCR with session activities 33
9.5 Use of presentation services 33
Annex A – CCR service-user rules 34
A.1 Introduction 34
A.2 Compliance 34
A.3 CCR service primitive usage rules 34
A.4 Atomic action data manipulation rules 37
A.5 Bound data manipulation rules 38
A.6 CCR service-user data transfer rules 39
Annex B – Relationship of CCR to the Application Layer Structure 40
B.1 CCR service-provider 40
B.2 CCR service-user 40
B.3 Atomic action graph 40
Annex C – CCR tutorial 42
C.1 Introduction 42
C.2 Strucutre of an atomic action tree 43
C.3 CCR service-user information resources 45
C.4 Concurrency 47
C.5 Recovery 48
C.6 Time relations and sequence of service primitives 52
C.7 Comments on implementation complexity 53
C.8 Using the User Data parameter on CCR services 53
C.9 Optional use of C-PREPARE 59

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

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 9804 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 33, *Distributed application services*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.851.

This third edition cancels and replaces the second edition (ISO/IEC 9804:1994), which has been technically revised.

Annexes A and B form an integral part of this International Standard. Annex C is for information only.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Introduction

This Recommendation | International Standard is one of a set of Recommendations | International Standards produced to facilitate the interconnection of information processing systems. It relates to other ITU-T Recommendations | International Standards in the set defined by the Reference Model for Open Systems Interconnection (see ITU-T Rec. X.200 | ISO/IEC 7498). The reference model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

The goal of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection Recommendations and International Standards, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different technologies.

This Recommendation | International Standard recognizes that application-processes may wish to communicate with each other for a wide variety of reasons. However, any communication requires certain services independent of the reasons for communication. The application-service-element defined in this Recommendation | International Standard provides such services.

This Recommendation | International Standard defines the facilities of the application-service-element for Commitment, Concurrency and Recovery (CCR). CCR provides services for a single association. A referencing specification uses these services for starting and ending a specific sequence of distributed application operations despite application or communication failure.

This Recommendation | International Standard is referenced by a specification to apply CCR to its operation. CCR services may be used with presentation services (see ITU-T Rec. X.216 | ISO/IEC 8822), or with other Application Layer services. However, the use of CCR services is subject to the restrictions specified in clause 9. The use of CCR services allows a referencing specification to define its activity as an atomic action. An atomic action may use many associations, possibly with different protocols on each association.

Annex A describes the rules that shall be followed by a specification that references this Recommendation | International Standard.

Annex B presents the relationship of the CCR model and concepts to the Application Layer Structure (see ITU-T Rec. X.207 | ISO/IEC 9545).

Annex C is a tutorial to aid the understanding of the concepts and facilities of CCR.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SERVICE DEFINITION FOR THE COMMITMENT, CONCURRENCY AND RECOVERY SERVICE ELEMENT

1 Scope

This Recommendation | International Standard is intended for reference by other specifications when the functionality of commitment, concurrency and recovery is required. It can be referenced whenever the processing of two or more application-entity invocations in a distributed application needs to be organized into an atomic action.

This Recommendation | International Standard defines services that are used on a single association to coordinate two application-entity invocations involved in an atomic action. The determination of which application-entity invocations are involved in an atomic action is not within the scope of this Recommendation | International Standard.

This Recommendation | International Standard establishes the general principles for the coordinated use of the CCR services when more than two application-entity invocations are involved in a single atomic action, or when recovery is required after failure. The coordination of multiple associations and the related application-entity invocations that constitute an atomic action is achieved by a referencing specification in conjunction with this Recommendation | International Standard.

This Recommendation | International Standard is only applicable to a distributed application whose specification references this Recommendation | International Standard.

This Recommendation | International Standard does not specify individual implementations or products. It does not constrain the implementation of entities and interfaces within a computer system.

No requirement is made for conformance to this Recommendation | International Standard.

This Recommendation | International Standard includes requirements for compliance that apply to a referencing specification.

The CCR service defined in this Recommendation | International Standard requires that CCR Protocol Version 2 (or a later version) is being used.

2 Normative references

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

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*
- ITU-T Recommendation X.207 (1993) | ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application layer structure*.
- ITU-T Recommendation X.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model : Conventions for the definition of OSI services*.
- ITU-T Recommendation X.215 (1995) | ISO/IEC 8326:1996, *Information technology – Open Systems Interconnection – Session service definition*.

-`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*.
- ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the association control service element*.
- ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the association control service element: Protocol specification*.
- ITU-T Recommendation X.650 (1996) | ISO/IEC 7498-3:1997, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*.
- ITU-T Recommendation X.852 (1997) | ISO/IEC 9805-1:1998, *Information technology – Open Systems Interconnection – Protocol for the commitment, concurrency and recovery service element: Protocol specification*.

2.2 Paired Recommendations | International Standards equivalent in technical content

- ITU-T Recommendation X.860 (1997), *Open Systems Interconnection – Distributed transaction processing: Model*.

  ISO/IEC 10026-1[1)], *Information technology – Open Systems Interconnection – Distributed Transaction Processing – Part 1: OSI TP Model*.
- ITU-T Recommendation X.862 (1997), *Open Systems Interconnection – Distributed transaction processing: Protocol Specification*.

  ISO/IEC 10026-3[1)], *Information technology – Open Systems Interconnection – Distributed Transaction Processing – Part 3: Protocol specification*.

3 Definitions

3.1 Reference Model definitions

3.1.1 Basic Reference Model definitions

This Recommendation | International Standard is based on the concepts developed in ITU-T Rec. X.200 | ISO/IEC 7498-1. It makes use of the following terms defined in them:

a) application-association; association;

b) application-entity;

c) Application Layer;

d) application-process;

e) application-service-element;

f) presentation-connection;

g) presentation-service;

h) session-connection;

i) session-service.

3.1.2 Naming and addressing definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.650 | ISO/IEC 7498-3:

- application-entity title[2)].

---

1) To be published.

2) As defined in ITU-T Rec. X.650 | ISO 7498-3, an application-entity title is composed of an application-process title and an application-entity qualifier.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.2 Service conventions definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.210 | ISO/IEC 10731:

a) service-provider;

b) service-user;

c) confirmed service;

d) non-confirmed service;

e) provider-initiated service;

f) primitive;

g) request (primitive);

h) indication (primitive);

i) response (primitive); and

j) confirm (primitive).

3.3 Presentation service definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.216 | ISO/IEC 8822:

a) abstract syntax;

b) abstract syntax name;

c) defined context set;

d) functional unit [presentation];

e) presentation context; and

f) presentation data value.

3.4 ACSE service definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.217 | ISO/IEC 8649:

a) association-initiator;

b) association-responder; and

c) disrupt.

3.5 Application Layer Structure definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.207 | ISO/IEC 9545:

a) application-context;

b) application-entity invocation;

c) application-service-object;

d) control function;

e) multiple association control function;

f) single association control function;

g) single association object.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.6 CCR service definitions

3.6.1 acceptor: The CCR service-user that receives the indication primitive for a particular CCR service. For a confirmed service, it also issues the response primitive.

3.6.2 application failure: The failure of an application-entity invocation to meet its normal specification.

3.6.3 atomic action: A specific set of operations of a distributed application that may be characterized by the properties of atomicity, consistency, isolation, and durability.

3.6.4 atomic action branch; branch: A relationship between two CCR service-users representing an integral part of an atomic action. The relationship may survive both communication or application failure. It is begun by the use of CCR services and later completed by either the use of CCR services or by an application or communication failure.

3.6.5 atomic action branch identifier; branch identifier: A value assigned by the atomic action branch-initiator that uniquely identifies a branch within the scope of the atomic action.

3.6.6 atomic action data: State and control information about an atomic action and its branches. Atomic action data required for recovery persists if an application or communication failure occurs.

3.6.7 atomic action graph: A connected graph consisting of CCR service-users as nodes and atomic action branches as arcs that represents the structure of an atomic action.

3.6.8 atomic action identifier: A value assigned by the atomic action owner that uniquely identifies an atomic action within the OSI environment. (The value is first used in a CCR service by the atomic action initiator. However, the initiator may have received the value from another source via a mechanism that is not visible in CCR services).

3.6.9 atomic action initiator: The begin-tree root.

3.6.10 atomic action owner: The CCR service-user that established the atomic action identifier.

3.6.11 atomicity: A property of a set of related operations such that the operations are either all performed, or none of them are performed.

3.6.12 begin-tree; atomic action begin-tree: An atomic action graph that has been formed into a rooted tree where the direction of an arc is from the CCR service-user that initiates the atomic action branch.

3.6.13 bound data: Data that are accessed and manipulated by a CCR service-user as part of an atomic action. Their state is bound by the rules of CCR. Bound data survive application and communication failures and exist beyond the atomic action branch.

3.6.14 branch-initiator; atomic action branch-initiator: The CCR service-user that begins a specific branch.

3.6.15 branch-responder; atomic action branch-responder: On a specific branch, the CCR service-user that did not initiate the branch.

3.6.16 commit-tree; atomic action commit-tree: An atomic action graph that has been formed into a rooted tree in which the direction of an arc is from the CCR service-user (the commit-superior) that may order commitment to the peer (the commit-subordinate).

3.6.17 CCR service-provider: Two peer CCR application-service-elements involved in the same atomic action branch.

3.6.18 CCR service-user: That part of an application-entity invocation that makes use of CCR services to coordinate one or more branches of an atomic action graph.

3.6.19 commit coordinator: A CCR service-user that receives ready signals from all of its neighbours.

3.6.20 commit-decider: A CCR service-user that orders commitment to (usually, all of) its neighbours, without having received an order of commitment. It is the root of the commit-tree (In certain cases, one of two roots).

3.6.21 commitment of an atomic action branch; commitment: Completion of an atomic action branch with the release of bound data in the final state.

3.6.22 commit-subordinate: (With reference to a branch.) The CCR service-user that sends a ready signal to its neighbour; (with reference to a particular CCR service-user) another CCR service-user from whom a ready signal has been received on any branch (there may be several commit-subordinates for one CCR service-user).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.6.23 commit-superior: (With reference to a branch.) The CCR service-user that receives a ready signal from its neighbour; (with reference to a particular CCR service-user) another CCR service-user to whom a ready signal has been sent (CCR ensures there can be at most one).

3.6.24 communication failure: The unexpected release of the supporting association.

3.6.25 compensating action: Operations used to re-establish either the initial or the final state from a mixed situation that was brought about by a conflict between heuristic decision(s) and the decision of the commit coordinator.

3.6.26 concurrency control: A real open system mechanism that coordinates modifications to bound data used by concurrent atomic actions so the isolation property of the atomic action is guaranteed.

3.6.27 confirmation of commitment: A statement from a commit-subordinate to the commit-superior that the commit-subordinate has completed local commitment procedures.

3.6.28 continuing two-phase branch; continuing two-phase neighbour: A branch/neighbour in an atomic action except any

i) which have been rolled back (by C-ROLLBACK request or indication); or

ii) which the CCR service-user has determined will be rolled-back, but has not done so; or

iii) on which C-NOCHANGE indication has been received.

NOTE – ii) includes branches where the supporting association has failed prior to a ready signal, as well as branches to which C-ROLLBACK request is about to be issued.

3.6.29 connected graph: A graph that consists of a set of nodes and a set of arcs. Two nodes may be connected by an arc. Each arc connects two nodes. The terms "node" and "arc" are used here in the normal mathematical sense.

3.6.30 consistency: A property of a set of related operations such that the effects of the operations are performed accurately, correctly, and with validity, with respect to application semantics.

3.6.31 distributed application: An information processing endeavour that is accomplished using two or more application-entity invocations interconnected within the OSI environment.

NOTE – This term will be removed from this subclause when its definition becomes available in another referenced Recommendation | International Standard.

3.6.32 doubt period: For a CCR service-user, the period during an atomic action that begins when it decides to send a ready signal to its superior and ends when it receives either the order to commit or to rollback. A CCR service-user that does not send a ready signal does not have a doubt period.

3.6.33 durability: A property of a completed set of related operations such that all the effects of the operations are not altered by any sort of failure.

3.6.34 final state: The state of bound data produced as a result of the completed application operations of the atomic action.

3.6.35 graph: An object that consists of a set of nodes and a set of arcs. Two nodes may be connected by an arc. Each arc connects two nodes.

NOTE – As used in this Recommendation | International Standard, "graphs" are always acyclic and connected, although this is not a general property of graphs. See also the definition of "tree" below.

3.6.36 heuristic decision: A decision of a CCR service-user that has sent a ready signal to the commit-superior and then releases all or part of its bound data before it is ordered to commit or to roll back by the commit-superior.

3.6.37 initial state: The state of bound data at the time of first use by an atomic action.

3.6.38 intermediate: A node in a rooted tree that is neither a leaf nor the root. An intermediate always has precisely one incoming arc.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.6.39 intermediate state: One of the states of bound data produced during the manipulation of bound data that is neither the initial nor the final state.

3.6.40 interrupted branch: An atomic action branch whose supporting association was normally or abnormally released because of an application or communication failure.

3.6.41 isolation: A property of a set of related operations such that partial results of the set of operations are not accessible, except by operations of the set. This definition implies that different sets of related operations that have this property and that share bound data are serializable.

3.6.42 leaf: A node in a graph that only has one arc. In a rooted tree, the term is restricted to nodes with only one incoming arc. Thus, in a rooted tree, the root is never considered to be a leaf.

3.6.43 local commitment procedures: Establishing the final state of all bound data, removal of concurrency controls, and release of all resources used in performing the atomic action.

3.6.44 local rollback procedures: Re-establishing the initial state of all bound data, removal of concurrency controls, and release of all resources used in performing the atomic action.

3.6.45 mixed heuristic situation; mixed situation: The state of bound data produced as the result of heuristic decision(s) when a CCR service-user releases bound data in a state different from the commit coordinator.

3.6.46 neighbour (of a node in a graph): A node within a connected graph that has an arc in common with this node. For CCR, the logically adjacent CCR service-user directly connected by an atomic action branch.

3.6.47 neighbourhood (of a node): The connected part of a tree that consists of a neighbour of the node and all the nodes in the tree that are disconnected from this node (i.e. they do not have a path to the node) when the neighbour is removed from the tree.

3.6.48 node; CCR node: A CCR service-user for a particular atomic action.

3.6.49 non-reference mapping: Any mapping of CCR services to the ACSE and Presentation service other than that specified in the body of ITU-T Rec. X.852 | ISO/IEC 9805-1. Annex B of ITU-T Rec. X.852 | ISO/IEC 9805-1 specifies constraints on such mappings.

3.6.50 order of commitment of an atomic action branch; order of commitment: A statement by a CCR service-user to a neighbour that has given a ready signal that the atomic action branch is committed.

3.6.51 phase I: For a CCR service-user that sends a ready signal, the period during an atomic action that ends when it decides to send a ready signal to its superior. For a CCR service-user that does not send a ready, phase I ends when, and if, it decides to commit the atomic action (i.e. when a commit coordinator becomes commit-decider). This Recommendation | International Standard does not specify when phase I starts.

3.6.52 phase II: For a CCR service-user that is not the commit-decider, the period during an atomic action that begins when it is ordered to commit by its commit-superior. For the commit-decider CCR service-user, phase II begins when it decides to commit the atomic action. Phase II ends for any CCR service-user when it completes all of its branches and its involvement with the atomic action ends.

3.6.53 presumed rollback: The recovery mechanism used by CCR. It conditionally allows a CCR service-user to treat an application or communication failure as a rollback. This occurs if it has not recorded atomic action data for the branch. In addition, a CCR service-user acting as a commit-subordinate may presume rollback under the following condition. It has recorded atomic action data for the branch but, during recovery, it discovers that the commit-superior does not.

3.6.54 ready-to-commit state: A state of bound data in which, until the atomic action has been terminated by commitment or rollback, the bound data can be released in either their initial or their final state.

3.6.55 reference mapping: The mapping of CCR services to the ACSE and Presentation service specified in the body of ITU-T Rec. X.852 | ISO/IEC 9805-1.

3.6.56 recovery of an atomic action branch; recovery: Procedures used by a CCR service-user to complete an interrupted atomic action branch for which it has recovery responsibility.

3.6.57 recovery responsibility for an atomic action branch; recovery responsibility: A property of a CCR service-user that determines whether it attempts recovery. The CCR service-user acquires this property as a result of using certain CCR services. It retains the property until the completion of the atomic action branch.

3.6.58 referencing specification: An Application Layer Recommendation | International Standard or other specification that specifies the use of CCR services. CCR services are always used in conjunction with a referencing specification.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.6.59 requesting neighbourhood: With reference to some request primitive, the neighbourhood including the requesting CCR service-user and excluding the accepting CCR service-user.

3.6.60 requestor: The CCR service-user that issues the request primitive for a particular CCR service. For a confirmed service, it also receives the confirm primitive.

3.6.61 rollback of an atomic action branch; rollback: Completion of an atomic action branch with the release of bound data in the initial state.

3.6.62 ready signal: A statement from a CCR service-user to a neighbour that indicates the CCR service-user's readiness to commit

3.6.63 subordinate of an arc: The node at the incoming end of an arc.

3.6.64 subordinate of a node: The node at the other end of an outgoing arc. A node may have zero, one, or more subordinates.

3.6.65 subordinate subtree: The subtree of a subordinate node.

3.6.66 subtree (of a node): A tree (within another tree) with the node as the root. A leaf node is its own subtree.

3.6.67 superior of an arc (in a rooted tree): The node from which the arc is outgoing.

3.6.68 superior of a node (in a rooted tree): The node at the other end of the single incoming arc. The root of the tree has no superior. Otherwise, a node has exactly one superior.

3.6.69 root: The single node in a tree that has outgoing arcs only.

3.6.70 tree: An acyclic, connected graph whose arcs are oriented from a single node with outgoing arcs only. The nodes of a tree are arranged in a hierarchical structure according to the orientation of the arcs. The terms "node" and "arc" are used here in the normal mathematical sense.

NOTE – In at least some mathematical uses, a tree is any acyclic, connected graph without any necessary orientation of the arcs. In this Recommendation | International Standard, "tree" is distinguished from the more general "graph" by the presence of some orientation of the arcs.

3.6.71 user-ASE: An application-specific ASE.

4 Abbreviations

For the purposes of this Recommendation | International Standard the following abbreviations apply:

ACSE	Association Control Service Element
AE	Application-entity
AEI	Application-entity-invocation
Amd	Amendment to an ITU-T Recommendation | ISO/IEC International Standard
ASE	Application-service-element
ASO	application-service-object
CCR	Commitment, Concurrency, and Recovery application-service-element
CCR-sp	Commitment, Concurrency, and Recovery service-provider
cnf	Confirm primitive
ind	Indication primitive
MACF	Multiple Association Control Function
OSI	Open Systems Interconnection
OSIE	Open Systems Interconnection Environment
req	Request primitive
rsp	Response primitive
SACF	Single Association Control Function
SAO	Single Association Object
U-ASE	User application-service-element

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5 Conventions

This Recommendation | International Standard defines services for CCR following the descriptive conventions defined in ITU-T Rec. X.210 | ISO/IEC 10731.

In clause 7, the definition of each CCR service includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values:

Blank	Not applicable
C	Conditional
M	Mandatory
U	User option

In these tables, the notation (=) indicates that a parameter value is semantically equal to the value to its left in the table.

6 Concepts

6.1 Use of CCR in a distributed application environment

CCR services are defined for a single association. They are not concerned with and do not address the organization and topology of a distributed application. A referencing specification is always required to coordinate the use of CCR services. However, the use of CCR services requires an understanding of the distributed application environment.

6.1.1 Atomic action environment

6.1.1.1 Atomic action properties

An atomic action is a specific set of related distributed application operations that may be characterized by the following properties:

a) *Atomicity* – A property of a set of related operations such that the operations are either all performed or none of them are performed.

b) *Consistency* – A property of a related set of operations such that the effect of the operations are performed accurately, correctly and with validity, with respect to application semantics.

c) *Isolation* – A property of a set of related operations such that partial results are not accessible, except by operations of the set. This definition implies that different sets of related operations that have this property and that share bound data are serializable.

d) *Durability* – A property of a set of related operations such that all the effects of the operations are not altered by any sort of failure.

In the ideal case, all these atomic action properties are maintained by the CCR service-user. However, the degree of achievement of these properties depends on the level of compliance to the CCR service-user rules (see Annex A) and the local strategies of the CCR service-users.

Taking heuristic decisions is an example of a local strategy that might violate the atomic action properties (see 6.3). Heuristic decisions do not guarantee atomicity of the atomic action. Another example is the use of a concurrency mechanism that allows intermediate states of bound data to be visible outside the atomic action.

6.1.1.2 The atomic action graph and begin-tree

The CCR service-users that participate in an atomic action form a relationship that is a graph with no closed loops. This is called the **atomic action graph.** An example is shown in Figure 1. An atomic action graph consists of CCR service-users (as nodes) and atomic action branches (as arcs).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

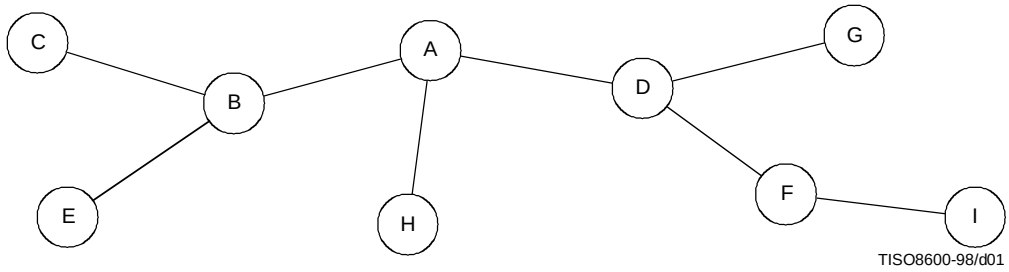

Figure 1 – Atomic action graph

A given AEI can represent one or more CCR service-users of the same or different atomic action graphs.

NOTE 1 – Atomic action branches between CCR service-users in the same AEI are outside the scope of this Recommendation | International Standard.

A **branch** of the atomic action is the relationship between two logically adjacent CCR service-users in the atomic action graph. These CCR service-users are called **neighbours**.

An atomic action graph is dynamically constructed by the formation of its branches. The atomic action graph and its branches only exist for the lifetime of the atomic action.

An atomic action graph starts when a CCR service-user begins the first branch. This CCR service-user is called the **atomic action initiator**. It makes use of an **atomic action identifier** to unambiguously identify the new atomic action within the OSIE. The atomic action identifier can be assigned by the atomic action initiator or some other entity can assign the value and communicate it to the atomic action initiator. This value is propagated throughout the atomic action. A CCR service-user uses it to maintain concurrency controls. Following an application or communication failure, it is used to correlate recovery for interrupted branches of the atomic action.

NOTE 2 – A referencing specification may specify mechanisms for passing the atomic action identifier to the initiator from some other entity that creates and assigns the identifier.

Figure 2 shows the atomic action graph of Figure 1, with the CCR service-users ordered and lettered in the sequence in which they joined the atomic action. CCR service-user A is the atomic action initiator.

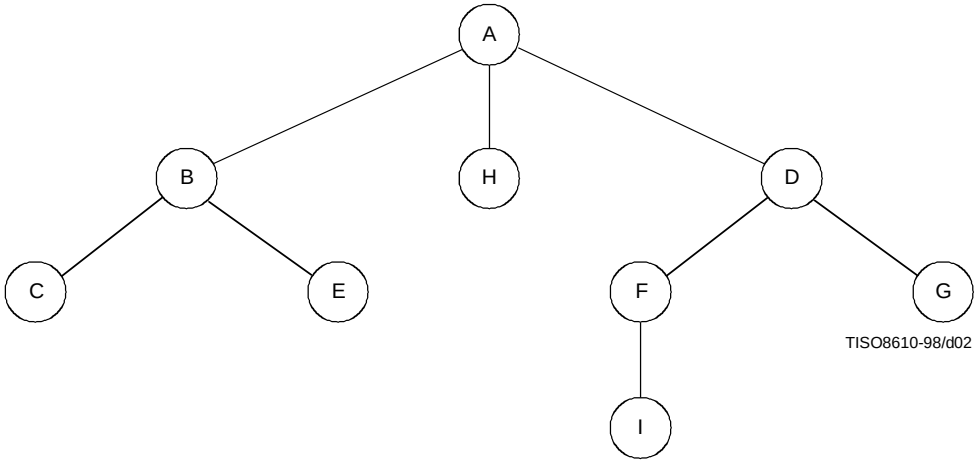

Figure 2 – Atomic action graph ordered according sequence of joining

Based on the requirements of the referencing specification, a CCR service-user can introduce another CCR service-user into the atomic action graph. This adds a new branch to the atomic action graph.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Beginning from any CCR service-user, an atomic action graph can be ordered hierarchically. Such an ordering that begins with the atomic action initiator defines the **atomic action begin-tree.** Figure 2 shows an atomic action graph in this ordering, when the atomic action was started by CCR service-user A. Each branch was started from the end nearer the top of the diagram.

Following failure, the recovery facilities of CCR are used to ensure that branch completion procedures are correctly applied throughout the atomic action. An atomic action graph ends with the completion of all the individual branches.

6.1.2 Atomic action branch

An atomic action branch is a relationship between two neighbours, i.e. logically adjacent CCR service-users. This relationship performs a portion of the work of an atomic action.

Each branch has a **branch-initiator** and a **branch-responder**. The branch-initiator requests the branch with the branch-responder. Within the atomic action begin-tree hierarchy, the branch-responder is one level lower than the branch-initiator.

The branch-initiator uses the appropriate atomic action identifier. It assigns a **branch identifier** whose value is unique within the scope of the atomic action. This branch identifier is used to identify a particular branch of the atomic action graph during recovery following an application or communication failure.

A branch is supported by an association. If an application or communication failure occurs, the branch may endure and continue with another association (see 6.2.2.2).

6.1.3 Bound data

The operations of an atomic action involve specific CCR service-user data as determined by the requirements of the referencing specification. For this Service Specification, such data under the control of an atomic action are called bound data.

Modifications made by the operations of the atomic action change the bound data from an initial state to a final state. The modifications are indivisible and either all are applied (placing the bound data in the final state) or none are applied (placing the bound data in the initial state).

During an atomic action, an intermediate state of the bound data is invisible outside of the atomic action. Any modifications are isolated from concurrent operations that take place outside of the atomic action.

6.1.4 Atomic action data

For this Service Specification, the term atomic action data refers to state and control information about an atomic action and its branches. Atomic action data needed for recovery is required to persist if an application or communication failure occurs.

6.1.5 Operation of an atomic action

The overall goal of an atomic action is to exchange application semantics to coordinate the setting of the final state of all bound data. To achieve this, CCR supports a two-phase commitment mechanism. During phase I ready signals are collected. During phase II commitment is ordered and confirmed.

Within the atomic action, each CCR service-user may send a ready signal on exactly one branch or it may send no ready signal. The atomic action may therefore be represented as a hierarchical tree ordered on the basis of the opposite of the directions in which the ready signals are sent – this is the atomic action commit-tree. (Called the commit-tree because commit-orders will be sent, if they are sent, in the opposite direction to the ready signals, which will be down the arcs of the commit-tree).

6.1.6 Commit-tree

The order of an atomic action graph can be based on the sending of ready signals. For this Recommendation | International Standard, this hierarchical tree is called the **atomic action commit-tree**. In this structure, the following roles are defined:

a) **commit-superior**: On a given branch, the CCR service-user which receives the ready signal.

b) **commit-subordinate**: On a given branch, the CCR service-user which sends a ready signal.

c) **commit coordinator**: The CCR service-user which is the commit-superior of all of its neighbours.

d) **commit-decider**: A commit coordinator that orders commitment (only a commit coordinator can order commitment).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figures 3 to 5 show three examples of different commit-trees for the same atomic action begin-tree shown in figure 2. The arrows indicate the directions the ready signals were sent:

– The commit-tree in Figure 3 has the same ordering as the begin-tree. The atomic action initiator (CCR service-user A) is the commit coordinator. This occurred because all ready signals were from branch-responder to branch-initiator.

– Figure 4 has a single commit coordinator (CCR service-user E) that is different from the atomic action initiator (CCR service-user A).

– Figure 5 has two commit coordinator (CCR service-users D and F). This occurred because of a collision of ready signals between them.

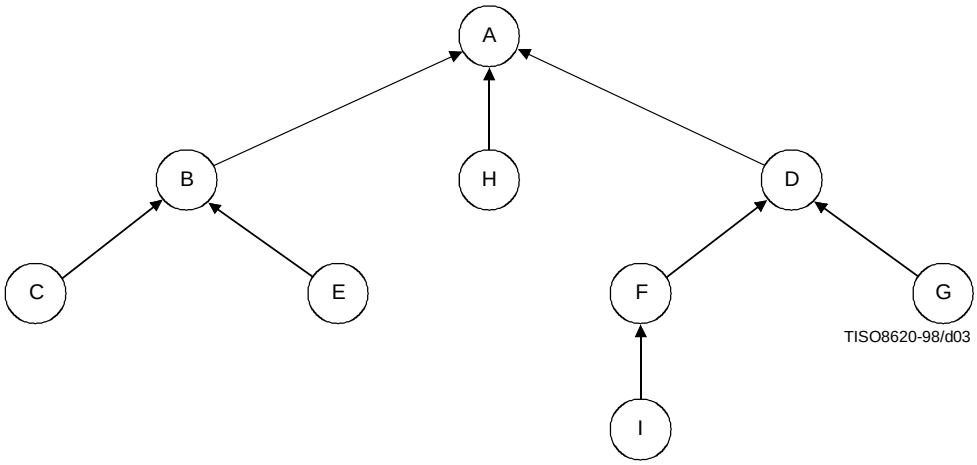

Figure 3 – Commit-tree the same as the begin-tree

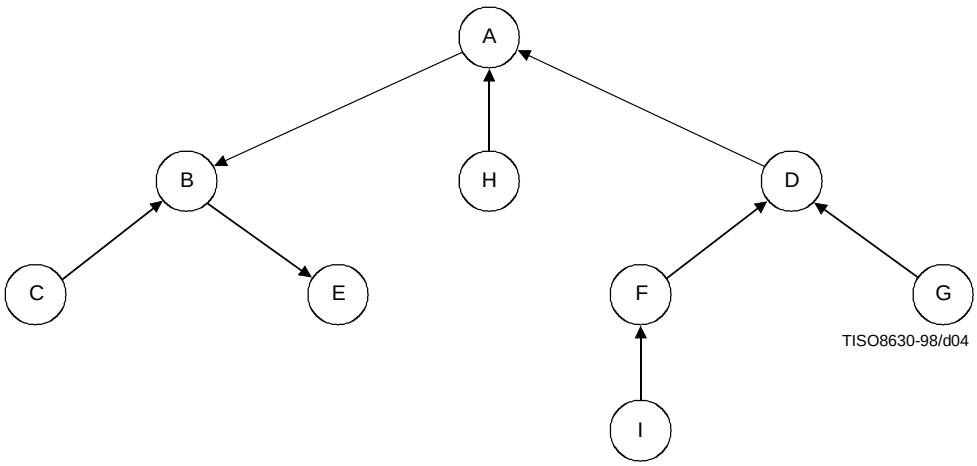

Figure 4 – Single commit coordinator different from the atomic action initiator

--`,,```,,,,````-`-,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

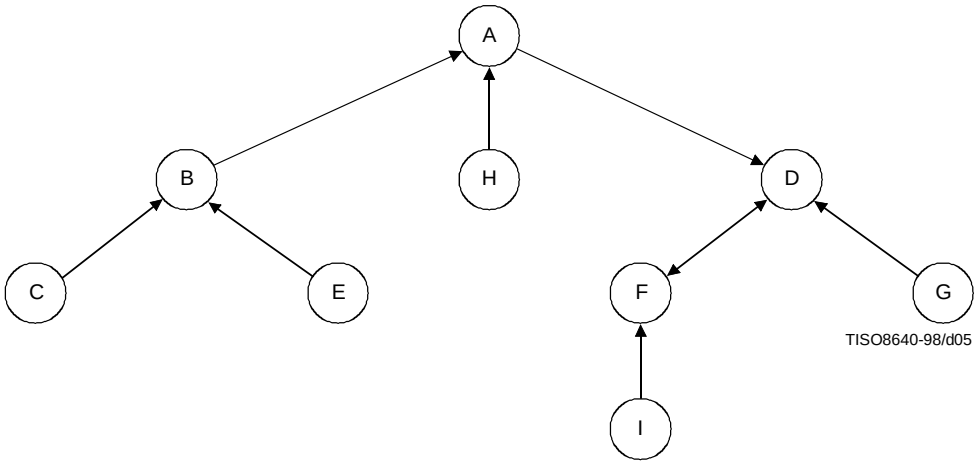

Figure 5 – Two commit coordinators following collision of ready signals

A commit-tree may have one or two commit coordinators.

When there is only one commit coordinator, it can decide to commit the atomic action, thus becoming commit-decider, and will then send commit-orders to all its commit-subordinates.

A commit-tree has two commit-coordinators if, and only if, a ready signal collision occurs between them. In such a case, both CCR service-users are treated as commit-superior and both are treated as commit-subordinate as far as actions on their common branch are concerned.

Unless the collision is the result of incorrect application behaviour, the atomic action should proceed to commitment. This is because both commit coordinators have signalled readiness and they cannot roll back their common branch without contradicting their previous ready signal.

Following a ready signal collision, in the normal case, at least one commit coordinator becomes commit-decider and orders its commit-subordinates to commit. The referencing specification may require one commit coordinator to wait, and not send an order to its other subordinates until it receives the commit order from the other – in this case, the referencing specification will define what polarity of the branch between them is to be used to determine who waits and who sends commit-order immediately (i.e. which side becomes commit-decider). This polarity may be static (e.g. which side established the association or which side initiated a dialogue), or dynamic (e.g. which side currently possess the sync-minor token).

Alternatively, the referencing specification may determine that both commit coordinators immediately order their own commit-subordinates to commit, and thus there are two commit-deciders (which come to the same decision). The commit-deciders can send commit-orders to each other, as far as the semantics of this Recommendation | International Standard are concerned. However, the mapping of CCR to supporting services may make it impossible for both commit-deciders to send the commit order to each other – in this case, the referencing specification will determine which one will send the order. (The reference mapping for CCR, specified in ITU-T Rec. X.852 | ISO/IEC 9805-1, makes it impossible for the both to send the commit-order on the original association, although it is possible if the association has failed and the commit-order is being sent via the recovery mechanism.)

As is described later, before a commit coordinator can become a commit-decider, and initiate ordering commitment, it is required to write atomic action data indicating a commit order. A CCR service-user that observes a ready signal collision will already have atomic action data indicating the (outbound) ready signal. It must replace that atomic action data with atomic action data that indicates a commit-order. This conversion of the atomic action data must be performed such that following any failure, the atomic action data are complete and consistent one way or the other.

Depending on details of the implementation, it is possible that this seamless conversion of the atomic action data cannot be guaranteed. This limitation could apply in all cases (e.g. if writing atomic action data damages any existing atomic action data before the new data is completely written), or could only apply occasionally (e.g. under conditions of heavy load, any writing may be impossible). Conceivably, both CCR service-users could be in this position and the branch

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

would get stuck. To avoid this, it is necessary to allow a CCR service-user that is unable to change the atomic action data to initiate a rollback. However, to ensure preservation of the ACID properties, this requires that the other CCR service-user does not itself initiate commit-ordering. Accordingly, at association establishment, the CCR service-users can indicate whether, in the event of a ready signal collision, they will follow the strict logic of the ready signals or whether they are permitted to initiate a rollback.

The referencing specification will provide rules that avoid one side initiating rollback while the other side orders commitment. This can be combined with referencing specification provisions that ensure that only one of the two commit coordinators attempts to order commitment – if only one of the commit coordinators is permitted to order commitment, there is no risk to the ACID properties if it initiates rollback when it is unable convert the atomic action data.

NOTE – When OSI Distributed Transaction Processing, (see ITU-T Rec. X.862 | ISO/IEC 10026-3), is the referencing specification, the position of the synchronize minor token is used to determine which commit coordinator becomes commit decider. The other one waits for the commit (or rollback) order.

6.1.7 Two-phase commitment

CCR supports a two-phase commitment mechanism. During phase I ready signals are collected. This Recommendation | International Standard does not specify when phase I starts.

A CCR service-user can send a ready signal to one of its neighbours when it has received ready signals from all its other neighbours, has completed all operations and is capable of placing its bound data in either the initial or final state. The neighbours from whom ready signals were received are the commit-subordinates of the CCR service-user; the neighbour to whom the ready signal is sent is the commit-superior of the CCR service-user.

A CCR service-user, leaves phase I and enters the doubt period when, and if, it decides to send a ready signal. It leaves the doubt period and enters phase II when it receives the order to commit from its commit-superior. If it has commit-subordinates, then orders them to commit. Finally, it leaves phase II when it sends commitment confirmation to its commit-superior.

A CCR service-user that receives ready signals from all of its neighbours (a commit-coordinator), has also completed all operations and is capable of placing its bound data in the final state can decide to commit the atomic action. Such a CCR service-user leaves phase I (doubt period if a ready signal collision occurs) and enters phase II when it decides to commit the atomic action – it is then the commit-decider. The commit-decider then orders its commit-subordinates to commit. The commit-decider leaves phase II after receiving commitment confirmation from all its commit-subordinates to which it has ordered commitment.

NOTE – The term "master" is no longer used in this Recommendation | International Standard to avoid confusion with the term "commit-master", used in ITU-T Rec. X.860 | ISO/IEC 10026-1.

6.1.8 Commitment procedure

Commitment is the procedure whereby the CCR service-users participating in an atomic action, release their bound data in the final state.

Commitment only occurs after all participating CCR service-users (other than the commit coordinator, if there is only one) have sent ready signals. The commit-decider initiates commitment. When the commit-decider decides to commit, it enters phase II. As each CCR service-user commits, it releases its bound data in the final state and orders all of its commit-subordinates to commit.

6.1.9 Rollback procedure

Rollback is the procedure used to force the completion of some or all the branches of an atomic action. The procedure results in the release of related bound data in the initial state. Rollback may apply to an entire atomic action. It may also apply to a sub-tree of the atomic action graph.

A CCR service-user, may initiate rollback prior to (and instead of) sending a ready signal, if it is a commit coordinator, prior to and instead of ordering commitment.

For rollback, a CCR service-user releases its bound data in the initial state. It forces the completion of the branches to its commit-subordinates by propagating the rollback on them. If it initiated rollback, it forces the completion of the branch to its commit-superior.

Prior to sending a ready signal (i.e. before entering the doubt phase), a CCR service-user may order any of its subordinates to roll back even if it does not roll back or release its own bound data. The branches with such subordinates are completed. The CCR service-user remains in the atomic action.

After sending a ready signal, a CCR service-user that has not taken a heuristic decision only rolls back if it receives an order to roll back from its commit-superior (see 6.3).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.1.10 No-change completion

No-change completion is a procedure by which a CCR service-user which has not modified any bound data during the atomic action indicates that it has completed processing and, as far as it is concerned, the atomic action can commit. It can be used in two ways – as a **one-phase commit** order and as a **read-only** signal.

One-phase commitment is ordered by a CCR service user that wishes the atomic action to proceed to commitment, if possible, but has made no changes to local bound data and which does not require the result of the atomic action to be reliably communicated to it via the CCR procedures. If there is no failure, the result will be communicated.

When used as a one-phase commit order, the C-NOCHANGE service is confirmed, with the confirmation delayed until the result of the atomic action is known, or it is determined that the result will not be known. The branch continues to exist until the confirmation is sent/issued or the association fails.

NOTE – Although one-phase commitment requires that the CCR service user has made no changes to bound data, this refers only to data that is bound data according to the definition in 3.6.10. There may be other data that is accessed and manipulated by the CCR service-user as a result of the atomic action, but which is not bound by the rules of CCR. Maintaining the ACID properties for such other data, if this is required, is outside the scope of CCR.

Read-only is used by a CCR service-user that has not modified any bound data during the atomic action and has no requirement to be informed of the result, or even to know when the distributed atomic action completes. The read-only signal removes the CCR service-user from the atomic action.

When used as a read-only signal, the C-NOCHANGE service is optionally confirmed, but the confirmation is just an acknowledgement that the signal has been received and that the branch is known to be completed. If there is no confirmation, the next use of the association (for CCR) shall be by the acceptor of the C-NOCHANGE indication – the CCR service-user that issued the C-NOCHANGE request cannot initiate a new branch.

A CCR service-user can only use the no-change completion procedure if it has only one neighbour (to whom it issues the C-NOCHANGE request), other than any neighbours that have sent C-NOCHANGE indications to it.

A CCR service-user may use the no-change completion procedure any time prior to signalling readiness, ordering commitment or ordering rollback. A CCR service-user that uses the no-change completion procedure may not subsequently initiate any new branches of the atomic action.

There is a possible risk to the ACID properties of the bound data if the no-change completion procedure is used by a CCR service-user that has accessed but not changed bound data (i.e. read-only access). This bound data could be subsequently changed by some other activity and then accessed by another CCR service-user within the original atomic action. Consequently, care should be exercised when using the no-change completion procedure. It should only be used if serializability can be guaranteed.

6.1.11 Concurrency control

Concurrency control is a real open system mechanism. It coordinates modifications to bound data used by concurrent atomic actions. A concurrency control mechanism guarantees the atomic action isolation property.

NOTE – A concurrency control mechanism ensures that at least one serial sequence of a given set of atomic actions exists that produces the same result to the common bound data as the concurrent (parallel) operation of the same atomic actions on the same bound data. That is, the concurrent execution of atomic actions is serializable.

CCR requires concurrency control for the control of atomic actions. However, the facility to accomplish concurrency is outside the scope of this Recommendation | International Standard.

6.2 CCR facilities

CCR facilities support the beginning and completion of a single branch. The overall goal of a branch is to exchange application semantics to cause the modification of bound data in a coordinated manner.

6.2.1 Operation of a branch

The operation of a branch is divided into two parts:

a) creation of the branch and the exchange of application semantics between the two CCR service-users to produce the final state of the bound data; and

b) commitment whereby the final state of the bound data is made permanent (i.e. committed) or rollback whereby the bound data are restored to the initial state.

At any time before starting the commitment procedure, either CCR service-user may roll back the branch.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A branch can be **interrupted** by an application or communication failure. A CCR service-user with **recovery responsibility** attempts to recover an interrupted branch using another association. A CCR service-user acquires recovery responsibility for a branch before it uses specific CCR services (see 6.2.2.2). Both CCR service-users may have recovery responsibility for the branch.

This Recommendation | International Standard defines CCR services for creating and controlling an individual branch. It also defines rules that govern the exchange of application semantics on a branch.

NOTE – The exchange of application semantics within the framework of a branch is defined by the referencing specification.

6.2.2 Recovery

CCR addresses failure and subsequent recovery at the branch level.

6.2.2.1 Failure

AEIs involved in an atomic action can fail at any time. However, CCR functionality and applicability rely upon the preservation of the bound data and atomic action data over such failures. The loss of such data causes a breakdown of the CCR functionality and applicability and the atomic action properties are no longer guaranteed.

Following an application or communication failure, recovery on another association may be needed. This is done to preserve the atomic action properties and to place the bound data into a consistent state. In particular, the CCR service-user may invoke CCR recovery facilities on another association to recover CCR semantic exchanges that may have been lost.

The CCR service-user accesses atomic action data when it invokes the CCR recovery facilities. Atomic action data and the CCR recovery facilities enable the CCR service-user to complete the branch.

NOTE – Following an application failure, local recovery mechanisms may be needed to restore the CCR service-user. These mechanisms may be used at a later time and may involve human intervention.

6.2.2.2 Recovery mechanism

A recovery mechanism determines when the CCR service-users of a branch acquire recovery responsibility for the branch. If a failure occurs, a CCR service-user with recovery responsibility attempts the recovery of that branch.

CCR employs the presumed rollback (sometimes called "presumed abort") recovery mechanism. For this mechanism, the commit-subordinate acquires recovery responsibility when it decides to send a ready signal. The commit-superior acquires recovery responsibility when it decides to order commitment. Both keep recovery responsibility until the completion of the branch.

NOTE – For the commit-decider, the presumed rollback recovery mechanism does not require the recording of atomic action data until it decides to commit the atomic action. For a CCR service-user that is not the commit-decider, the recording of atomic action data does not occur until it decides to send a ready signal. This reduces the overhead of recording atomic action data at the beginning of the branch.

The CCR recovery mechanism for an individual branch makes three basic requirements of the CCR service-user:

a) the maintenance of atomic action data;

b) the ability to set the initial or final state of bound data; and

c) the initiation of recovery when it has recovery responsibility.

The CCR service-user uses atomic action data to determine if it has recovery responsibility.

Before commitment, a CCR service-user does not have recovery responsibility. If an application or communication failure occurs, the CCR service-user shall be capable of restoring its bound data to the initial state.

During the doubt period, a CCR service-user has recovery responsibility. If an application or communication failure occurs, the CCR service-user shall be capable of placing its bound data in either the initial or final state.

After an application failure, local recovery mechanisms re-establish the operation the CCR service-user. The CCR service-user then attempts to use a new association to recover any branch for which it has recovery responsibility.

After a communication failure, the CCR service-user attempts to use another association to recover the branch if it has recovery responsibility.

Recovery responsibility is determined by the atomic action data.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.3 Heuristic decisions

This Recommendation | International Standard does not explicitly provide capabilities to communicate heuristic decisions, nor the means to reduce the impact of such decisions. This discussion is included because a referencing specification may define conditions concerning heuristic decisions that affect the use of CCR services.

6.3.1 Rationale for heuristic decisions

After a CCR service-user sends a ready signal, the CCR service-user is in the doubt period. It keeps the capability to commit or to roll back until ordered to do so by the commit-superior. In practice, this may not be acceptable. A prolonged failure may occur or an exceptionally long delay may take place before the decision to commit or roll back is communicated to it.

In such circumstances, a CCR service-user may decide to take a heuristic decision. It puts some or all of its bound data into the initial state, the final state or some intermediate state. It does this while still in the doubt period.

For a heuristic decision, the CCR service-user considers the trade-off between:

a) keeping the capability to commit or to roll back (e.g. keeping locks on valuable data); and

b) taking a heuristic decision that possibly violates the atomic action properties and then coping with the effects of this violation.

6.3.2 Taking a heuristic decision

Any CCR service-user that has sent a ready signal may take a heuristic decision. This includes a CCR service-user involved in a branch interrupted by an application or communication failure. A CCR service-user may take more than one heuristic decision for a given atomic action.

A referencing specification may specify constraints on the taking of heuristic decisions. This includes not allowing heuristic decisions.

A heuristic decision of a CCR service-user that is different from that taken by the commit coordinator results in a mixed situation.

Mixed situations are resolved by compensating actions. These compensating actions are application-specific as well as situation-specific. Compensating actions are outside the scope of this Service Specification.

6.3.3 Detection of heuristic mixed situation

The use of CCR services guarantees that any CCR service-user that took a heuristic decision eventually detects whether its decision was in line with the decision of the commit coordinator or if a mixed situation has occurred.

6.3.4 Reporting of heuristic mixed situation

When a mixed situation is detected, the referencing specification is responsible for reporting to an entity capable of resolving the mixed situation.

A referencing specification may use the User Data parameter of some CCR service primitives to communicate the existence of a heuristic decision or a heuristic mixed situation. Such communication may not be reliable.

7 Service definition

This clause defines each CCR service. Clause 8 describes the allowed sequences of CCR service primitives used on one branch of an atomic action. Annex A specifies CCR service-user rules that a referencing specification shall incorporate.

Table 1 lists the CCR services, the type of service (confirmed, optionally confirmed or non-confirmed), and the requestor of the service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1 – CCR services

Service	Type	Requestor
C-INITIALIZE	Confirmed	Association-initiator
C-BEGIN	Optionally confirmed	Either CCR service-user
C-PREPARE	Non-confirmed	Branch-initiator
C-READY	Non-confirmed	Either CCR service-user
C-COMMIT	Confirmed	Either CCR service-user
C-ROLLBACK	Confirmed	Either CCR service-user
C-NOCHANGE	Optionally confirmed	Either CCR service-user
C-CANCEL	Non-confirmed	Either CCR service-user
C-RECOVER	Confirmed, or Optionally confirmed	Either CCR service-user
C-P-ERROR	Provider initiated	

7.1 C-INITIALIZE service

7.1.1 Purpose and use

7.1.1.1 C-INITIALIZE service is a confirmed service. It is used to specify requirements and determine capabilities of the CCR services to be used on the association that is being established.

7.1.1.2 The service is used during association establishment by the A-ASSOCIATE requestor (i.e. association-initiator). The C-INITIALIZE acceptor is the A-ASSOCIATE acceptor (i.e. the association-responder).

7.1.1.3 If the C-INITIALIZE service is not used during association establishment, only the static commitment functional unit is available on the association.

7.1.2 C-INITIALIZE parameters

Table 2 lists the C-INITIALIZE service parameters. Each parameter is discussed below.

Table 2 — C-INITIALIZE parameters

Parameter name	Req	Ind	Rsp	Cnf
CCR Requirements	M	C	M	C(=)
Version		M		M(=)
User Data	U	C(=)	U	C(=)

7.1.2.1 CCR Requirements

7.1.2.1.1 The requestor uses this parameter to indicate the functional units requested for the CCR-service to be used on the association. In supporting this negotiation mechanism, the CCR service-provider modifies the parameter specified in the request primitive before its delivery in the indication primitive. It does this by any removing functional units from the requested set that it does not support.

7.1.2.1.2 The acceptor uses this parameter to indicate which of the requested functional units it selects. The acceptor does not select a functional unit for the response primitive that was not on the indication primitive.

7.1.2.1.3 The value of the parameter on the response primitive is delivered unchanged on the confirm primitive.

7.1.2.1.4 This parameter assumes one or more of the following values:

- static-commitment;
- dynamic-commitment;
- no-change completion;
- cancel.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.1.2.1.5 At least one of "static-commitment" and "dynamic-commitment" shall be specified on the request primitive. If both are present on the indication primitive, the acceptor shall include only one on the response primitive.

7.1.2.1.6 The remaining values may be specified in any combination.

7.1.2.2 Version

This parameter indicates the version of the CCR protocol used on the association.

7.1.2.3 Ready-collision-reservation

This boolean parameter is absent unless the "dynamic-commitment" functional unit is selected.

It is set "true" on the request and indication if the requestor is to be permitted to initiate rollback after a ready signal collision. It is set "false" if the requestor will not initiate rollback after a ready signal collision.

It is set "true" on the confirm and response if the responder is to be permitted to initiate rollback after a ready signal collision. It is set "false" if the responder will not initiate rollback after a ready signal collision.

7.1.2.4 User Data

This parameter can carry an unlimited amount of information. Its use is determined by the referencing specification. It can contain one or more presentation data values from presentation contexts in the defined context set when the C-INITIALIZE request primitive is issued.

7.2 C-BEGIN service

7.2.1 Purpose and use

7.2.1.1 A CCR service-user, called the branch-initiator, uses the C-BEGIN request primitive to request the beginning of a branch with another CCR service-user, called the branch-responder. The C-BEGIN service is used on an established association. The branch-responder is included in the same atomic action as the branch-initiator.

7.2.1.2 The branch-responder may optionally use the C-BEGIN response primitive before sending the first application semantics for this new branch. In this case, application semantics sent before the C-BEGIN response primitive are not part of this new branch.

7.2.1.3 If the reference mapping is being used, the use of the C-BEGIN service has the effect of establishing a minor synchronization point on the underlying session-connection that supports the branch. The branch-initiator shall own the synchronize-minor token.

7.2.1.4 The association used by the requestor shall not be in use for the recovery of an interrupted branch (see 7.9). The C-BEGIN service shall not be invoked on an association that is being used for another "active" branch.

7.2.1.5 The C-BEGIN service may be jointly issued with the C-COMMIT services (see 7.5).

7.2.2 C-BEGIN parameters

Table 3 lists the C-BEGIN service parameters. Each parameter is discussed below.

Table 3 – C-BEGIN parameters

Parameter name	Req	Ind	Rsp	Cnf
Atomic Action Identifier – Owner's Name	M	M(=)		
Atomic Action Identifier – Suffix	M	M(=)		
Branch Identifier – Initiator's Name	M	M(=)		
Branch Identifier – Suffix	M	M(=)		
User Data	U	C(=)	U	C(=)

7.2.2.1 Atomic action identifier

7.2.2.1.1 The Atomic Action Identifier unambiguously identifies the atomic action to which this branch belongs. Its value is assigned by the atomic action owner. This value is subsequently used by the branch-initiator of each branch.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.2.2.1.2 The Atomic Action Identifier consists of the Owner's Name parameter together with the Suffix parameter.

7.2.2.1.3 The value of the Owner's Name parameter has the form of an AE title. This value unambiguously identifies the number-space within which the Suffix is unique.

NOTE – The Owner's Name parameter can be the atomic action owner's AE title, or it may be a value passed to the atomic action owner by some means.

7.2.2.1.4 The atomic action owner assigns the value of the Suffix parameter so the value unambiguously identifies the atomic action among all those with the same atomic action owner's name.

NOTE 1 – If the atomic action initiator is not the atomic action owner, the Atomic Action Identifier is communicated to the atomic action initiator by a mechanism defined by the referencing specification.

NOTE 2 – The atomic owner's name is used only to ensure the combination of Owner's Name and Suffix are globally unambiguous. The atomic owner need not be a participant in the atomic action, or even be capable of supporting CCR.

7.2.2.2 Branch identifier

7.2.2.2.1 The Branch Identifier unambiguously identifies a branch of an atomic action within the scope of the value of the Atomic Action Identifier. It consists of the Initiator's Name parameter together with the Suffix parameter.

7.2.2.2.2 The value of the Initiator's Name is the branch-initiator's AE title. This value unambiguously identifies the initiator of the branch.

NOTE – The A-ASSOCIATE service of ACSE provides a facility to exchange AE title values (see ITU-T Rec. X.217 | ISO/IEC 8649).

7.2.2.2.3 This Service Specification requires the use of either the Calling AE Title or Responding AE Title parameters of the A-ASSOCIATE service to identify the branch-initiator.

7.2.2.2.4 The branch-initiator assigns the value of the Suffix parameter so the value unambiguously identifies this branch among all those branches of this atomic action with the same branch-initiator's name.

7.2.2.3 User Data

This parameter may carry an unlimited amount of information as determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context set when the C-BEGIN request primitive is issued.

NOTE – The referencing specification determines the use of this parameter. For example, it can indicate the minimum requirements for commitment, the preferred character sets for diagnostics, or additional information about the nature of this branch.

7.3 C-PREPARE service

7.3.1 Purpose and use

7.3.1.1 C-PREPARE is a non-confirmed service. If the Static Commitment functional unit was selected for the supporting association, the requestor shall be the branch-initiator. If the Dynamic Commitment functional unit was selected, a CCR service-user may optionally invoke C-PREPARE if it has not received a ready signal from the neighbour. By issuing the request primitive, the requestor tells its neighbour the requestor will not send the neighbour any further application semantics that change the bound data of this atomic action. The requestor also indicates that the neighbour should complete processing for the branch and send a ready signal.

7.3.1.2 The C-PREPARE service is not needed when the application semantic exchange of a branch provides an equivalent prepare request.

7.3.2 C-PREPARE parameter

Table 4 lists the C-PREPARE service parameter.

Table 4 – C-PREPARE parameter

Parameter name	Req	Ind
User Data	U	C(=)

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.2.1 User Data

The requestor may use this parameter to carry an unlimited amount of information determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context.

NOTE – The referencing specification determines the use of this parameter.

7.4 C-READY service

7.4.1 Purpose and use

7.4.1.1 C-READY is a non-confirmed service that a CCR service-user invokes to send a ready signal. If the Static Commitment functional unit was selected for the supporting association, the requestor must be the branch-responder. The requestor issues the request primitive only if it has ensured that bound data can be released in either the initial or final state.

7.4.1.2 The requestor has recovery responsibility for this branch. That is, it shall attempt recovery after an application or communication failure (see 7.9).

7.4.1.3 The requestor shall not send the neighbour any further application semantics that change the bound data of this atomic action.

7.4.2 C-READY parameter

Table 5 lists the C-READY service parameter.

Table 5 – C-READY parameter

Parameter name	Req	Ind
User Data	U	C(=)

7.4.2.1 User Data

The requestor may use this parameter to carry an unlimited amount of information as determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context set when the C-READY request primitive is issued.

NOTE – The referencing specification determines the use of this parameter. For example, it can carry warnings of requested action variations.

7.5 C-COMMIT service

7.5.1 Purpose and use

7.5.1.1 C-COMMIT is a confirmed service that a commit-superior invokes to order commitment.

NOTE – If the Static Commitment functional unit was selected for the supporting association, the requestor is the branch-initiator.

7.5.1.2 The commit-superior issues the request primitive only if the following is true:

a) the commit-subordinate has sent a ready signal (see 7.4); and

b) the commit-superior has its bound data in the final state.

The commit-superior has recovery responsibility for this branch. That is, it shall attempt recovery after an application or communication failure (see 7.9).

7.5.1.3 The commit-subordinate issues the response primitive to complete the branch. Before issuing the primitive, the commit-subordinate shall release all of its resources (e.g. bound data in the final state). The commit-subordinate no longer has recovery responsibility for this branch. That is, it shall not attempt recovery after an application or communication failure.

7.5.1.4 For the commit-superior, the branch is completed when it receives the confirm primitive. It no longer has recovery responsibility for this branch. That is, it shall not attempt recovery after an application or communication failure.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.5.1.5 If the reference mapping is being used, then the use of C-COMMIT service has the effect of establishing a minor synchronization point on the underlying session-connection that supports the branch. The commit-superior shall own the synchronize-minor token.

7.5.1.6 The commit-superior may jointly issue a C-BEGIN request primitive with the C-COMMIT request primitive.

7.5.2 C-COMMIT parameter

Table 6 lists the C-COMMIT service parameter.

Table 6 – C-COMMIT parameter

Parameter name	Req	Ind	Rsp	Cnf
User Data	U	C(=)	U	C(=)

7.5.2.1 User Data

This parameter may carry an unlimited amount of information determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context set when the C-COMMIT request primitive is issued.

NOTE – The referencing specification determines the use of this parameter.

7.6 C-ROLLBACK service

7.6.1 Purpose and use

7.6.1.1 C-ROLLBACK is a confirmed service that either CCR service-user may invoke to force completion of the branch. This service may cause the loss of application semantics in transit on the underlying session-connection that supports the branch.

7.6.1.2 The use of this service depends on the CCR services previously used on this branch.

a) If a CCR service-user has neither signalled readiness nor ordered commitment on this branch, it may invoke C-ROLLBACK at any time. This forces the completion of the branch. If the requestor or the acceptor is also the branch-responder, it performs local rollback procedures for the branch.

b) If a CCR service-user is the commit-superior, it may invoke C-ROLLBACK any time before ordering commitment. This forces completion of the branch and orders the commit-subordinate to roll back.

c) If a CCR service-user has both sent and received a ready signal, it may invoke C-ROLLBACK, unless it sets the Ready-collision-reservation parameter "false" on the C-INITIALIZE request or response sent on that association. Invoking C-ROLLBACK forces completion of the branch and orders the peer CCR service-user to roll back.

7.6.1.3 A service collision of two C-ROLLBACK request primitives can occur. For this reason, the delivery of the indication primitive to the CCR service-user that was the association-initiator is not guaranteed. However, both CCR service-users are aware that the branch has been rolled back.

7.6.1.4 If the reference mapping is being used, the use of the C-ROLLBACK service involves resynchronization of the underlying session-connection that supports the branch.

7.6.2 C-ROLLBACK parameter

Table 7 lists the C-ROLLBACK parameter.

Table 7 – C-ROLLBACK parameter

Parameter name	Req	Ind	Rsp	Cnf
User Data	U	C(=)	U	C(=)

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.6.2.1 User Data

This parameter may carry an unlimited amount of information as determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context set when the C-ROLLBACK request primitive is issued.

NOTE – The delivery of the User Data parameter is not guaranteed because of the chance of collision.

7.7 C-NOCHANGE service

7.7.1 Purpose and use

7.7.1.1 C-NOCHANGE is an optionally confirmed service used by a CCR service-user to inform its neighbour that it has completed its processing and that it has not modified any bound data during the atomic action. The service is optionally confirmed. The requestor may ask to be informed on the confirm of the eventual result (commit or rollback) of the atomic action, or may ask just to be informed that the C-NOCHANGE indication has been received or may ask that the service be not confirmed at all.

7.7.1.2 To issue the request primitive, the requestor:

a) shall not have signalled readiness, ordered commitment or ordered rollback;

b) shall have made no changes to bound data;

c) shall have no neighbours, other than the one to whom the C-NOCHANGE request is issued, from whom a C-NOCHANGE indication has been received.

NOTE – The requirement to have made no changes to bound data does not preclude the use of C-NOCHANGE by a CCR service-user that has made changes to other data that will not be made subject to CCR's restrictions on the manipulation of bound data. By definition, such data are not bound data within the sense of this Recommendation | International Standard.

7.7.1.3 The C-NOCHANGE service should only be used if serializability between atomic actions can be guaranteed. A problem may occur if a CCR service-user accesses data in a read-only manner and uses C-NOCHANGE, but subsequently the (read-only) bound data is changed by another activity and then accessed by another CCR service-user within the original atomic action.

7.7.2 C-NOCHANGE parameters

Table 8 lists the C-NOCHANGE service parameters.

Table 8 – C-NOCHANGE parameters

Parameter name	Req	Ind	Rsp	Cnf
Confirmation	M	M(=)		
Outcome	U		U	C(=)
User Data	U	C(=)	U	C(=)

7.7.2.1 Confirmation

The requestor uses this parameter to indicate whether a confirmation is required to report the atomic action outcome. It takes one of the following symbolic values:

– result-requested; or

– not-required.

If the value on the indication is "with-result", the acceptor shall not issue the response until the result of the atomic action is known to the acceptor, or it has been determined that the result will not be known in a timely manner.

If the value on the indication is "not-required", the acceptor is not required to issue a response, although it is permitted to.

NOTE – The referencing specification may determine the circumstances in which an acceptor replies "not-determined" to a "with-result" indication or when a response is issued following a "not-required" indication.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.7.2.2 Outcome

The acceptor uses this parameter to express the result of the atomic action. It takes one of the following symbolic values:

- committed; or
- rolled-back; or
- no-change; or
- not-determined.

The value "not-determined" shall be used when the result of the atomic action is not known to the acceptor when the response is issued.

7.7.2.3 User Data

This parameter can carry an unlimited amount of information. Its use is determined by the referencing specification. It can contain one or more presentation data values from presentation contexts in the defined context set when the C-NOCHANGE request primitive is issued.

7.8 C-CANCEL service

7.8.1 Purpose and use

7.8.1.1 C-CANCEL is a non-confirmed service that a requestor invokes to inform its peer that the branch will be rolled back. This allows the peer to take any "ending" action prior to the actual roll back of the branch. After sending a C-CANCEL request primitive or receiving a C-CANCEL indication primitive, only a C-ROLLBACK request or indication shall be issued or received.

7.8.2 C-CANCEL parameter

Table 9 lists the C-CANCEL service parameter.

Table 9 – C-CANCEL parameter

Parameter name	Req	Ind
User Data	U	C(=)

7.8.2.1 User Data

The requestor may use this parameter to carry an unlimited amount of information determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context set when the C-CANCEL request primitive is issued.

NOTE — The referencing specification determines the use of this parameter.

7.9 C-RECOVER service

7.9.1 Purpose and use

7.9.1.1 Either CCR service-user that has recovery responsibility may invoke the C-RECOVER service after a failure. It is either a confirmed or optionally confirmed service depending on the requestor.

7.9.1.2 When a commit-superior is the requestor, C-RECOVER is a confirmed service.

7.9.1.3 When a commit-subordinate is the requestor, C-RECOVER is an optionally confirmed service. The acceptor (i.e. the commit-superior) may reply with a C-RECOVER response primitive thus continuing this service procedure. Alternatively, the commit-superior may reply with a C-RECOVER request primitive *for the same atomic action branch* thus ending this C-RECOVER service procedure and starting a second service procedure. The commit-superior is the requestor for the second C-RECOVER service procedure.

7.9.1.4 The association used by the requestor shall not be in use for another recovery or for another branch. However, this requirement is relaxed if the requestor is the superior and it issues a C-RECOVER request primitive as the reply to a C-RECOVER indication primitive.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.9.1.5 If the reference mapping is in use, the requestor shall own the synchronize-minor token for the underlying session-connection that supports the branch. However, this requirement is also relaxed if the requestor is the commit-superior and it issues a C-RECOVER request primitive as the reply to a C-RECOVER indication.

7.9.2 C-RECOVER parameters

Table 10 lists the C-RECOVER service parameters. Each parameter is discussed below.

Table 10 – C-RECOVER parameter

Parameter name	Req	Ind	Rsp	Cnf
Recovery State	M	M(=)	M	M(=)
Atomic Action Identifier	M	M(=)	M	M(=)
Branch Identifier	M	M(=)	M	M(=)
Reversed Branch	C	C(=)	C	C(=)
User Data	U	C(=)	U	C(=)

7.9.2.1 Recovery state

7.9.2.1.1 The requestor uses this parameter to express its perception of the state of the identified branch. The acceptor uses the parameter to express its reply. Table 11 lists the values for this parameter when the commit-subordinate or the commit-superior is the requestor.

Table 11 – Recover State parameter values

Requestor: acceptor	Req value	Rsp value
commit-subordinate: commit-superior	ready	unknown; or retry-later
commit-superior: commit-subordinate	commit	done; or retry-later

7.9.2.1.2 The values of this parameter have the following meanings:

a) "ready" is used by the commit-subordinate to inform the commit-superior that it had previously sent a ready signal. The following is true about the commit-subordinate:

   1) Its bound data can be released in either the initial or final state.

   2) It has recovery responsibility for this branch. That is, it shall attempt recovery after an application or communication failure (see 7.9).

   3) It shall not send any further application semantics to the commit-superior that change the bound data of this atomic action.

b) "commit" is used by the commit-superior to inform the commit-subordinate that it had previously ordered commitment. The following is true about the commit-superior:

   1) Its bound data is in the final state.

   2) It has recovery responsibility for this branch. That is, it shall attempt recovery after an application or communication failure.

c) "unknown" is used by the commit-superior to indicate that it has no atomic action data for this branch.

d) "retry-later" is used by the acceptor to indicate that it cannot proceed with recovery now. In this situation, the requestor reissues the C-RECOVER request primitive at a later time.

   NOTE – A common use of retry-later is when a commit-subordinate receives a C-RECOVER indication primitive and it cannot establish an association with one or more of its commit-subordinates.

e) "done" indicates that the commit-subordinate has completed commitment.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.9.2.2 Atomic Action Identifier

This parameter identifies the atomic action whose branch is being recovered. It has the same form and value as that on the corresponding C-BEGIN service (see 7.2.2.1).

7.9.2.3 Branch Identifier

This parameter identifies the branch being recovered. It has the same form and value as that of the corresponding C-BEGIN service (see 7.2.2.2).

7.9.2.4 Reversed Branch

This boolean parameter identifies whether the initiator of the branch is the commit-subordinate. It shall be set "true" if the branch-initiator is the commit-subordinate. It shall be set "false" if the branch-initiator is not the commit-subordinate.

NOTE – The Reversed Branch parameter can only be "true" if the Dynamic Commitment functional unit was selected on the original association.

7.9.2.5 User Data

This parameter may carry an unlimited amount of information determined by the referencing specification. It may contain one or more presentation data values from presentation contexts in the defined context set when the C-RECOVER request primitive is issued.

NOTE – The referencing specification determines the use of this parameter. For example, it can repeat information carried by a CCR service primitive lost in the failure, or to provide an optional retry time in the case that the Recovery State parameter value is retry-later.

7.10 C-P-ERROR service

7.10.1 Purpose and use

7.10.1.1 C-P-ERROR is a provider-initiated service. The CCR service-provider issues the indication primitive to signal a CCR service-provider error condition (e.g. protocol error).

7.10.1.2 After a C-P-ERROR indication primitive is issued, the CCR service-user shall not issue any further CCR service request or response primitives. It shall not receive any further CCR service indication or confirm primitives.

7.10.2 C-P-ERROR parameter

Table 12 lists the C-P-ERROR service parameter.

Table 12 – C-P-ERROR parameter

Parameter name	Ind
Provider Reason	M

7.10.2.1 Provider Reason

This parameter indicates the reason for the error. It takes on of the following symbolic values:

- protocol-error; or
- local-error.

8 Sequencing information

The allowed sequences of CCR service primitives for an association are defined by the state tables in this clause. The state tables are presented in terms of events and states.

8.1 General

8.1.1 The sequences specified in this clause are for a CCR service-user using a single association for a single atomic action branch. However, the overlap of two branches can occur when a C-BEGIN request primitive is issued jointly with a C-COMMIT request primitive. The overlapped use of the C-RECOVER service is also allowed.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.1.2 Tables 13 to 15 define the elements used in the state tables:

- Table 13 specifies the identifier and description for each state used for a branch that is not being recovered.
- Table 14 specifies the events.
- Table 15 specifies the predicates.

Table 13 – States of CCR ASE

Name	Abbreviation	Description
A1	bgn>	C-BEGIN req issued
A13	bgn> <bga (stat)	C-BEGIN req issued
A2	<bgn	C-BEGIN ind received
A23	<bgn bga> (stat)	C-BEGIN ind received
A3	bgn*	C-BEGIN service completed
A4	bgn>prp>	C-BEGIN req and C-PREPARE req issued
A5	bgn* prp>	C-BEGIN service completed and C-PREPARE req issued
A6	<bgn <prp	C-BEGIN ind and C-PREPARE ind received
A7	bgn* <prp	C-BEGIN service completed and C-PREPARE ind received
A8	prp> <prp	C-PREPARE req issued and C-PREPARE ind received
B1	bgn> rdy>	C-BEGIN req and C-READY req issued
B2	bgn> prp> rdy>	C-BEGIN req, C-PREPARE req and C-READY req issued
B3	bgn* rdy>	C-BEGIN service completed and C-READY req issued
B4	bgn* prp> rdy>	C-BEGIN service completed, C-PREPARE req and C-READY req issued
B5	<prp rdy>	C-PREPARE ind received and C-READY req issued
B6	<prp prp> rdy>	C-PREPARE and C-READY req issued, C-PREPARE ind received
C1	<rdy	C-READY ind received
D1	rdy> <rdy	C-READY req issued then C-READY ind received
E1	<cmt	C-COMMIT ind received
E2	cmt>	C-COMMIT req issued
F1	rbk>	C-ROLLBACK req issued
F2	<rbk	C-ROLLBACK ind received
F3	<rdy rbk>	C-READY ind received then C-ROLLBAACK ind issued
G1	<cmtbg	C-COMMIT+C-BEGIN ind received
G2	cmtbg>	C-COMMIT+C-BEGIN req issued
G3	cmtbg> prp>	C-COMMIT+C-BEGIN req issued then C-PREPARE req issued
G4	<cmtbg <prp	C-COMMIT+C-BEGIN ind received then C-PREPARE ind received
G5	cmtbg> rdy>	C-COMMIT+C-BEGIN req issued then C-READY req issued
G6	<cmtbg<rdy	C-COMMIT+C-BEGIN ind received then C-READY ind received
G7	cmtbg> can>	C-COMMIT+C-BEGIN req issued then C-CANCEL req issued
G8	<cmtbg <can	C-COMMIT+C-BEGIN ind received then C-CANCEL ind received
H1	rbkbg>	C-READY ind not received; C-ROLLBACK+C-BEGIN req issued
H2	<rbkbg	C-ROLLBACK+C-BEGIN ind received
H3	<rdy rbkbg>	C-READY ind received then C-ROLLBACK+C-BEGIN req issued
H4	rbkbg> <rbk	C-ROLLBACK+C-BEGIN req issued then C-ROLLBACK ind received
I	Idle	No action in progress

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 13 *(concluded)*

Name	Abbreviation	Description
J1	bgn> o-p>	C-BEGIN req then C-ONE-PHASE req issued
J2	bgn* o-p>	C-BEGIN service completed and C-ONE-PHASE req issued
J3	o-p> <prp	C-ONE-PHASE req issued then C-PREPARE ind received
J4	o-p><rdy	C-ONE-PHASE req issued then C-READY ind received
K1	<o-p	C-ONE-PHASE ind received
L1	r-o>	C-READ-ONLY req issued
M1	can>	C-CANCEL req issued
M2	<can	C-CANCEL ind received
R1	Rcmt>	C-RECOVER(commit) req issued
R2	<Rrdy	C-RECOVER(ready) ind received
R3	Rrdy>	C-RECOVER(ready) req issued
R4	<Rcmt	C-RECOVER(commit) ind received
S0	void	No association, CCR-sp void
S1	ini>	C-INITIALIZE req issued
S2	<ini	C-INITIALIZE ind received
X	error	Protocol error has been detected

Table 14 – Events

Name	Source	Description
BEGINcnf	CCR-sp	C-BEGIN confirm primitive
BEGINind	CCR-sp	C-BEGIN indication primitive
BEGINreq	CCR-su	C-BEGIN request primitive
BEGINrsp	CCR-su	C-BEGIN response primitive
CMT+BGNind	CCR-su	C-COMMIT indication primitive with a C-BEGIN indication primitive
CMT+BGNreq	CCR-su	C-COMMIT request primitive with a C-BEGIN request primitive
CMTcnf	CCR-sp	C-COMMIT confirm primitive
CMTind	CCR-sp	C-COMMIT indication primitive
CMTreq	CCR-su	C-COMMIT request primitive
CMTrsp	CCR-su	C-COMMIT response primitive
Disrupt	ACSE or CCR-su	A-ABORT indication, A-P-ABORT indication or A-ABORT request
INITcnf	CCR-sp	C-INITIALIZE confirm primitive
INITind	CCR-sp	C-INITIALIZE indication primitive
INITreq	CCR-su	C-INITIALIZE request primitive
INITrsp	CCR-su	C-INITIALIZE response primitive
NOCHANGEcnf	CCR-sp	C-NOCHANGE confirm primitive
NOCHANGEind	CCR-sp	C-NOCHANGE indication primitive
NOCHANGEreq	CCR-su	C-NOCHANGE request primitive
NOCHANGErsp	CCR-su	C-NOCHANGE response primitive
PREPAREind	CCR-sp	C-PREPARE indication primitive
PREPAREreq	CCR-su	C-PREPARE request primitive
RCV(commit)ind	CCR-sp	C-RECOVER indication primitive (Recovery State = "commit")
RCV(commit)req	CCR-su	C-RECOVER request primitive (Recovery State = "commit")

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 14 *(concluded)*

Name	Source	Description
RCV(done)cnf	CCR-sp	C-RECOVER confirm primitive (Recovery State = "done")
RCV(done)rsp	CCR-su	C-RECOVER response primitive (Recovery State = "done")
RCV(ready)ind	CCR-sp	C-RECOVER indication primitive (Recovery State = "ready")
RCV(ready)req	CCR-su	C-RECOVER request primitive (Recovery State = "ready")
RCV(retry-later)cnf	CCR-sp	C-RECOVER confirm primitive (Recovery State = "retry-later")
RCV(retry-later)rsp	CCR-su	C-RECOVER response primitive (Recovery State = "retry-later")
RCV(unknown)cnf	CCR-sp	C-RECOVER confirm primitive (Recovery State = "unknown")
RCV(unknown)rsp	CCR-su	C-RECOVER response primitive (Recovery State = "unknown")
READYind	CCR-sp	C-READY indication primitive
READYreq	CCR-su	C-READY request primitive
ROLLBACKcnf	CCR-sp	C-ROLLBACK confirm primitive
ROLLBACKind	CCR-sp	C-ROLLBACK indication primitive
ROLLBACKreq	CCR-su	C-ROLLBACK request primitive
ROLLBACKrsp	CCR-su	C-ROLLBACK response primitive

Table 15 – Predicates

Code	Description
pdy	Dynamic commitment functional unit is selected.
pnc	The No-change completion functional unit is selected.
pcan	The Cancel functional unit is selected.
prcl	The Ready-collision-reservation parameter on the C-INITIALIZE request or response (whichever was sent) had the value "true" or the parameter was absent. NOTE 1 – An implementation can ensure that prcl is always true if it never issues a C-INITIALIZE request or response with the Ready-collision-reservation parameter "false".
prcr	The Ready-collision-reservation parameter on the C-INITIALIZE indication or confirm (whichever was received) had the value "true" or the parameter was absent. NOTE 2 – A referencing specification can ensure that prcr is always true when CCR is used with that referencing specification it if requires that a C-INITIALIZE request or response is never issued with the Ready-collision-reservation parameter "false".

8.1.3 Tables 16 to 23 specify the State Table. The tables are presented separately for convenience and clarity. They utilize the abbreviated names and identifiers of Tables 13 to 15. Tables 16 to 23 show the states and events for the CCR ASE:

- Table 16 shows the states and events that may occur during initialization, when the CCR ASE is in the idle state and following an error.
- Table 17 shows the states and events that may occur in phase I. This includes initiation of rollback, a ready signal, the beginning of a no-change completion procedure, or failure.
- Table 18 shows the states and events that may occur from sending a ready signal to the ordering of commitment or initiation of rollback.
- Table 19 shows the states and events that may occur from receiving a ready signal to the ordering of commitment or initiation of rollback.
- Table 20 shows the states and events that may occur following the initiation of cancel or rollback.
- Table 21 shows the states and events that may occur following a commit order.
- Table 22 shows the states and events that may occur during and following no-change completion.
- Table 23 shows the states and events that may occur during the recovery procedure.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 16 – State table – Initialization, idle and error

Incoming event	Predicate	Preceding state				
		S0 void	S1 ini>	S2 <ini	I idle	X error
INITreq		S1				
INITind		S2				
INITrsp				I		
INITcnf			I			
BEGINreq					A1	
BEGINind					A2	
RCV(commit)req					R1	
RCV(commit)ind					R4	
RCV(ready)req					R3	
RCV(ready)ind					R2	
DISRUPT			S0	S0	S0	S0

Table 17 – State table – Phase I

Incoming event	Predicate	Preceding state									
		A1 bgn>	A2 <bgn	A13 bgn> <bga (stat)	A23 <bgn bga> (stat)	A3 bgn*	A4 bgn> prp>	A5 bgn* prp>	A6 <bgn <prp	A7 bgn* <prp	A8 prp> <prp
BEGINrsp			pdy A3 ~pdy A23						A7		
BEGINcnf		pdy A3 ~pdy A13					A5				
PREPAREreq		A4	pdy A5	A5		A5			pdy A8	pdy A8	
PREPAREind		pdy A7	A6		A7	A7	pdy A8	pdy A8			
READYreq		pdy B1	B3		B3	B3	pdy B2	pdy B4	B5	B5	B6
READYind		C1	pdy C1	C1		C1	C1	C1	pdy C1	pdy C1	C1
ROLLBACKreq		F1	F1	F1	F1	F1	F1	F1	F1	F1	F1
ROLLBACKind		F2	F2	F2	F2	F2	F2	F2	F2	F2	F2
CANCELreq	pcn	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1
CANCELind	pcn	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2
NOCHANGEreq	pnc	J1	J1	J1	J1	J1	J1	J1	J1	J1	J1
NOCHANGEind	pnc	K1	K1	K1	K1	K1	K1	K1	K1	K1	K1
DISRUPT		S0	S0	S0	S0	S0	S0	S0	S0	S0	S0

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 18 – State table – After sending a ready signal

Incoming event	Predicate	Preceding state					
		B1 bgn> rdy>	B2 bgn> prp> rdy>	B3 bgn* rdy>	B4 bgn* prp> rdy>	B5 <prp rdy>	B6 <prp prp> rdy>
PREPAREind		 B5	pdy B6	 B5	pdy B6		
READYind		pdy D1	 D1	pdy D1	 D1	pdy D1	 D1
ROLLBACKind		F2	F2	F2	F2	F2	F2
CANCELind	pcn	M2	M2	M2	M2	M2	M2
COMMITind		E1	E1	E1	E1	E1	E1
CMT+BGNind		E2	E2	E2	E2	E2	E2
NOCHANGEind	pnc	K1	K1	K1	K1	K1	K1
DISRUPT		S0	S0	S0	S0	S0	S0

Table 19 – State table – After receiving a ready signal

Incoming event	Predicate	Preceding state	
		C1 <rdy	D1 rdy> <rdy
ROLLBACKreq		 F3	prcl F3
ROLLBACKind			prcr F2
CANCELreq	pcn	M1	
COMMITreq		G1	G1
COMMITind			E1
CMT+BGNreq		G2	G2
CMT+BGNind			E2
NOCHANGEreq	pnc	J4	
DISRUPT		S0	S0

Table 20 – State table – After cancel or rollback

Incoming event	Predicate	Preceding state				
		M1 can>	M2 <can	F1 rbk>	F2 <rbk	F3 <rdy rbk>
ROLLBACKreq		F1	F1			
ROLLBACKind		F2	F2	F2		
ROLLBACKrsp					I	
ROLLBACKcnf				I		I
CANCELind	pcn	M2				
DISRUPT		S0	S0	S0	S0	S0

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 21 – State table – After commit order

Incoming event	Predicate	Preceding state			
		E1 <cmt	E2 <cmtbg	G1 cmt>	G2 cmtbg>
COMMITrsp		I	A2		
COMMITcnf				I	A1
DISRUPT		S0	S0	S0	S0

Table 22 – State table – No-change completion

Incoming event	Predicate	Preceding state	
		J n-c>	K1 <n-c
BEGINreq			A1
BEGINind		A2	
ROLLBACKreq			F1
ROLLBACKind		F2	
CANCELind	pcn	M2	
NOCHANGErsp			I
NOCHANGEcnf		I	
RCV(commit)req			R1
RCV(commit)ind		R4	
RCV(ready)req			R3
RCV(ready)ind		R2	
DISRUPT		S0	S0

Table 23 — State table — recovery

Incoming event	Predicate	Preceding state			
		R1 Rcmt>	R2 <Rrdy	R3 Rrdy>	R4 <Rcmt
RCV(commit)req			R1		
RCV(commit)ind				R4	
RCV(done)rsp					I
RCV(done)cnf		I			
RCV(unknown)rsp			I		
RCV(unknown)cnf				I	
RCV(retry-later)rsp			I		I
RCV(retry-later)cnf		I		I	
DISRUPT		S0	S0	S0	S0

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2 Events

8.2.1 In the state tables, the events are:

a) a CCR request or response primitive issued by the CCR service-user to the CCR ASE; or

b) a CCR indication or confirm primitive issued by the CCR ASE to the CCR service-user; or

c) two CCR request primitives jointly issued by the CCR service-user to the CCR ASE; or

d) two CCR indication primitives jointly issued by the CCR ASE to the CCR service-user.

8.2.2 The events are listed in Table 14. The joint issue of CCR service primitives that are not shown as an event are treated as the consecutive occurrence of the individual events in the table.

8.3 States

The CCR ASE states used in the state tables are represented by the notation Zn assigned in Table 13, where Z is an upper-case letter and n is null or is an integer.

The description of the states in the tables includes the abbreviated name for the state. This abbreviated name appears under the "Zn" form in Tables 16 to 23.

8.4 Predicates

Predicates used in the state tables are represented by the notation "pxxx" assigned in Table 15, where "p" is itself and "xxx" is two or three alphanumerics. Predicates are boolean values and can be combined using the operators "|" for OR, "&" for AND and "~" for NOT.

Predicate expressions appear either in the "Pred" column or in an individual cell. A predicate in the "Pred" column applies in all states for the appropriate event. If the predicate expression in the "Pred" column, or the only predicate expression in a cell evaluates to false, the cell shall be treated as if it were blank. If the expression evaluates to true, the state transition specified applies.

Some cells contain two action lists, each consisting of a predicate and a state. The predicates in such cells are always complements of each other. The action list whose predicate evaluates to true shall be applied.

8.5 Interpretation of the state table

8.5.1 General

8.5.1.1 A CCR ASE is initialized in state "S0" (see Table 16) when the ACSE association establishment service procedure (A-ASSOCIATE) is invoked. After this procedure, the CCR ASE is in state "I". This may occur as a result of including a C-INITIALIZE request primitive as user information on the A-ASSOCIATE request primitive. Otherwise, by default, the state changes to "I" at the conclusion of the association establishment procedure.

8.5.1.2 In the state tables, the intersection of an event (row) and a state (column) forms a cell. A non-blank cell represents the combination of an event and state that is defined for the CCR ASE. When such an intersection occurs, the state of the CCR ASE is changed to the state specified in the cell.

8.5.1.3 A blank cell represents the combination of an event and a state that is not defined for the CCR ASE. When such an intersection occurs, the CCR service-provider issues a C-P-ERROR indication primitive to its CCR service-user. Any action taken by the CCR service-user is a local matter.

8.6 Completing the branch

For a CCR service-user, a branch is completed by any of the following:

a) C-COMMIT response/confirm primitive; or

b) C-ROLLBACK response/confirm primitive; or

c) C-RECOVER(done) response/confirm primitive; or

d) C-RECOVER(unknown) confirm primitive; or

e) C-NOCHANGE request/indication primitive with Confirmation parameter of "not-required"; or

f) C-NOCHANGE response/confirm primitive; or

g) application or communication failure before a C-READY or C-COMMIT request primitive has been issued.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.7 Collisions and disruptive services

8.7.1 Application semantics in transit on the underlying session-connection can be lost by using the C-ROLLBACK service, if the corresponding indication or confirm primitive has not already been issued.

8.7.2 No CCR services other than C-ROLLBACK are disruptive.

8.7.3 The requirement that the requestor of the C-BEGIN or the C-RECOVER service hold the synchronize-minor token prevents service collisions between C-BEGIN and C-RECOVER.

8.7.4 A commit-superior may reply with a C-RECOVER(commit) request primitive to a C-RECOVER(ready) indication primitive without holding the synchronize-minor token, but a collision is not possible.

8.7.5 It is possible that a commit-tree may have two commit coordinators. This can happen when two CCR service-users have signalled readiness to each other. They are neighbours and share a common branch. A C-READY service collision occurs if each CCR service-user signals readiness to the other.

NOTE – A referencing specification can specify the use of CCR in a manner that does not allow a C-READY service collision.

8.7.6 If a C-READY service collision occurs, the atomic action will invariably proceed to commitment if both of the "Ready Collision Reserved" parameters on the C-INITIALIZE primitives are "false" (see 6.1.6), except if the supporting association fails, in which case the result depends on the recovery procedures.

9 Using CCR

9.1 General

9.1.1 For a given association, CCR services may be used at any time during the sequence of any other ASE service or the presentation service, except as defined below.

9.1.2 The CCR ASE uses its own presentation context to separate its semantics from the semantics of other ASEs using the same association. CCR User Data parameters (if present) are carried as one or more presentation data values that are passed directly to and from the presentation-service.

9.2 Use of CCR with non-reference mapping

9.2.1 CCR may be used as a component of an ASO that defines a non-reference mapping to supporting services, i.e. that is different from the mapping specified in the body of ITU-T Rec. X.852 | ISO/IEC 9805-1. Annex B of ITU-T Rec. X.852 | ISO/IEC 9805-1 defines the operating characteristics for the use of CCR in such an ASO.

9.2.2 Restrictions on token usage defined in this Recommendation | International Standard may not apply to the use of CCR as a component of an ASO that uses a non-reference mapping. The ASO specification is responsible for preventing collision situations that are not covered by CCR sequencing rules (see clause 8).

9.3 Use of session synchronization and resynchronization services

9.3.1 CCR services cannot be used by a referencing specification that uses session resynchronization in a manner unrelated to CCR semantics. In particular, a referencing specification may only use session resynchronization in circumstances where there is no possibility of the resynchronization disrupting any CCR service procedures other than the C-READY request primitive, the C-PREPARE request primitive or the C-BEGIN response primitive.

NOTE – For example, a referencing specification may use the P-RESYNCHRONIZE(restart) or P-RESYNCHRONIZE (set) services before the end of Phase 1.

9.3.2 A referencing specification may use session synchronization points (minor or major). However, the referencing specification must be aware that CCR also uses session synchronization points.

9.4 Use of CCR with session activities

The CCR services cannot be used outside of a session activity if the session activity management functional unit was selected for the supporting association.

9.5 Use of presentation services

The C-BEGIN request primitive cannot be issued if a P-ALTER-CONTEXT confirm primitive is pending and the Context Restoration functional unit of Presentation is selected.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A

CCR service-user rules

(This annex forms an integral part of this Recommendation | International Standard)

A.1 Introduction

This annex defines CCR service-user rules that shall be incorporated by a referencing specification. In addition to these rules, a referencing specification may include more restrictive rules provided they do not conflict with the rules specified in this annex.

A.1.1 Rule categories

Four categories of CCR service-user rules are defined in this annex:

a) *CCR service primitive usage rule* – A constraint on the use of a CCR service primitive for a branch of an atomic action. CCR service primitive usage rules are defined in A.3.

b) *Atomic action data manipulation rule* – The conditions required for a CCR service-user to record or forget atomic action data for a branch. Atomic action data manipulation rules are defined in A.4.

c) *Bound data manipulation rule* – A constraint on a CCR service-user for its manipulation of bound data. Bound data manipulation rules are defined in A.5.

d) *CCR service-user data transfer rule* – A constraint on a CCR service-user for its use of presentation data transfer service primitives that request the manipulation of bound data. CCR service-user data transfer rules are defined in A.6.

A.2 Compliance

A.2.1 To be compliant with this Recommendation | International Standard, a referencing specification shall include provisions a) and b) below or it shall require referencing specifications that are compliant to it to include provisions a) and b).

A.2.2 The provisions for compliance are:

a) the incorporation, by reference or inclusion, of the CCR service-rules defined in this annex; and

b) the inclusion of conformance requirements to the CCR service-user rules defined in this annex in any requirements that are to be met by an implementation conformance to the specification.

A.3 CCR service primitive usage rules

A CCR service primitive usage rule is a constraint on the use of a CCR service primitive for a branch of an atomic action.

Five types of CCR service primitive usage rules are defined:

a) *multi-branch sequence rule* – A constraint based on the previous issue or receipt of CCR service primitives on the other branches of this atomic action for this CCR service user.

b) *multi-branch recovery rule* – A constraint based on either the previous recording or forgetting of atomic action data for the branches of this atomic action for this CCR service-user. The atomic action data are used for recovery.

c) *single branch recovery rule* – A constraint based on either the previous recording or forgetting of atomic action data for this branch. The atomic action data are used for recovery.

d) *bound data rule* – A constraint based on the release of bound data in either the initial or final state.

e) *association use rule* – A constraint based on owning a session token for the underlying session-connection. These rules are not applicable when CCR is used with a non-reference mapping (see 9.2).

A.3.1 C-INITIALIZE request primitive

The C-INITIALIZE request primitive shall be issued jointly with an A-ASSOCIATE request primitive (association use rule – Rule A.3.1).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.3.2 C-INITIALIZE response primitive

A C-INITIALIZE response primitive shall be issued jointly with the A-ASSOCIATE response primitive (association use rule – Rule A.3.2).

A.3.3 C-BEGIN request primitive

A C-BEGIN request primitive used to begin a branch in an atomic action graph cannot be issued if:

a) a C-READY request primitive has been issued on another branch of that atomic action [multi-branch sequence rule – Rule A.3.3 a)]; or

b) a C-ROLLBACK request primitive has been issued to the branch-initiator [multi-branch sequence rule – Rule A.3.3 b)]; or

c) a C-COMMIT request primitive has been issued to a commit-subordinate [multi-branch sequence rule – Rule A.3.3 c)]; or

d) the atomic action branch identifier has already been used for a different branch within the same atomic action [multi-branch sequence rule – Rule A.3.3 d)]; or

e) the CCR service-user does not own the synchronize-minor token unless the primitive is jointly issued with a C-COMMIT request primitive for another atomic action [association use rule – Rule A.3.3 e)]; or

f) a C-NOCHANGE request primitive has been issued (multi-branch sequence rule – Rule A.3.3 f)].

A.3.4 C-PREPARE request primitive

A C-PREPARE request primitive has no CCR service-user rules controlling its use.

A.3.5 C-READY request primitive

A C-READY request primitive can only be issued if:

a) the CCR service-user has received a C-READY or C-RECOVER(ready) indication primitive from other continuing two-phase neighbours [multi-branch sequence rule – Rule A.3.5 a)]; and

b) atomic action data have been recorded that are required for a ready signal as defined in A.4.1 [multi-branch and single branch recovery rule – Rule A.3.5 b)]; and

c) bound data can be released in either the initial or the final state [bound data rule – Rule A.3.5 c)].

A.3.6 C-COMMIT request primitive

A.3.6.1 A C-COMMIT request primitive can only be issued if either a) or b) is true, and in addition c) is true:

a) the CCR service-user has received a C-COMMIT or C-RECOVER(commit) indication [multi-branch sequence rule – Rule A.3.6.1 a)]; and

b) all of the following are true:

1) the CCR service-user has received a C-READY or C-RECOVER(ready) indication primitive from all continuing two-phase neighbours [multi-branch sequence rule – Rule A.3.6.1 b), 1)]; and

2) atomic action data have been recorded that are required for an order of commitment as defined in A.4.3.2 [multi-branch recovery rule – Rule A.3.6.1 b), 2)]; and

3) bound data can be released in the final state for a master [bound data rule – Rule A.3.6.1 b), 3)]; and

c) the CCR service-user owns the synchronized-minor token if CCR Protocol Version 2 is being used [association use rule – Rule A.3.6.1 c)].

A.3.6.2 If a C-COMMIT request primitive is issued by a commit-decider, the commit-decider shall release bound data in the final state (bound data rule – Rule A.3.6.2).

A.3.7 C-COMMIT response primitive

A C-COMMIT response primitive can only be issued if:

a) the CCR service-user has recorded atomic action data indicating an order to commit for all branches to commit-subordinates for which it has not received a C-COMMIT or C-RECOVER(done) confirm primitive [combination multi-branch sequence rule and multi-branch recovery rule – Rule A.3.7 a)]; and

b) bound data have been released in the final state [bound data rule – Rule A.3.7 b)]; and

c) atomic action data for this branch have been forgotten [single branch recovery rule – Rule A.3.7 c)]; and

d) atomic action data for all its subordinates have been forgotten or the atomic action data have been changed to indicate an order to commit [multi-branch recovery rule – Rule A.3.7 d)].

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Rule A.3.7 c) can be ignored if the other parts of rule A.3.7 are true and if Rule A.3.7 b) will remain true regardless of subsequent failures and recovery indication or confirm primitives.

NOTE – The relaxation of Rule A.3.7 c) requires, for example, that a failure followed by recovery, with reinstatement of lock of bound data, and a subsequent C-RECOVER confirm primitive does not cause release in the initial state.

A.3.8 C-ROLLBACK request primitive

A.3.8.1 A C-ROLLBACK request primitive can only be issued if:

a) a C-READY or C-RECOVER(ready) request primitive has not been issued [multi-branch sequence rule – Rule A.3.8.1 a)]; or

b) a C-ROLLBACK indication primitive or a C-RECOVER(unknown) confirmation primitive or C-CANCEL indication primitive has been received from the commit-superior [multi-branch sequence rule – Rule A.3.8.1 b)]; or

c) a C-READY request primitive has been issued and a C-READY indication has been received on the same branch and the "Ready-collision-reservation" on the C-INITIALIZE request or response (whichever was sent) on the association that supported the branch was absent or had the value "true" [multi-branch sequence rule – Rule A.3.8.1 c)]; or

d) a C-READY or C-RECOVER(ready) request primitive has been issued and C-RECOVER(ready) indication has been received [multi-branch sequence rule – Rule A.3.8.1 d)].

If Rule A.3.8.1 d) is to be used for branches originally supported by an association on which the "Ready-collision-reservation" parameter of the C-INITIALIZE request or response was "false", the referencing specification shall ensure that consistent decisions are taken by the CCR service users.

A.3.8.2 If a C-ROLLBACK request primitive is issued to the branch initiator, bound data shall be released in the initial state (bound data rule – Rule A.3.8.2).

A.3.9 C-ROLLBACK indication primitive

If C-ROLLBACK indication primitive is received from the branch initiator (in phase I) or from the commit-superior (in phase II), then one of the following must be true:

a) bound data shall be released in the initial state; or

b) a heuristic mix is detected.

(bound data rule – Rule A.3.9).

A.3.10 C-CANCEL request primitive

A C-CANCEL request primitive can only be issued if the conditions for issuing a C-ROLLBACK request primitive, as specified in A.3.8 are satisfied, that is rules A.3.8.1 a), A.3.8.1 b), A.3.8.1 c) and A.3.8.2 apply.

A.3.11 C-CANCEL indication primitive

If C-CANCEL indication primitive is received from the branch initiator (in phase I) or from the commit-superior (in phase II), then rule A.3.9 applies, as for a C-ROLLBACK indication primitive.

A.3.12 C-NOCHANGE request primitive

A C-NOCHANGE request primitive can only be issued if:

a) the CCR service-user has no continuing two-phase branches other than the one the C-NOCHANGE request is to be issued on [multi-branch sequence rule – Rule A.3.12 a)]; and

b) no bound data has been modified [bound data rule – Rule A.3.12 b)].

A.3.13 C-NOCHANGE response primitive

A C-NOCHANGE response primitive has no CCR service-user rules controlling its use.

A.3.14 C-RECOVER(ready) request primitive

A C-RECOVER(ready) request primitive can only be issued if:

a) the CCR service-user has received a C-READY or C-RECOVER(ready) indication primitive from other continuing two-phase neighbours [multi-branch sequence rule – Rule A.3.14 a)]; and

b) atomic action data have been recorded that are required for a ready signal as defined in A.4.1 [multi-branch and single branch recovery rule – Rule A.3.14 b)]; and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) bound data can be released in either the initial or the final state [bound data rule – Rule A.3.14 c)]; and

d) either the overlapped-recovery functional unit is selected on the association or the CCR service-user is the owner of the synchronize-minor token [association use rule – Rule A.3.14 d)].

A.3.15 C-RECOVER(commit) request primitive

A.3.15.1 A C-RECOVER(commit) request primitive can only be issued if either a) or b) is true and c) is true:

a) the CCR service-user has received a C-COMMIT or C-RECOVER(commit) indication primitive (multi-branch sequence rule – Rule A.3.15.1 a); and

b) all of the following are true:

1) the CCR service-user has received a C-READY or C-RECOVER(ready) indication primitive from all continuing two-phase neighbours [multi-branch sequence rule – Rule A.3.15.1 b), 1)]; and

2) atomic action data required to perform recovery have been recorded for all continuing two-phase neighbours(multi-branch recovery rule – Rule A.3.15.1 b), 2)]; and

3) bound data can be released in the final state [bound data rule – Rule A.3.15.1 b), 3)]; and

c) either the overlapped-recovery functional unit is selected on the association or the CCR service-user is the owner of the synchronize-minor token unless the request primitive is in answer to a C-RECOVER(ready) indication from a commit-subordinate (association use rule – Rule A.3.15.1);

d) either the overlapped-recovery functional unit is selected on the association or the CCR service-user is the owner of the synchronize-minor token or the primitive is in answer to a C-RECOVER(ready) indication from a commit-subordinate [association use rule – Rule A.3.15.1 e)].

A.3.15.2 If a C-RECOVER(commit) request primitive is issued by a commit-decider, the commit-decider shall release bound data in the final state (bound data rule – Rule A.3.15.2).

A.3.16 C-RECOVER(done) response primitive

A C-RECOVER(done) response primitive can only be issued if:

a) the CCR service-user has recorded atomic action data indicating an order to commit for all branches to commit-subordinates for which it has not received a C-COMMIT or C-RECOVER(done) confirm primitive [combination multi-branch sequence rule and multi-branch recovery rule – Rule A.3.16 a)]; and

b) bound data have been released in the final state or a heuristic mix has been detected (bound data rule – Rule A.3.16 b); and

c) atomic action data for this branch have been forgotten [single branch recovery rule – Rule A.3.16 c)]; and

d) atomic action data for all its branches to commit-subordinates have been forgotten or the atomic action data have been changed to indicate an order to commit [multi-branch recovery rule – Rule A.3.16 d)].

A.4 Atomic action data manipulation rules

An atomic action data manipulation rule is a set of conditions required for a CCR service-user to record or forget atomic action data for a branch. A CCR service-user uses the atomic action data to maintain recovery responsibility information for the branch.

The atomic action data required by the CCR service-user to perform recovery for a branch includes:

a) atomic action identifier;

b) branch identifier;

c) whether the CCR service-user was the branch-initiator or branch-responder;

d) Recovery Context data from the C-INITIALIZE service;

e) whether the CCR service-user has decided to send a ready signal or order commitment;

f) information required to re-establish an association with the neighbour; and

g) any other information needed for recovery with the neighbour.

A.4.1 Recording atomic action data that sending a ready signal

Atomic action data indicating sending a ready signal can only be recorded if:

a) Rule A.3.5 a) is satisfied; and

b) Rule A.3.5 c) is satisfied.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The following atomic action data are recorded by a CCR service-user for sending a ready signal:

a) atomic action data required to perform recovery for all its commit-subordinates that will not be rolled back; and

b) atomic action data required to perform recovery with the commit-superior.

A.4.2 Forgetting atomic action data that indicates sending a ready signal

Atomic action data indicating sending a ready signal can only be forgotten if:

a) a C-ROLLBACK indication primitive or C-RECOVER(unknown) confirm primitive is received from the commit-superior; or

b) a C-COMMIT indication primitive or a C-RECOVER(commit) indication primitive is received from the commit-superior and a C-COMMIT confirmation primitive or a C-RECOVER(done) confirm primitive has been received from all its commit-subordinates; or

c) a C-READY indication has been received from the commit-superior and the "Ready-collision-reservation" on the C-INITIALIZE request or response (whichever was sent) on the association that (originally) supported the branch was absent or had the value "true"; or

d) a C-RECOVER(ready) indication has been received from the commit-superior.

If rule d) is to be used for branches originally supported by an association on which the "Ready-collision-reservation" parameter of the C-INITIALIZE request or response was "false", the referencing specification shall ensure that consistent decisions are taken by the CCR service users.

A.4.3 Recording atomic action data indicating an order of commitment

A.4.3.1 Atomic action data indicating an order of commitment can only be recorded if:

a) Rule A.3.6.1 a) is satisfied; and

b) Rule A.3.6.1 b) is satisfied; and

c) Rule A.3.6.1 d) is satisfied.

A.4.3.2 The following atomic action data are recorded by a commit-decider for an order of commitment:

– atomic action data required to perform recovery for all its commit-subordinates that will not be rolled back.

A.4.4 Forgetting atomic action data indicating an order of commitment

Atomic action data indicating an order of commitment can only be forgotten if a C-COMMIT confirm primitive or a C-RECOVER(done) confirm primitive is received from each of its commit-subordinates.

A.5 Bound data manipulation rules

A bound data manipulation rule is a constraint on a CCR service-user for its manipulation of bound data.

A.5.1 Changing bound data to produce the final state

Changes to produce the final state of bound data through the normal progression of the atomic action require that:

a) atomic action data indicating sending a ready signal are not recorded; and

b) atomic action data indicating an order of commitment are not recorded.

A.5.2 Releasing bound data in the initial state as part of rollback

Release of bound data in the initial state as part of rollback requires that:

a) a C-ROLLBACK indication or C-RECOVER(unknown) confirm has been received from the commit-superior if sending a ready signal is recorded; and

b) atomic action data indicating an order of commitment are not recorded.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.5.3 Releasing bound data in the final state as part of commitment

Release of bound data in the final state as part of commitment requires that:

a) the CCR service-user has received a C-COMMIT or C-RECOVER(commit) indication primitive from the commit-superior if it is not the commit-decider; and

b) atomic action data required to perform recovery have been recorded for all commit-subordinate branches that will not be rolled back for a commit-decider; and

c) atomic action data reflect either sending a ready signal or an order of commitment.

A.5.4 Releasing bound data as part of a heuristic decision

Release of bound data as the result of a heuristic decision requires that atomic action data reflects sending a ready signal.

A.6 CCR service-user data transfer rules

A CCR service-user data transfer rule is a constraint on a CCR service-user for its use of presentation data transfer service primitives that request the manipulation of bound data.

A.6.1 Data transfer request and response primitive

A data transfer request or response primitive that requests the manipulation of bound data as part of an atomic action can only be issued on an association for a branch on which:

a) the CCR service-user has not received a C-READY or C-ROLLBACK indication primitive; and

b) the CCR service-user has not issued a C-PREPARE, C-READY, or C-ROLLBACK request primitive.

A.6.2 Data transfer indication and confirm primitive

A data transfer indication or confirm primitive that requests the manipulation of bound data as part of an atomic action can only be received on an association for a branch on which:

a) the CCR service-user has not received a C-READY, C-PREPARE, or C-ROLLBACK indication primitive; and

b) the CCR service-user has not issued a C-ROLLBACK request primitive; and

c) the CCR service-user has not recorded atomic action data for sending a ready signal.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B

Relationship of CCR to the Application Layer Structure

(This annex forms an integral part of this Recommendation | International Standard)

The material in this annex is aligned with ITU-T Rec. X.207 | ISO/IEC 9545, but uses the terminology of the first edition, ISO/IEC 9545:1989.

NOTE – Figure B.1 is only given as an illustration to the text.

B.1 CCR service-provider

The CCR service-provider is modelled to consist of two peer CCR ASEs involved in the same atomic action branch. Each CCR ASE is placed within the context of a Single Association Object (SAO) of the supporting association.

B.2 CCR service-user

B.2.1 The CCR service-user within the AEI consists of two parts:

a) the SAO part; and

b) the Multiple Association Control Function (MACF) part.

B.2.2 The SAO part consists of the SACF and one or more ASEs that use the CCR ASE services. Such an ASE is called a User application-service-element (U-ASE).

B.2.3 The MACF part of the CCR service-user represents the multiple association coordination function that is part of the CCR related activities.

NOTE – The MACF part is even needed for an atomic action that only consists of a single branch. Here, the MACF part is needed for recovery if an application or communication failure occurs.

B.3 Atomic action graph

Figure B.1 is an example of an atomic action graph based on this architecture.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

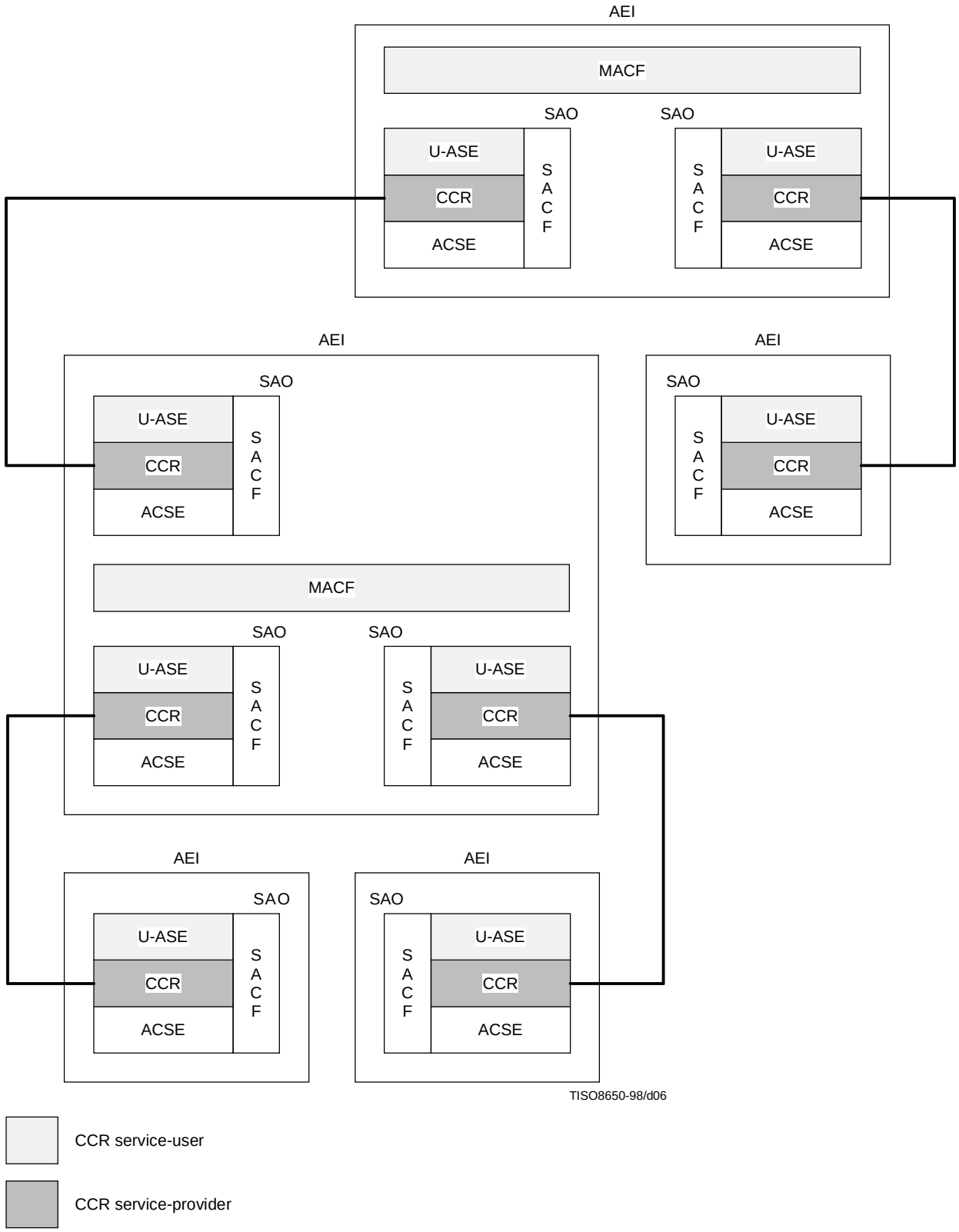

Figure B.1 – Atomic action tree architecture

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C

CCR tutorial

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

NOTE – This annex mostly describes CCR as specified in the previous (1993) edition of this Recommendation | International Standard. The terms used have been updated but the annex does not cover dynamic commitment, no-change termination or cancel.

C.1 Introduction

This tutorial describes the main functions of CCR and its use in a distributed environment. It expands, but, in general, it does not repeat the concepts presented in clause 6. Therefore, before you read this annex, you should be familiar with concepts and terminology presented in clause 6.

This tutorial mentions the CCR services defined in clause 7. You should be familiar with the "purpose and use" of each of the six CCR services:

- C-BEGIN;
- C-PREPARE;
- C-READY;
- C-COMMIT;
- C-ROLLBACK; and
- C-RECOVER.

NOTE – The C-INITIALIZE, C-NOCHANGE and C-CANCEL services are not discussed in this annex.

All the terms defined in the normative part of this Recommendation | International Standard are repeated in 3.6. Subclauses 3.1 through 3.5 list the terms used by CCR that are defined in other Recommendations and International Standards.

Since this annex is confined to static commitment, the terms "superior" is used to refer to both "branch-initiator" and "commit-superior". Similarly "subordinate" is used to refer to both "branch-responder" and "commit-subordinate".

CCR defines services for use on a single association. Therefore, CCR always requires a referencing (i.e. controlling) specification. This is true even if the use of CCR only involves two CCR service-users. In this case, the referencing specification would need to specify, as a minimum, the recovery actions when a failure occurs.

The use of CCR may be constrained by a particular referencing specification. That is, some of the features described here may not occur when used by a particular referencing specification.

C.1.1 What is CCR?

CCR is an application-service-element (ASE) plus rules for the use of the ASE. As an ASE, CCR has a service definition (this Recommendation | International Standard) and a protocol specification (ITU-T Rec. X.852 | ISO/IEC 9805-1). The usage rules (called CCR service-user rules) are indicated in Annex A.

CCR is like the Association Control Service Element (ACSE – ITU-T Rec. X.217 | ISO/IEC 8649 and ITU-T Rec. X.227 | ISO/IEC 8650-1) in many respects:

- ACSE facilities provide a relationship between two ACSE service-users for their use of a presentation-connection. This relationship is called an application-association, or, simply, an association.
- CCR facilities provide a relationship between two CCR service-users for their use of an association. This relationship is called an atomic action branch, or, simply, a branch.
- ACSE provides a bracketing facility. ACSE services establish and release an association. ACSE is unaware of the flow of semantics on the association between its establishment and release.
- CCR also provides a bracketing facility. CCR services begin and complete a branch. CCR is unaware of the ensuing flow of semantics on the branch.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

On the other hand, CCR and ACSE differ in many aspects:

- ACSE establishes the presentation-connection that it uses. This is done as part of the A-ASSOCIATE service that establishes the association. Later, ACSE releases the presentation-connection when the association is released.
- CCR requires the prior existence of the association that it uses. That is, when C-BEGIN is invoked, the association must already be in existence. At the completion of a branch, the association continues.
- For ACSE, if an application or communication failure occurs, its relationship (i.e. the association) is gone. That is, the association and the supporting presentation-connection are abnormally released. The association does not persist.
- For CCR, if an application or communication failure occurs, its relationship (i.e. the branch) may or may not persist. This is determined by the presumed rollback paradigm that CCR uses. This is discussed in 6.2.2, 7.6, and C.5.2. If the branch persists, it is recovered and completed by using another association.
- ACSE has no rules about how an association is used.
- CCR has rules about how a branch is used. Like ACSE, it has no concern about the semantics that flow on the branch. However, CCR assumes that its user provides the atomic action properties for its branches. These properties are:

  1) atomicity;
  2) consistency;
  3) isolation; and
  4) durability.

  They are discussed in 6.1.1.

C.1.2 Atomic action

For CCR, a distributed application is defined as an information processing endeavor that is accomplished by two or more application-process Invocations (APIs). The application-entity Invocations (AEIs – the OSI communication aspects of the APIs) are interconnected within the OSI environment by associations.

An atomic action is a specific, delimited sequence of operations of a distributed application. An atomic action may be characterized by the properties of atomicity, isolation, consistency, and durability. These are discussed in 6.1.1.

At the end of an atomic action, if the results of its operations happen in all of the involved AEIs, the atomic action has been committed. If the results do not happen, the atomic action has been rolled back.

An atomic action does not have an abnormal end. An atomic action ends by rollback (it never happened) or by commitment (it happened). Rollback may be the expected outcome.

C.1.3 Purpose

CCR provides facilities that allow an atomic action of a distributed application to commit or to roll back. CCR provides an ASE to support one branch (i.e. segment) of an atomic action. It also defines rules that must be incorporated by a referencing specification and followed by real implementations.

C.1.4 Using CCR

Annex B discusses how CCR fits into the Application Layer structure based on ITU-T Rec. X.207 | ISO/IEC 9545.

This CCR Recommendation and the CCR ITU-T Rec. X.852 | ISO/IEC 9805-1 define the CCR ASE. CCR does not define the Single Association Control Function (SACF) or Multiple Association Control Function (MACF). Something else must define these control functions. Throughout the normative parts of CCR, this something else is called the referencing specification. However, Annex A defines rules that must be included in the SACF and the MACF.

C.2 Structure of an atomic action tree

NOTE – Several of the terms used in this subclause are those used in the earlier edition of this Recommendation | International Standard, which covered only static commitment. Notes give the correspondence with the more precise terms used with dynamic commitment, used in the body of this Recommendation | International Standard. In summary:

atomic action tree = begin-tree + commit-tree;

superior = branch-initiator + commit-superior;

subordinate = branch-responder + commit-subordinate.

The old term "master" has been replaced by the appropriate equivalent – atomic action initiator, commit-coordinator, commit-decider – as appropriate in the context.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.2.1 Model

The model pulls together the concepts discussed so far (distributed application, atomic action, and branch) and adds the following:

- A CCR service-user is that part of an AEI that uses CCR services for one or more related branches of one atomic action. (More about "related branches" below.)
- Finally, an atomic action tree is a hierarchical relationship between CCR service-users of an atomic action. It is made up of CCR service-users and branches.

NOTE – In the absence of dynamic commitment, the begin-tree and the commit-tree are the same. The term "atomic action tree" used in this annex refers to this single arrangement of the atomic action graph.

Figure C.1 shows the model of an atomic action tree. The circles represent the CCR service-users. Lines between pairs of CCR service-users represent the branches of the atomic action.

The referencing specification identifies the ASEs to be used on the branches of an atomic action. Different ASEs can be used on the different branches of the atomic action.

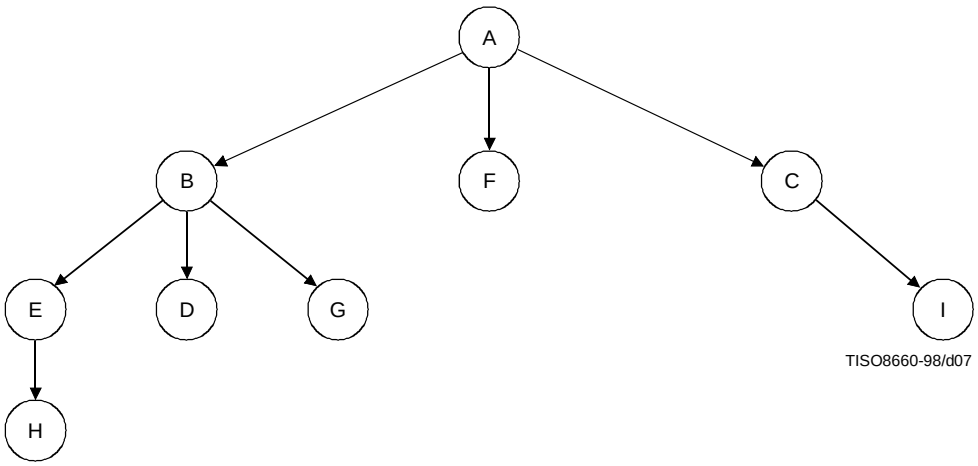

Figure C.1 – Atomic action tree

C.2.2 CCR service-user

A CCR service-user is part of an AEI. It uses CCR services to coordinate related branches of the same atomic action tree. The "related branches" are:

a) for an atomic action initiator, the branches to the subordinates;

b) for an intermediate, the branch to its superior plus the branches to its subordinates; and

c) for a leaf, the single branch to its superior.

NOTE — Because it is confined to static commitment, this annex uses the term "intermediate" to refer to CCR service-users that are branch-responder on one branch and branch-initiator on at least one other branch. This annex uses the term "leaf" to refer to CCR-service-users that have a single branch, on which they are branch-responder. Given the restrictions of static commitment, these terms are equivalent to the OSI TP use of the terms within a transaction tree (see IUT-T Rec. X.862 | ISO/IEC 10026-3).

A CCR service-user is involved in a single atomic action.

A given AEI may contain one or more CCR service-users. The CCR service-users within an AEI may all be a part of the same atomic actions or different atomic actions.

The CCR services (see clause 7) and their sequencing rules (see clause 8) apply independently to each CCR service-user. As mentioned above, an AEI may contain more than one CCR service-user involved in the same atomic action. In this case, it is not possible to distinguish which branch belongs to which CCR service-user from the CCR parameters alone. This relationship is defined by the referencing specification and the internal workings of the AEI.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A referencing specification may place restrictions on the creation of CCR service-users within an AEI. For example, a referencing specification may only allow one CCR service-user of the same atomic action to occur in a given AEI.

C.2.3 Branch and its identifiers

A branch is a relationship between two logically adjacent CCR service-users. The discussion on atomic action branches in 6.1.2 is quite thorough.

A branch has both an atomic action identifier and a branch identifier. The atomic action identifier is assigned by the owner of the atomic action. The branch identifier is assigned by the branch-initiator of the branch.

An atomic action identifier is made up of two parts: the Owner's Name and a suffix assigned by the owner. Although the Owner's Name has the form of an AE-title, and can be the AE-title of the atomic action initiator, it is required only to identify the number space for the suffix, such the combination is globally unambiguous. A branch identifier is made up of two parts: the AE title of the superior and a suffix assigned by the superior.

C.2.4 An example using JTM

An example of an atomic action tree that involves several branches occurs in the use of Job Transfer and Manipulation (JTM – ISO/IEC 8831). JTM references CCR for all transfers of JTM material (i.e. documents and reports). Figure C.2 shows an example of a JTM-based atomic action.

In Figure C.2 a), an initial transfer of JTM material takes place within a CCR atomic action branch between AEI A and AEI B. The initiating CCR service-user A-1 is the initiator of the atomic action. It acts as the superior for the branch. The receiving CCR service-user B-1 acts as the subordinate of the branch. At this point, B-1 is the single leaf of the atomic action tree.

After receiving and processing the material, B-1 immediately invokes further JTM activity to transfer generated documents and JTM reports. These transfers are performed on new branches of the same atomic action. Transfers take place between:

1) CCR service-user B-1 and CCR service-user A-2 in AEI A; and

2) between CCR service-user B-1 and CCR service-user C-1 in AEI C. This is shown in Figure C.2 b).

CCR service-user B-1, the subordinate of the first branch, is the superior of the new branches. B-1 is now an intermediate. CCR service-users A-2 and C-1 are leaves.

For JTM, CCR service-users A-1 and A-2 may be in the same or different invocations of the same AE. In Figure C.2 b), they are in the same AEI. The atomic action tree model of this example is shown in Figure C.2 c).

This second "wave" of processing may, in turn, cause further transfers of documents and reports. This adds a deeper level of branches to the atomic action tree. For JTM, this may continue to any depth.

This example illustrates two key features of atomic action trees:

a) an atomic action tree is dynamically built-up as the action proceeds; and

b) the actual structure of an atomic action tree depends on the referencing specification and on the application data.

C.3 CCR service-user information resources

Three classes of data may be used during a CCR atomic action:

a) bound data;

b) atomic action data; and

c) operational data.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

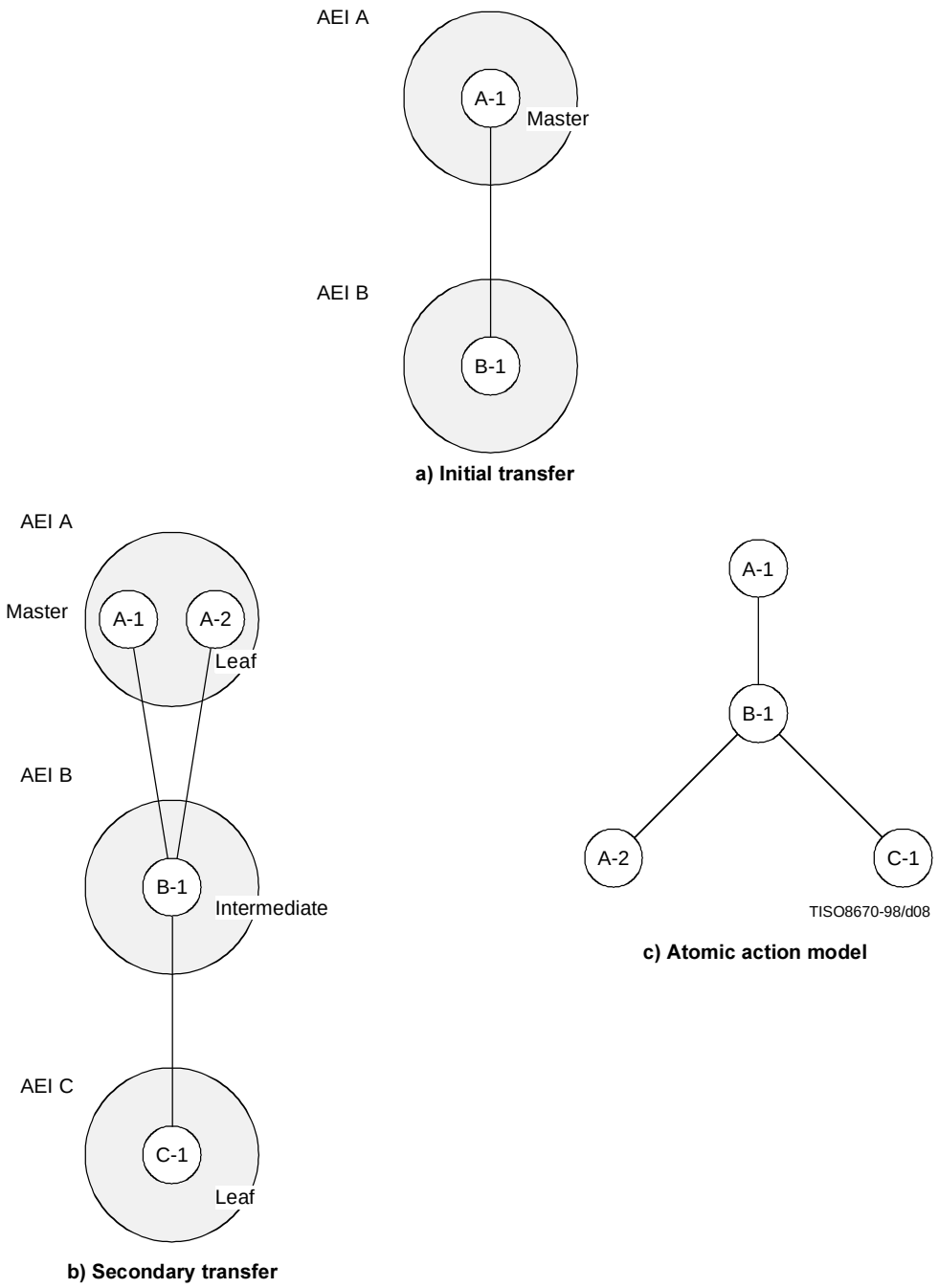

Figure C.2

C.3.1 Bound data

Bound data are a set of information resources on a CCR service-user's open system. The CCR service-user manipulates these information resources during an atomic action. Bound data are subject to commitment or rollback. The referencing specification identifies the bound data for a CCR service-user for a particular atomic action.

The information resources that are a part of bound data may change during the lifetime of an atomic action. An atomic action initiator's bound data start when the atomic action starts. That is, its bound data start when the first branch to a subordinate begins. However, the initial state of the bound data may reflect an even earlier state. An intermediate's or leaf's bound data start when its branch to the superior starts.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NOTE 1 – It is possible that a CCR service-user may have no "bound" information resources. In this case, bound data are null.

As new branches are started to subordinates, additional information resources at the superior's open system may be added as bound data. If a branch to a subordinate is rolled back, these resources may be removed from the bound data if they are not also "bound" to another branch.

NOTE 2 – The possibility of rolling back one branch of an atomic action but not all of the atomic action is permitted by CCR, but requires particular properties of the distributed bound data to avoid risking violation of the ACID properties.

Semantic flows occur between the CCR service-user and its logically adjacent peers on the branches of the atomic action. The flow of application-semantics[3)] changes the CCR service-user's bound data from its initial state to its final state.

The C-BEGIN request and indication primitives are needed to determine the initial state of bound data for possible rollback.

For commitment, the CCR service-user's bound data are placed in the final state at the end of the atomic action. For rollback, including presumed rollback caused by failure, the CCR service-user's bound data are restored to the initial state.

If an application or communication failure occurs, bound data must persist (i.e. not be lost). Their loss would cause the breakdown of the atomic action properties. Of course, during the operation of real systems, an unpredictable loss of data can occur. This Recommendation | International Standard assumes that the loss of bound data occurs infrequently.

C.3.2 Atomic action data

Atomic action data are used by the open system to maintain knowledge about a current atomic action. Atomic action data consist of state and control information about the CCR service-user and its branches. CCR specifies when atomic action data must persist if an application or communication failure occurs.

For CCR, atomic action data are an integral part of recovery. This topic is discussed further in C.5.1.

As with bound data, atomic action data required for recovery must persist if an application or communication failure occurs. Their loss would also cause the breakdown of the atomic action properties. This Recommendation | International Standard assumes that the loss of atomic action data occurs infrequently.

C.3.3 Operational data

Other information resources associated with a branch that are not bound data or atomic action data form the operational data for a CCR service-user. Operational data are like bound data in that they are manipulated during an atomic action. However, operational data are not subject to commitment or rollback.

Operational data are not identified by CCR, nor is their use specified. It is identified and mentioned here for completeness and to distinguish between operational data and bound data.

The referencing specification determines the classification of bound data and operational data. For example, the contents of a file being accessed would normally be regarded as bound data, and subject to commitment or rollback. The "date last accessed" attribute of the file would normally not be regarded as bound data. It would not be subject to commitment or rollback. Accounting data could also be defined as operational data. Processing and communications charges could occur whether an atomic action is committed or rolled back.

Unlike bound data and atomic action data required for recovery, the loss of operational data does not affect the atomic action properties. Their loss is not addressed by CCR.

C.4 Concurrency

CCR services do not provide a concurrency mechanism. However, the preservation of the atomic action properties requires that an implementation considers concurrency.

C.4.1 General considerations

Concurrency control takes place from the perspective of individual CCR service-users. A concurrency control mechanism provides the isolation property for each concurrent CCR service-user across an open system.

3) In this annex, the expression "application-semantics" refers to semantics exchanged as part of an atomic action branch that manipulate bound data.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A concurrency mechanism used in conjunction with CCR should include the following properties:

a) A CCR service-user does not offer commitment if its bound data are modified by an entity other than a CCR service-user.

b) Consider an information resource that is a member of the bound data of atomic action A. This information resource is in the final state for atomic action A but it has not yet been committed. This information resource may become a member of the bound data of another atomic action B. However, a CCR service-user of atomic action B shall not offer or order commitment unless atomic action A has committed.

Concurrency controls remain in place until the final C-COMMIT, C-ROLLBACK, or C-RECOVER exchange occurs.

C.4.2 Concurrency example – Locking

One way of achieving concurrency control is to utilize the mechanism of locking. An information resource is treated by the operating system as a serially owned resource. An application is granted ownership of the information resource by the operating system. All other applications are not granted access to the resource. When the locking application completes its use of the resource, it releases the resource (i.e. it unlocks the resource). The resource is now available for use by other applications.

For a CCR service-user that uses locking for concurrency control, all bound data are locked from its first use until its local completion or commitment procedures. No other entity, such as another CCR service-user may access the locked bound data. This approach causes a strong serialization of atomic actions. Other implementation techniques are possible.

Frequently, locks are automatically released by an operating system following the failure of the owning application or of the system. However, if such a failure occurs after commitment has been offered, concurrency controls must remain in place until the C-RECOVER exchange occurs. This type of locking, unmodified, is therefore not suitable for CCR.

Concurrency control can be achieved if the locking is recorded as atomic action data. This information is then used to reinstate the locks during recovery of the application or the system after failure.

A lock used for concurrency control is "owned" by the (single) atomic action in process. It serves only to prevent interference by other entities.

NOTE – The provision for more elaborate concurrency controls may be the subject of future CCR standardization. This could include handling atomic sub-actions within atomic actions.

Within an AEI, the processing of one branch may need to access bound data previously modified by another uncommitted branch of the same atomic action. This is explicitly allowed by concurrency controls. The use of locks should not prevent this.

As an example, consider a remote file operation. The file representing the bound data has been modified and closed. The CCR service-user (the subordinate) has not offered commitment.

Now, consider a read-only file operation for the same file. If the atomic action identifier shows that this operation is part of the same atomic action, then reading the data should be permitted.

If the second file operation is not part of the same atomic action, file access should be refused. The refusal could be in the form of a C-ROLLBACK request primitive with an appropriate "retry-later" diagnostic in the User Data parameter. The refusal could also be expressed by diagnostic semantics on the branch. Alternately, refusal could simply result in a wait for the release of the lock.

Suppose, however, that the second file operation wished to modify the file before the first operation had closed it. It would be refused (as above) if not part of the original atomic action. If, however, the second operation was part of the original atomic action, then an illegal action is being requested. The action taken depends on the granularity of concurrency controls associated with the file. For example, C-ROLLBACK request primitive might be issued with a no-retry diagnostic indicating the error.

C.5 Recovery

Recovery procedures are an integral and essential part of CCR. Without them, concurrency controls (e.g. locks) might be imposed and never released, and commitment could fail to provide atomicity.

CCR provides services for the recovery of a single branch of an atomic action. The coordination of the recovery of multiple branches of an atomic action is the responsibility of the referencing specification and its implementation.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.5.1 Atomic action data

The backbone of CCR recovery is atomic action data maintained by the CCR service-user. Atomic action data consist of state and control information about the CCR service-user and its branches.

CCR does not define the specific information that must be saved as atomic action data. This is the responsibility of the referencing specification.

However, CCR requires that the information maintained by a CCR service-user is sufficient to support the recovery of a branch interrupted by an application or communication failure. Atomic action data must also be able to identify the initial and final states of bound data.

CCR defines when atomic action data must persist. After an application or communication failure, a CCR service-user must be able to find the atomic action data for recovery. The existence of atomic action data for a branch, causes the CCR service-user to attempt its recovery (using the C-RECOVER service).

The act of making atomic action data persistent is called recording the atomic action data. (Recording atomic action data is sometimes called logging, but no specific implementation is implied.) The act of making atomic action data not persistent is called removing or forgetting the atomic action data (see A.5). Forgetting atomic action data does not require that it is physically deleted. It only requires that forgotten atomic action data do not cause a recovery attempt.

Table C.1 is an example of the type of information that could be included as atomic action data. This information includes the atomic action identifier, and state information about whether the CCR service-user has offered commitment or ordered (received an order of) commitment.

Table C.1 also contains information about each of the CCR service-user's branches. This includes the branch identifier and the ACSE A-ASSOCIATE parameter values needed to re-establish an association to the peer CCR service-user if recovery is required.

Finally, Table C.1 includes information about bound data including the definition of initial and final states.

As in Table C.1, atomic action data may include information about the CCR service-user itself. However, in a particular implementation, this may be implicit rather than explicit.

Table C.1 – Example of atomic action data

Atomic action identifier: – Owner's Name; – Suffix.
CCR service-user information: – AE title; – API identifier; – AEI identifier; – Role [initiator / intermediate / leaf]; – State [offered / committed].
Branch information {repeated for each branch}: – Suffix; – this CCR service-user's role [Superior / subordinate]; – information required to establish association with peer: • AE title; • API identifier; • AEI identifier; • Presentation address; • Application context name.
Bound data information: – Resource identification; – Initial state information; – Final state information; – Concurrency information; – Access information.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.5.2 Presumed rollback

CCR makes use of the presumed rollback (or presumed abort) paradigm. It defines when a CCR service-user acquires recovery responsibility for a branch. This paradigm also defines the significance of not having atomic action data for an identified branch that a peer requests to recover.

C.5.2.1 READY record

With presumed rollback, a subordinate acquires recovery responsibility for that branch when it decides to offer commitment. It makes the atomic action data persistent and records READY for this branch (see A.3.3 and A.4.1). If the atomic action data are represented as shown in Table C.1, the state is "offered".

If the CCR service-user is a leaf, the Table C.1 atomic action data contain branch information for one branch, with role of "subordinate".

If the CCR service-user is an intermediate, the atomic action data will include the READY record for the branch to the superior[4]. The atomic action data also include branch information for the branches to its subordinates. If the atomic action data are as represented in Table C.1:

- the state is "offered";
- for one branch, the role is "subordinate"; and
- for one or more other branches, the role is "superior".

An intermediate is not considered to acquire recovery responsibility for the branches to its subordinates when it records READY for the branch to the superior. However, atomic action data can be found when it receives a C-RECOVER(ready) from a subordinate. If the intermediate cannot establish an association to its superior, it issues a C-RECOVER(retry-later) response primitive back to the subordinate.

C.5.2.2 COMMIT record

A CCR service-user that acts as the superior of a branch, acquires recovery responsibility for the branch when it decides to order commitment.

This applies most clearly to the commit-coordinator. If, after it has received offers of commitment from all its subordinates (see A.3.4), the commit-coordinator decides to order commitment (becoming commit-decider), it records the atomic action data, creating a COMMIT record. If the atomic action data are as represented in Table C.1:

- the state is "committed"; and
- the role for each branch is "superior".

Having created the COMMIT record, the CCR service-user orders commitment on each branch.

An intermediate has the option, after receiving an order of commitment, of rewriting the atomic action data to be a COMMIT record. This contains the same information as a commit-decider's COMMIT record. It does not, in the terms of Table C.1, contain branch information for the branch to the superior. Changing a log from an intermediate's READY record to a COMMIT record can help to optimize recovery actions if an application or communication failure occurs.

C.5.2.3 Forgetting the atomic action data

After a subordinate offers commitment, it forgets the atomic action data when the result (commit or rollback) of the atomic action is known. The result is rollback, if the subordinate receives a C-ROLLBACK indication or C-RECOVER(unknown) confirm primitive. The result is commit, if it receives a C-COMMIT or C-RECOVER(commit) indication primitive.

For commitment, the subordinate must forget the atomic action data before sending the C-COMMIT or C-RECOVER(done) response primitive. If it did not, the presence of the atomic action data would permit a future C-RECOVER(ready) request primitive. This would result in the receipt of a C-RECOVER(unknown) confirm primitive, implying that the atomic action had rolled back.

After a superior orders commitment, it forgets the atomic action data when CCR exchanges ensure that the subordinate has received the order and has acted on it. This occurs when the superior receives a C-COMMIT or C-RECOVER(done) confirm primitive.

---

[4] The branch to the superior is the only one on which it is the subordinate.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.5.2.4 Recovery

CCR considers two types of failure:

- application failure; and
- communication failure.

Application failure may be modelled as issuing or receiving an A-ABORT request or indication primitive, respectively. Communication failure may be modelled as receiving an A-P-ABORT indication primitive.

During local recovery procedures, commitment controls are released for any bound data that do not have a corresponding READY or COMMIT record.

After a failure, if a C-RECOVER(ready) indication primitive is received and atomic action data cannot be found for the referenced branch, the CCR service-user assumes that the branch has been rolled back. If a C-RECOVER(commit) indication primitive is received and atomic action data cannot be found for the referenced branch, the CCR service-user assumes that a commit order had been previously received and acted on.

The correct operation of CCR is achieved only by ensuring that READY record and COMMIT record atomic action data persist across failures.

A correct implementation of CCR ensures that information written as a READY or COMMIT record has been secured before the next service primitive is issued. For some operating systems, this will involve writing data to disc and ensuring that all relevant disc buffers have been flushed. This requires care in implementation.

C.5.2.5 Discussion

The presumed rollback mechanism minimizes the logging of atomic action data. However, this mechanism cannot always handle recovery without causing a rollback of a larger part of the atomic action than that directly involved in the failure. Minimizing the area affected by a failure is neither required not defined by CCR. The cost of such additional persistent data updates must be balanced against the advantages of localizing the need for recovery, and the probability of recovery being needed.

NOTE – CCR does not provide global checkpoints within a branch. This is a possible area of CCR standardization. The use of checkpoints could reduce the amount of repeated work if a failure occurs.

The local recovery procedures of one branch may require a CCR service-user to roll back other branches of the atomic action. This is determined by the referencing specification. The decision to roll back other branches may be influenced by whether or not intermediate results have been secured by the intermediate CCR service-users. This allows the use of pair-wise check pointing mechanisms.

C.5.3 AP and AE invocation identifiers

Subclause A.4 defines the atomic action data manipulation rules for a CCR service-user. If these rules are observed, the following will be true. After receiving a C-RECOVER indication primitive, the CCR service-user will either find the atomic action data for the referenced branch, or it will correctly determine that none exists. This is a key requirement of the CCR recovery mechanism.

Accessing (i.e. finding) atomic action data has implications on how AP and AE invocation identifiers[5)] are used. Two possibilities exist:

a) Atomic action data are accessible by any invocation (AEI) of a particular AE.

b) Atomic action data are accessible only by a particular AEI. That is, atomic action data are not accessible to an invocation of the same AE that has a different invocation-identifier.

CCR does not specify the use of either method. This is the responsibility of the referencing specification. However, interworking difficulties may result between implementations that use different methods. For this reason, each method is briefly discussed below as it relates to the use of invocation identifiers.

C.5.3.1 Accessibility by any AEI

The first method does not use AP or AE invocation identifiers. Atomic action data are recorded and accessed independent of the AP and AE invocations originally involved with the branch.

5) Values for the calling, called and responding AP and AE invocation identifier values may be exchanged on the A-ASSOCIATE service (see ITU-T Recommendation X.217 | ISO/IEC 8649).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

When establishing an association, either CCR service-user may inform its peer of its invocation identifiers. The association-requestor may do this by using the calling invocation identifiers on the A-ASSOCIATE request primitive. Likewise, the association-responder may use the responding parameters. The CCR service-users may exchange these values for reasons other than accessing atomic action data.

However, care should be taken if the association-initiator uses the called invocation identifier parameters. A called invocation identifier may no longer be in use by the association-responder. This may be especially true when an association is being established after a failure.

C.5.3.2 Accessibility by a specific AEI

The second method uses AP and AE invocation identifiers. Atomic action data are recorded and accessed specifically for the AP and AE invocations originally involved with the branch.

When establishing an association, each CCR service-user informs its peer of its invocation identifiers. The association-requestor does this by using the calling invocation identifiers on the A-ASSOCIATE request primitive. The association-responder uses the responding parameters. These invocation identifiers are included in the atomic action data for the branch.

If an association is to be used for recovery, the called invocation identifier parameters is also present on the A-ASSOCIATE request primitive. The values of the calling and called parameters on the A-ASSOCIATE request primitive are the same as those previously used on the association that carried the C-BEGIN on the interrupted branch.

An association is established with the invocation of the interrupted branch. If the association cannot be established to the called AEI, the CCR service-user of an interrupted branch cannot get appropriate C-RECOVER(done) or C-RECOVER(unknown) confirm primitives. It will continue recovery attempts (or, at least, association attempts) indefinitely.

Invocation identifier values are assigned to an API and AEI for an indefinite time. This is necessary because one CCR service-user may view a branch as completed while the other may view it as interrupted. For example, consider a failure that occurred before a C-READY indication primitive was received by the superior. The superior views the branch as completed (presumed rollback). However, for the subordinate, it is interrupted. The subordinate will attempt to send a C-RECOVER(ready) request primitive on another association to the superiors API and AEI.

In practice, an implementation may choose to use only a few invocation identifier values for this method.

C.6 Time relations and sequence of service primitives

CCR supports the two-phase commitment process (see 6.1.7). For the CCR service-users of an atomic action, transitions between their phases occur at different times.

CCR does not specify when phase I starts. Phase I can be regarded as including the period that follows C-BEGIN, when flows of application-semantics manipulate the bound data. Alternatively, phase I can be regarded as beginning only when C-PREPARE or an equivalent semantic is issued.

Figure C.3 shows the time relations for a two branch atomic action: the initiator, A; the intermediate, B; and the leaf, C. For each of these CCR service-users, Figure C.3 shows the start and end of phases, the times when atomic action data are recorded, and when application-semantics can be sent[6)] for an atomic action branch. On the branch between A and B, C-BEGIN is confirmed and C-PREPARE is used. On the branch between B and C, C-BEGIN is non-confirmed and the prepare semantic is implicit (see C.9).

A careful examination of Figure C.3 shows two important features of CCR. First, no matter when failures occur, a subordinate is never left with an incomplete branch that the superior believes is complete. The branch is either rolled back or one or both CCR service-users attempt recovery using the C-RECOVER service. Secondly, the reliability of CCR semantic-exchanges is achieved without requiring CCR service-users to retain knowledge (in persistent data) of the atomic action identifier beyond the end of the atomic action. However, to insure uniqueness, an atomic action identifier cannot be reused.

6) For Figures C.3 and C.7, the "DATA" service is used to express sending and receiving application-semantics that manipulate the bound data of the atomic action. Also, for Figure C.3, the "DATA (outside)" service expresses sending and receiving semantics on the association but before the branch begins.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figures C.4 to C.6 present additional atomic action scenarios. The atomic action consists of four CCR service-users:

– the initiator, A;

– an intermediate, B; and

– two leaves, C and D.

Each figure shows the sequence of CCR service primitives and the recording of atomic action data.

Figure C.4 shows the scenario of an atomic action that proceeds to commitment. Here, the optional C-PREPARE service is used. Notice that the intermediate, B, does not issue a C-PREPARE request primitives to its subordinates until it has received a C-PREPARE indication primitive from its superior (the initiator, A).

Figure C.5 shows a scenario where leaf D cannot offer commitment. It rolls back the branch to the superior. The intermediate then decides to roll back the branches to its other leaf (C) and to its superior (A).

Figure C.6 shows a scenario that is similar to that in Figure C.4 with two exceptions. The optional C-PREPARE is not used and the use of the C-BEGIN service is non-confirmed.

C.7 Comments on implementation complexity

The nature of using CCR depends on the level of the activity and the structure of the supporting operating system.

As an example, consider the use of CCR for remote file operations between two AEIs. For this example, the superior requests an operation on a remote file controlled by the subordinate. The atomic action is one file operation. The bound data are the file at the remote (subordinate's) end.

The following list describes several remote file operations that require increasing levels of implementation care:

a) *Read a file* – Here, the final state of the file is also its initial state. Here implementation is easy. The subordinate can issue a C-READY request primitive when the last record is read and sent to the superior. The superior can then issue a C-ROLLBACK request primitive or go through the commitment procedure. Some form of concurrency control is needed to ensure the isolation property.

b) *Creating a new file* – The initial state is the absence of the new file in the subordinate's system. The final state is the presence of the file with all its records. The superior issues a C-PREPARE request primitive (or equivalent) when it has sent all data for the new file. If the subordinate is ordered to commit, the new file is made accessible. If the subordinate is ordered to roll back, the new file is deleted and never made available.

c) *Overwriting an existing file* – When the subordinate offers commitment (i.e. it issues a C-READY request primitive), it retains the ability to produce either the old file or the overwritten file. If the subordinate is ordered to commit, the file has the overwritten content. If it is ordered to roll back, the file has its initial content.

d) *Appending to an existing file* – The final state is the old file with the appended records. If the subordinate is ordered to commit, it makes the file with the appended records available. If ordered to roll back, the appended records are removed before the file is made available. On some present-day systems, copying the old file may be necessary to handle recovery correctly from application failure during commitment, and the possibility of rollback.

e) *Writing a single record and closing a file* – Here, the initial state is an open file without the record. The final state is a closed file with the record.

C.8 Using the User Data parameter on CCR services

Each CCR service (e.g. C-BEGIN) has an optional User Data parameter on its primitives (see clause 7). The referencing specification determines the use (if any) of these User Data parameters.

This subclause discusses four different ways to use the User Data parameters:

a) specifying the level of commitment;

b) expressing diagnostics;

c) carrying application-semantics; and

d) reporting heuristic damage.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

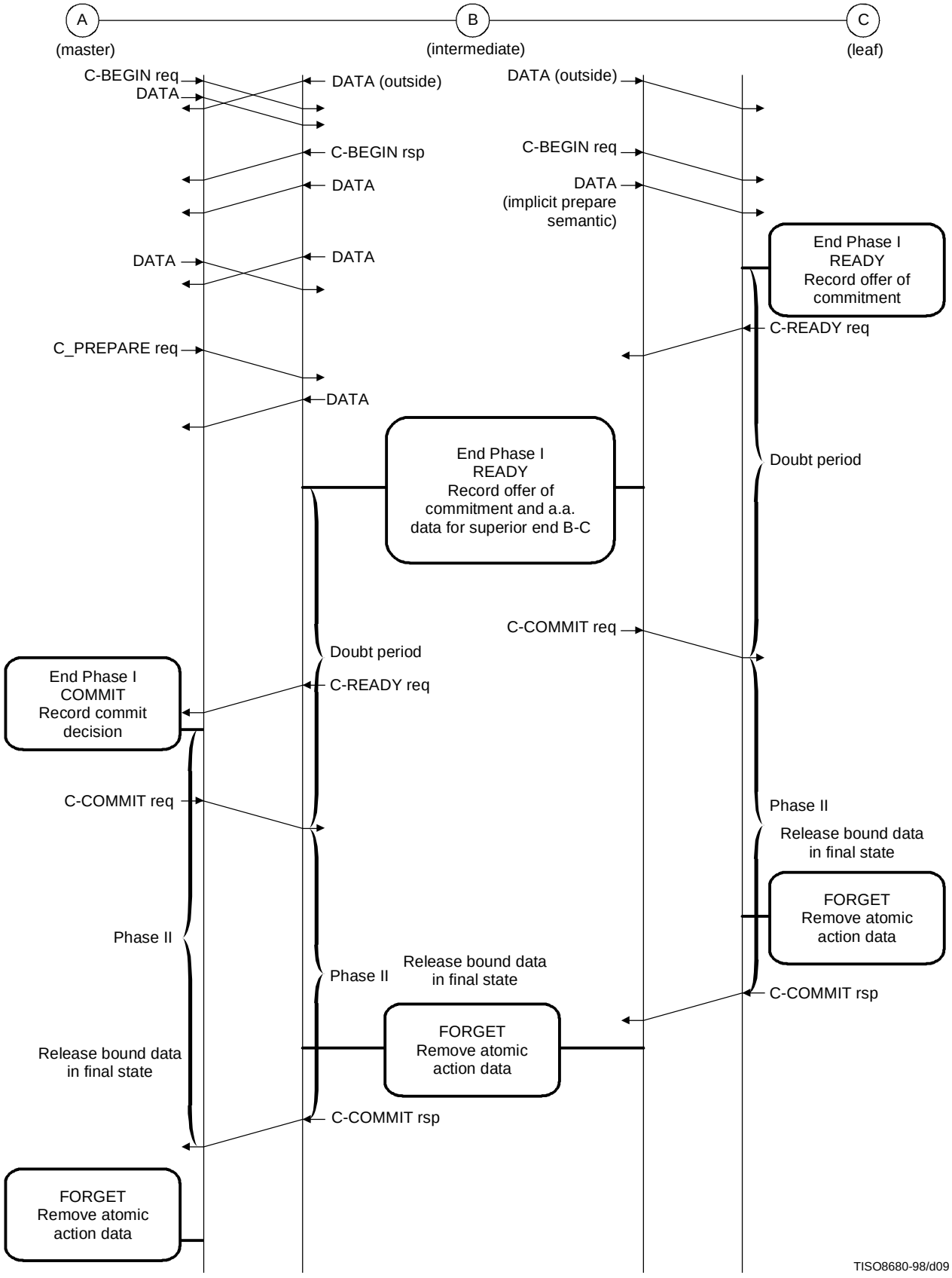

Figure C.3 – Time relations for two-branch atomic action tree

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

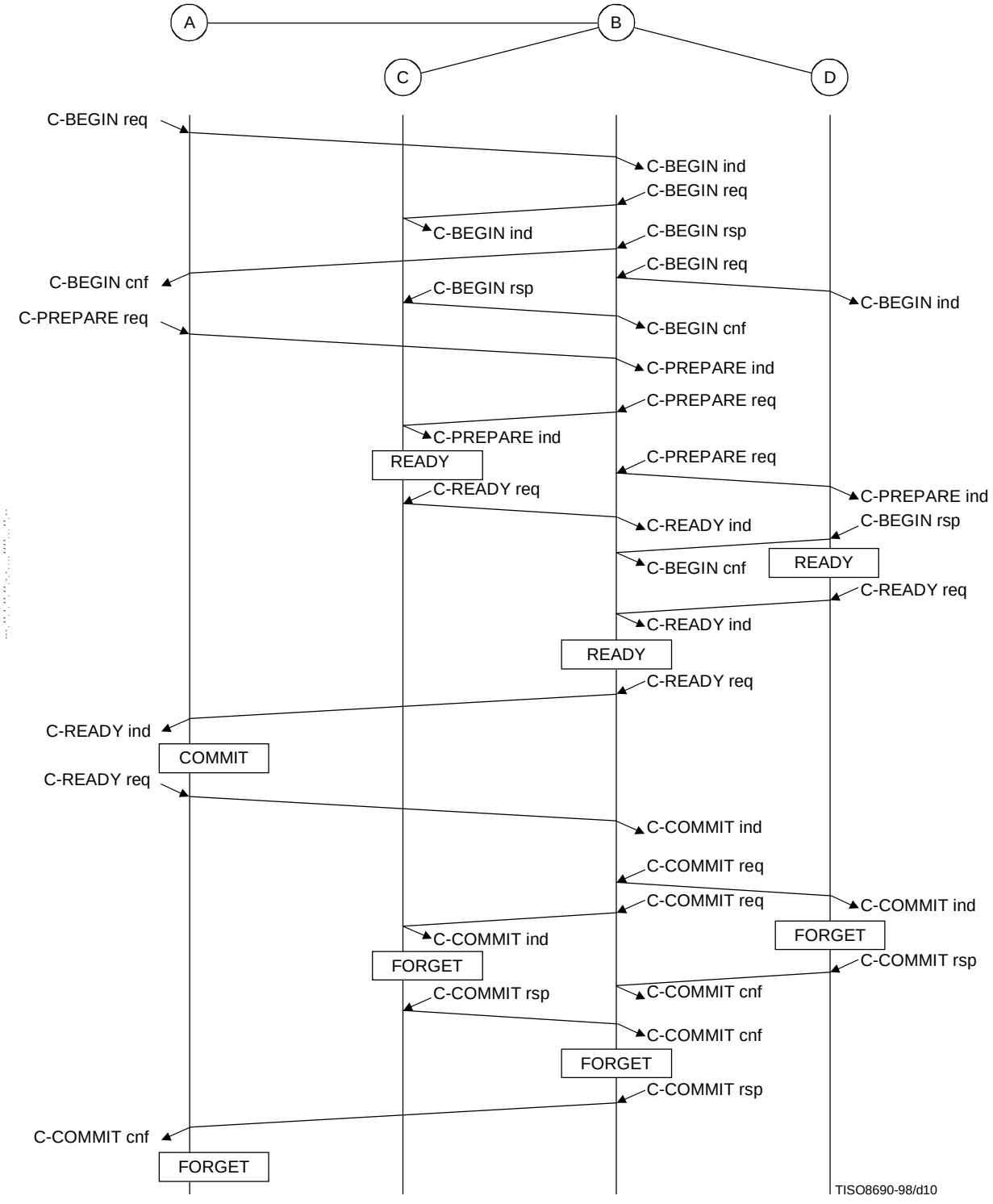

Figure C.4 – Sequence of primitives – Atomic action is committed – Explicit C-PREPARE

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

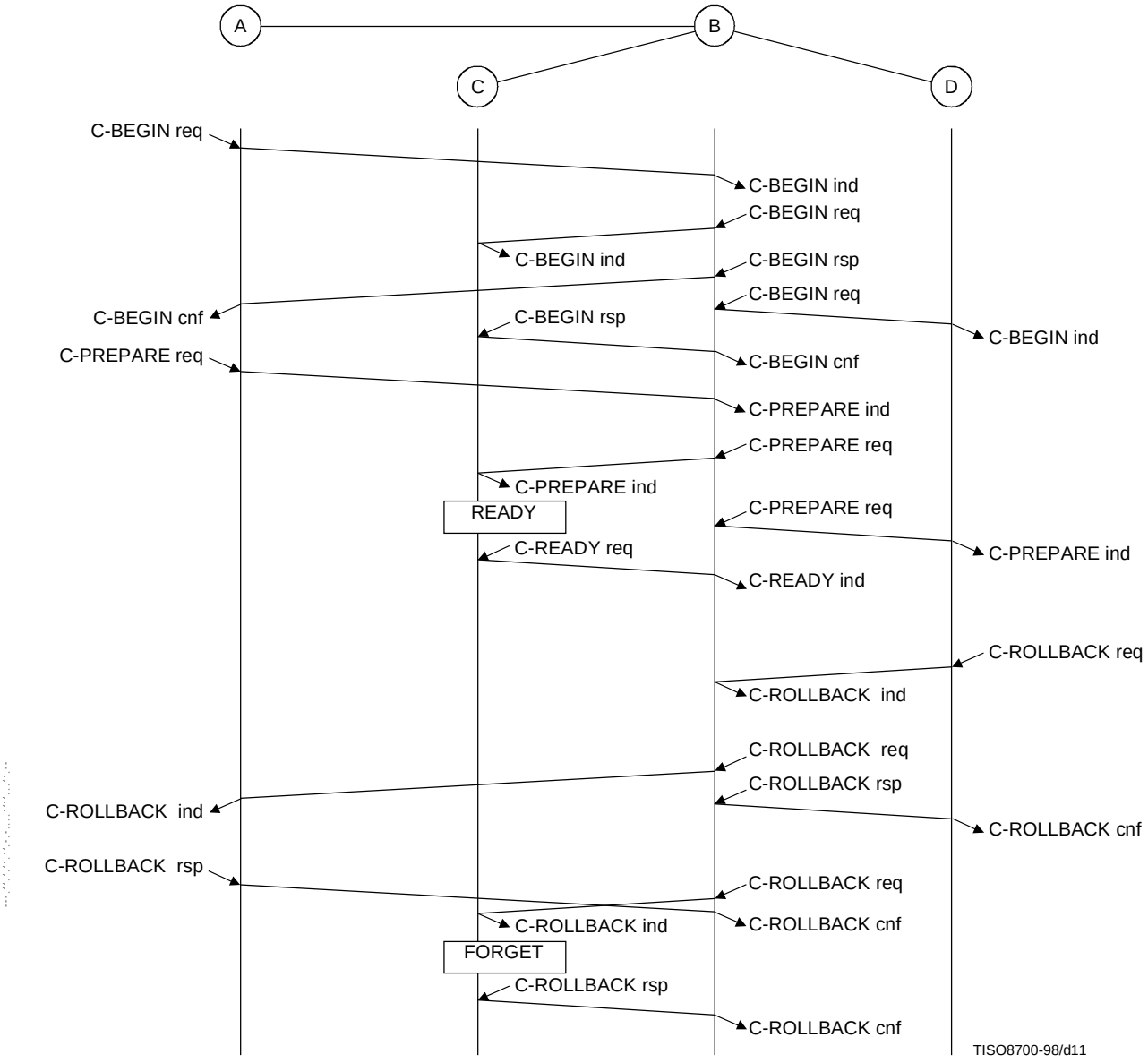

Figure C.5 – Sequence of primitives – Atomic action is rolled back

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

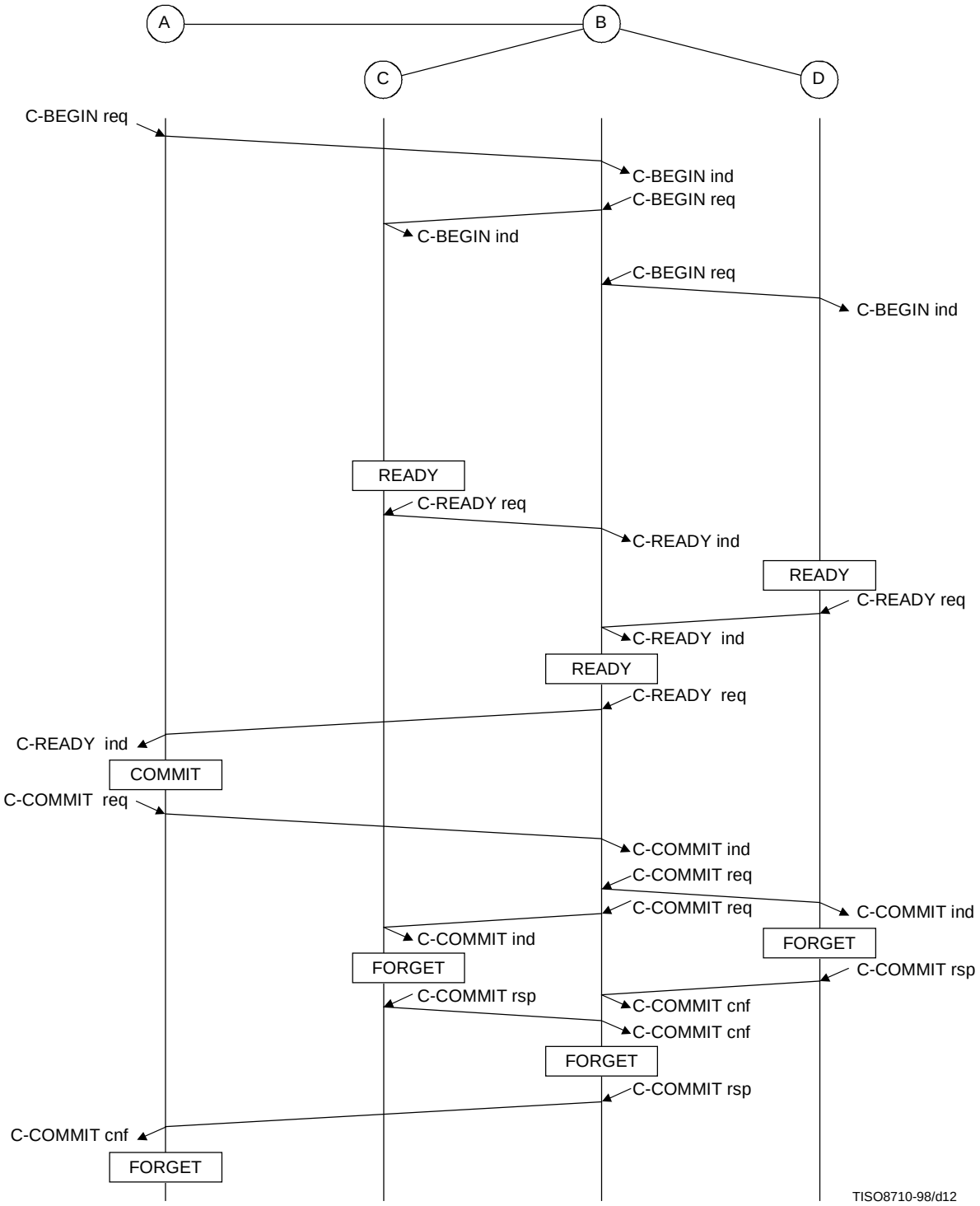

Figure C.6 – Sequence of primitives – Atomic action is committed – Implicit prepare – C-BEGIN is not confirmed

Semantics carried in a User Data parameter are, in effect, another ASE's APDU. Rather than using the CCR User Data parameter, a referencing specification may concatenate an ASE's APDU with the CCR APDU. The same options and restriction apply as for the use of the User Data parameters. For example, a concatenated APDU that occurs after the PREPARE or READY CCR APDUs should not manipulate bound data.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.8.1 Level of commitment

Commitment is the process of completing a branch with the release of bound data in the final state. For some applications the meaning of release and final state can be assigned varying levels of rigidity. The selection of the particular level of commitment for a branch can be expressed by semantics sent and received on a User Data parameter of a CCR service primitive.

For example, if a superior sends a subordinate material for printing, the superior may require the completion of the printing before commitment. Alternately, it may simply require the securing of the material, with the printing done later.

JTM, for example, allows commitment to a lesser action than requested by the superior. However, in doing this, the subordinate implicitly agrees to report any failure and to complete the full action as a future atomic action.

Another level of commitment example involves the archiving of a database by the subordinate. Here, a low level of commitment is assigned to changes to the main (disc) version. A higher level of commitment is assigned to changes that have also been archived.

In general, a superior expresses its required level of commitment by semantics sent in the User Data parameter of the C-BEGIN request primitive. The subordinate does at least what is required, and possibly more. The subordinate then reports its achievement in the User Data parameter of the C-READY request primitive.

The concept of commitment level may not be needed for many applications. When used, its semantics depends on the referencing specification.

C.8.2 CCR diagnostics

A referencing specification may need to express diagnostic semantics about the progress or completion of a branch. For example, diagnostics may be conveyed in one of the following CCR User Data parameters:

a) C-COMMIT request or response primitive: a warning about a potential problem;

b) C-RECOVER(retry-later) request primitive: a retry-later diagnostic that indicates when a future attempt is appropriate; and

c) C-ROLLBACK request primitive from a subordinate: a no-retry diagnostic that indicates a future attempt should not be attempted.

Because an atomic action tree may involve many CCR service-users, the referencing specification should consider defining each diagnostic message with three pieces of information:

a) the identity of the message source;

b) the machine-readable portion of the message; and

c) a human-readable portion.

C.8.3 Application-semantics

The User Data parameter on the C-BEGIN, C-PREPARE, C-READY, and C-RECOVER(ready) service primitives may be used to carry application-semantics (i.e. data that manipulate the bound data). For example, the superior may send its final application-semantics as User Data on the C-PREPARE request primitive. Likewise, the subordinate may send its last application-semantics as User Data on the C-READY request primitive.

The User Data parameter on the C-COMMIT, C-RECOVER(commit), and C-ROLLBACK services can only be used for non-application-semantics (i.e. semantics that do not directly affect the bound data).

C.8.4 Heuristic warning

The User Data parameter on a C-ROLLBACK and C-COMMIT response primitive may be used to carry reports of heuristic damage. However, such reports may be lost if failure occurs. The confirm primitive may not be received by the other CCR service-user. No further semantic flow will occur for the branch because atomic action data were forgotten before the response primitive was issued.

The User Data parameter on these primitives can also be used to warn that a failure might have caused the loss of heuristic damage reports farther down the atomic action tree.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C.9 Optional use of C-PREPARE

A superior may issue a C-PREPARE request primitive to inform the subordinate that it will not send the subordinate any further application-semantics. However, the superior may send data that do not manipulate the subordinate's bound data. Only the CCR service-user can distinguish between application and non-application-semantics.

Invoking the C-PREPARE service also tells the subordinate to complete its processing for this branch and to offer commitment. The superior can use the User Data parameter to send its last application-semantics to the subordinate.

C.9.1 Implicit prepare semantic

C-PREPARE is an optional service. If C-PREPARE is not used by a referencing specification, the last application-semantics sent from a superior to the subordinate must carry equivalent "implicit prepare" semantic. That is, the subordinate must know when it has received the last application-semantics from the superior. The subordinate may then offer commitment.

C.9.2 Implicit conditional prepare and continue

If a referencing specification does not use C-PREPARE, it can take an alternative approach to express prepare semantic.

The last application-semantics sent from the superior may include "conditional prepare" semantic. The conditional "prepare" received by the subordinate allows it to invoke C-READY. Alternately, the subordinate can send "continue" semantic to the superior. This indicates that it will not send a C-READY now and the branch continues.

After sending the "conditional prepare" semantic, the superior may not send application-semantics until it receives a C-READY indication primitive or the alternative "continue" semantic.

C.9.3 C-READY and application-semantics collision

One of the tasks of a referencing specification is to ensure that a C-READY request primitive from the subordinate does not collide with application-semantics from the superior. This is an application error. The collision violates the atomic action property of consistency.

This collision is not a CCR sequencing (i.e. protocol) error. The CCR ASE cannot detect this situation. It is not "aware of" application-semantics sent and received on the branch. That is, application-semantics (DATA) requests and indication primitives are not CCR events (see Tables 9 to 14).

NOTE 1 – CCR allows the collision of C-PREPARE and C-READY services. If the C-PREPARE carries application-semantics, a C-READY and application-semantics collision occurs. The CCR ASE cannot detect this situation either.

Figure C.7 shows a collision between application-semantics and the C-READY service. The superior has issued two DATA request primitives that send application-semantics to the subordinate. The subordinate receives the first DATA indication primitive. It assumes that this is the final application-semantics for the branch. It then issues a C-READY request primitive to the superior. Then, it receives the second DATA indication primitive containing application-semantics that will manipulate its bound data. The two services collide.

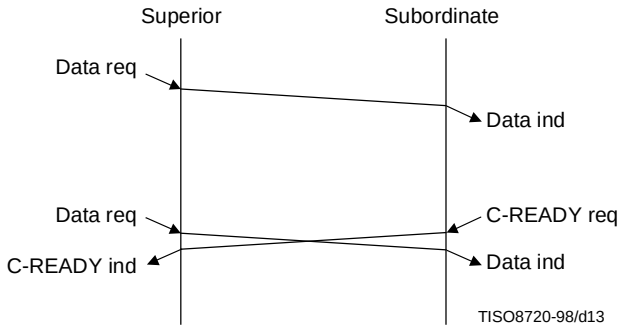

Figure C.7 – Collision between application data and C-READY

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The subordinate is aware of the collision. It received application-semantics after it sent a C-READY request primitive. This is an application error. However, the subordinate cannot roll back the branch.

On the other hand, the superior cannot detect the collision by just observing the sequence of CCR primitives at its end of the branch. It does not know if the subordinate issued the C-READY request primitive before or after received the second DATA indication primitive.

NOTE 2 – This situation may be the subject of future CCR standardization.

An implementation may consider several ways to avoid this situation:

a) Use the explicit C-PREPARE service or have a well defined implicit prepare semantic.

b) Send no data (application or non-application-semantics) to the subordinate after the explicit or implicit C-PREPARE request primitive.

c) Do not issue a C-READY request primitive until the subordinate receives either the explicit or implicit C-PREPARE indication primitive.

d) On the C-PREPARE request primitive, include a semantic "token" in the User Data parameter. The subordinate must return this token in the User Data parameter of the C-READY request primitive. If the superior does not receive the token, it rolls back the branch.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

© ISO/IEC

ICS 35.100.70

Price based on 60 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale