JTC1

INTERNATIONAL STANDARD

ISO/IEC 9804

Second edition
1994-12-15

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 de service d'engagement, concurrence et reprise

Reference number
ISO/IEC 9804:1994(E)

CONTENTS

- ITU-T Recommendation X.215 (1993) | ISO/IEC 8326:1994, *Information technology – Open Systems Interconnection – Connection oriented session service definition.*
- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Connection oriented presentation service definition.*
- ITU-T Recommendation X.217 (1994) | ISO/IEC 8649:1994, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element.*
- ITU-T Recommendation X.852 (1993) | ISO/IEC 9805:1994, *Information technology – Open Systems Interconnection – Protocol Specification for the Commitment, Concurrency and Recovery Service Element.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- ITU-T Recommendation X.210 (1993), *Open system interconnection layer service definition conventions.*

  ISO/IEC 10731: ...[1], *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of ISO services.*

3 Definitions

3.1 Reference model definitions

3.1.1 Basic Reference Model definitions

This Service Definition 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-entity;
b) Application Layer;
c) application-process;
d) application-service-element;
e) presentation-connection;
f) presentation-service;
g) session-connection;
h) session-service.

3.1.2 Naming and addressing definitions

This Service Definition makes use of the following terms defined in CCITT Rec. X.650 | ISO/IEC 7498-3:

application-entity title.[2]

3.2 Service conventions definitions

This Service Definition makes use of the following terms defined in ITU-T Rec. X.210 | ISO/TR 8509:

a) service-provider;
b) service-user;
c) confirmed service;
d) non-confirmed service;
e) provider-initiated service;
f) primitive;

---

1) Presently at the stage of draft.

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

g) request (primitive);

h) indication (primitive);

i) response (primitive); and

j) confirm (primitive).

3.3 Presentation service definitions

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

a) application-context;

b) application-entity invocation;

c) multiple association control function;

d) single association control function;

e) single association object.

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 superior 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 identifier: A value assigned by the master that uniquely identifies an atomic action within the OSI environment.

3.6.8 atomic action tree: A hierarchical relationship between CCR service-users involved in the operations of an atomic action.

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

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

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

3.6.12 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 tree.

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

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

3.6.15 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 master.

3.6.16 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.

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

3.6.18 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.

3.6.19 cooperating main service: A referencing specification that incorporates the CCR semantics within its own service primitives and carries CCR transfer syntax within its own protocol-data-units.

3.6.20 distributed application: An information processing endeavor 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.

3.6.21 doubt period: For a CCR service-user (that is not the master), the period during an atomic action that begins when it decides to offer commitment to its superior and ends when it receives either the order to commit or to rollback. The master CCR service-user does not have a doubt period.

3.6.22 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.

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

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

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

3.6.26 intermediate CCR service-user; intermediate: A CCR service-user that has the role of both subordinate and superior. It is a subordinate of the master CCR service-user or another intermediate CCR service-user. It is the superior of one or more other intermediate and/or leaf CCR service-users.

3.6.27 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.

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

3.6.29 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.

3.6.30 leaf CCR service-user; leaf: A CCR service-user that only has the role of subordinate. It is the subordinate of the master CCR service-user or an intermediate CCR service-user. It has no subordinates of its own.

3.6.31 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.

3.6.32 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.

3.6.33 master CCR service-user; master: A CCR service-user that has the role of superior. As the creator of the atomic action tree, it has no superior, but it is the superior of one or more intermediate and/or leaf CCR service-users.

3.6.34 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 master.

3.6.35 offer of commitment of an atomic action branch; offer of commitment: A statement from the subordinate to the superior that the subordinate is ready for either commitment or rollback.

3.6.36 order of commitment of an atomic action branch; order of commitment: A statement by the superior to the subordinate to initiate commitment of the atomic action branch.

3.6.37 phase I: For a CCR service-user that is not the master, the period during an atomic action that ends when it decides to offer commitment to its superior. For the master CCR service-user, phase I ends when it decides to commit the atomic action. This Recommendation | International Standard does not specify when phase I starts.

3.6.38 phase II: For a CCR service-user that is not the master, the period during an atomic action that begins when it is ordered to commit by its superior. For the master 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.

3.6.39 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 subordinate may presume rollback under the following condition. It has recorded atomic action data for the branch but, during recovery, it discovers that the superior does not.

3.6.40 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.

3.6.41 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.

3.6.42 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.

3.6.43 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.

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

3.6.45 subordinate of an atomic action branch; subordinate: The CCR service-user that receives the request to begin the branch, offers commitment and receives the order to commit.

3.6.46 superior of an atomic action branch; superior: The CCR service-user that requests the beginning of the branch, receives the offer of commitment and orders commitment.

4 Abbreviations

This Recommendation | International Standard uses the following abbreviations.

ACSE	Association control service element
AE	Application-entity
AEI	Application-entity invocation
Amd	Amendment to an ISO/IEC International Standard
ASE	Application-service-element

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

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 tree

The CCR service-users that participate in an atomic action form a relationship that has a tree structure. For this Recommendation | International Standard, such a relationship is modelled as an atomic action tree, as shown in Figure 1. An atomic action tree consists of CCR service-users and atomic action branches.

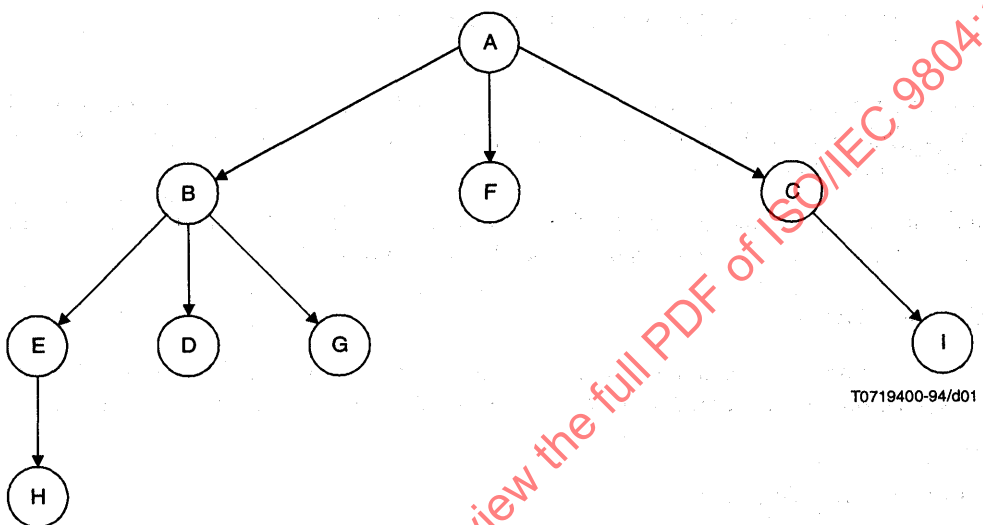

Figure 1 – Atomic action tree

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

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

A **branch** of the atomic action is the relationship between two logically adjacent CCR service-users.

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

An atomic action tree starts when a CCR service-user begins the first branch. This CCR service-user assigns this atomic action an atomic action identifier whose value uniquely identifies it within the OSIE. 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.

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

Beginning from any CCR service-user, an atomic action tree can be ordered hierarchically. Such an ordering that begins with the CCR service-user that started the atomic action defines the atomic action begin-tree. Figure 1 shows an atomic action tree in this ordering started by CCR service-user A. The arrow on each branch shows the direction in which it was started.

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 tree ends with the completion of all the individual branches.

6.1.2 Atomic action branch

An atomic action branch is a relationship between two logically adjacent CCR service-users. This relationship performs a portion of the work of an atomic action. The branch is requested by one of the CCR service-users and the other receives the request. Within the atomic action tree hierarchy, the CCR service-user that receives the request is one level lower than the CCR service-user that begins the branch.

The CCR service-user that begins a branch 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 tree 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 offers of commitment are collected. During phase II commitment is ordered and confirmed.

Within the atomic action, each CCR service-user may offer commitment on one branch or it may make no offer of commitment. The atomic action may therefore be represented as a hierarchical tree ordered on the basis of offers of commitment – this is the atomic action commit-tree.

6.1.6 Roles in an atomic action

Offers of commitment on a branch are only made from the CCR service-user that received the C-BEGIN indication primitive for the branch.

The following roles for a branch can therefore be distinguished:

a) *Superior* (of the branch) – The CCR service-user that requests the beginning of the branch, receives the offer of commitment and orders commitment.

b) *Subordinate* (of the branch) – The CCR service-user that receives the request to begin the branch, offers commitment and receives the order to commit.

Three types of CCR service-users exist within an atomic action:

a) *Master* – The role of superior. As the creator of the atomic action tree, it has no superior, but it is the superior of one or more intermediate and/or leaf CCR service-users.

b) *Intermediate* – Has the role of both subordinate and superior. It is a subordinate of the master CCR service-user or of another intermediate CCR service-user. It is the superior of one or more other intermediate and/or leaf CCR service-users.

c) *Leaf* – Only has the role of subordinate. It is the subordinate of the master CCR service-user or an intermediate CCR service-user. It has no subordinates of its own.

6.1.7 Two-phase commitment

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

A CCR service-user offers commitment to its superior when it has received offers of commitment from all its subordinates and has completed all operations. At this point, it is capable of placing its bound data in either the initial or final state.

The master leaves phase I and enters phase II when it decides to commit the atomic action. To do this, it has received offers of commitment from all its subordinates. It also is capable of placing its bound data in the final state. The master then orders its subordinates to commit. The master leaves phase II after receiving commitment confirmation from all its subordinates to which it has ordered commitment.

A CCR service-user, that is not the master, leaves phase I and enters the doubt period when it decides to offer commitment to its superior. It leaves the doubt period and enters phase II when it receives the order to commit from its superior. An intermediate then orders its subordinates to commit. Finally, it leaves phase II when it sends commitment confirmation to its superior.

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 master) have offered commitment. The master initiates commitment. When the master 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 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 tree whose root is an intermediate or a leaf.

A CCR service-user, that is not the master, may initiate rollback prior to offering commitment. The master may initiate rollback prior to 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 subordinates by propagating the rollback on them. If it initiated rollback, it forces the completion of the branch to its superior.

Prior to offering commitment (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 offering commitment, a CCR service-user that has not taken a heuristic decision only rolls back if it receives an order to roll back from its superior (see 6.3).

6.1.10 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.

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 subordinate acquires recovery responsibility when it decides to offer commitment. The superior acquires recovery responsibility when it decides to order commitment. Both keep recovery responsibility until the completion of the branch.

NOTE – For the master, the presumed rollback recovery mechanism does not require the recording of atomic action data until it decides to commit the atomic action. For a leaf or intermediate, the recording of atomic action data does not occur until it decides to offer commitment. 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.

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 offers commitment, 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 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 offered commitment 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 master 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 master 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.

The services provided by CCR allow two CCR service-users to communicate with each other. The use of the CCR services depends on whether a CCR service-user is acting as the superior or the subordinate.

A superior may invoke the following services:

a) C-BEGIN, to begin a branch;

b) C-PREPARE, to request that the subordinate offer commitment;

c) C-COMMIT, to order the subordinate to commit;

d) C-ROLLBACK, to order the subordinate to roll back; and

e) C-RECOVER, to perform recovery after an application or communication failure.

A subordinate may invoke the following services:

a) C-READY, to offer commitment to the superior;

b) C-ROLLBACK, to inform the superior that rollback has taken place; and

c) C-RECOVER, to perform recovery after an application or communication failure.

After a branch has been created by the C-BEGIN service, the application semantic exchange defined by the referencing specification occurs to progress the branch.

The completion of the branch is achieved either by

a) two-phase commitment, using the C-PREPARE, C-READY, C-COMMIT, and possibly C-RECOVER services; or

b) rollback, using the C-ROLLBACK, and possibly C-RECOVER services.

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

Table 1 – CCR services

Service	Type	Requestor
C-BEGIN	Optionally confirmed	Superior
C-PREPARE	Non-confirmed	Superior
C-READY	Non-confirmed	Subordinate
C-COMMIT	Confirmed	Superior
C-ROLLBACK	Confirmed	Superior or subordinate
C-RECOVER	Confirmed Optionally confirmed	Superior Subordinate

7.1 C-BEGIN service

7.1.1 Purpose and use

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

7.1.1.2 The subordinate 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.

7.1.1.3 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 superior shall own the synchronize-minor token.

7.1.1.4 The C-BEGIN service may be jointly issued with the C-COMMIT and C-ROLLBACK services (see 7.4 and 7.5, respectively).

7.1.2 C-BEGIN parameters

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

Table 2 – C-BEGIN parameters

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

7.1.2.1 Atomic action identifier

7.1.2.1.1 The Atomic Action Identifier unambiguously identifies the atomic action to which this branch belongs. Its value is assigned by the master when the first branch of the atomic action is begun. This value is subsequently used by the superior of each branch.

7.1.2.1.2 The Atomic Action Identifier consists of the Master's Name parameter together with the Suffix parameter.

7.1.2.1.3 The value of the Master's Name parameter is the master's AE title. This value unambiguously identifies the master of the atomic action.

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

7.1.2.2 Branch identifier

7.1.2.2.1 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 Superior's Name parameter together with the Suffix parameter.

7.1.2.2.2 The value of the Superior's Name is the superior's AE title. This value unambiguously identifies the superior 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).

7.1.2.2.3 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 superior.

7.1.2.2.4 The superior 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 superior's name.

7.1.2.3 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.

7.2 C-PREPARE service

7.2.1 Purpose and use

7.2.1.1 C-PREPARE is a non-confirmed service. The superior may optionally invoke C-PREPARE if it has not received an offer of commitment from the subordinate. The superior requests that the subordinate complete processing for the branch and offer commitment. The superior shall not send the subordinate any further application semantics that change the bound data of this atomic action.

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

7.2.1.3 If the subordinate is unable to offer commitment, the subordinate rolls back the branch (see 7.5).

7.2.2 C-PREPARE parameter

Table 3 lists the C-PREPARE service parameter.

Table 3 – C-PREPARE parameter

Parameter Name	Req	Ind
User Data	U	C(=)
NOTE – This parameter is set when the C-PREPARE request primitive is issued.		

7.2.2.1 User data

The superior 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.

7.3 C-READY service

7.3.1 Purpose and use

7.3.1.1 C-READY is a non-confirmed service that a subordinate invokes to offer commitment. The subordinate issues the request primitive only if it has ensured that bound data can be released in either the initial or final state.

7.3.1.2 The subordinate has recovery responsibility for this branch. That is, it shall attempt recovery after an application or communication failure (see 7.6).

7.3.1.3 The subordinate shall not send the superior any further application semantics that change the bound data of this atomic action.

7.3.2 C-READY parameter

Table 4 lists the C-READY service parameter.

Table 4 – C-READY parameter

Parameter Name	Req	Ind
User Data	U	C(=)

7.3.2.1 User Data

The subordinate 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.

7.4 C-COMMIT service

7.4.1 Purpose and use

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

7.4.1.2 The superior issues the request primitive only if the following is true:

a) the subordinate has offered commitment (see 7.3); and

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

The superior has recovery responsibility for this branch. That is, it shall attempt recovery after an application or communication failure (see 7.6).

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

7.4.1.4 For the 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.

7.4.1.5 If CCR Protocol Version 1 is being used, then the use of C-COMMIT service has the effect of establishing a major synchronization point on the underlying session-connection that supports the branch. The superior shall own the major/activity token.

7.4.1.6 If CCR Protocol Version 2 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 superior shall own the synchronize-minor token.

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

7.4.2 C-COMMIT parameter

Table 5 lists the C-COMMIT service parameter.

Table 5 – C-COMMIT parameter

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

7.4.2.1 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.

7.5 C-ROLLBACK service

7.5.1 Purpose and use

7.5.1.1 C-ROLLBACK is a confirmed service that either the superior or the subordinate may invoke to force completion of the branch. This service can cause the loss of application semantics in transit on the underlying session-connection that supports the branch.

7.5.1.2 The superior invokes this service to order the subordinate to roll back. It may issue the C-ROLLBACK request primitive ant time before invoking the C-COMMIT service.

7.5.1.3 The subordinate invokes this service to inform the superior that it refuses to proceed any further on the branch. In particular, the subordinate issues a C-ROLLBACK request primitive to inform the superior of refusal to commit. The subordinate may issue the C-ROLLBACK request primitive ant time before invoking the C-READY service. As part of the completion of the branch, the subordinate shall release all resources (e.g. bound data in the initial state).

7.5.1.4 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.

7.5.1.5 The use of the C-ROLLBACK service involves resynchronization of the underlying session-connection that supports the branch. If CCR Protocol Version 1 is being used, then this can result in a Presentation Layer indication of a change in the defined context set of the underlying presentation-connection that supports the branch.

7.5.1.6 The superior may jointly issue a C-BEGIN request primitive with a C-ROLLBACK request or response primitive.

7.5.2 C-ROLLBACK parameter

Table 6 lists the C-ROLLBACK parameter.

Table 6 – C-ROLLBACK 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 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.

7.6 C-RECOVER service

7.6.1 Purpose and use

7.6.1.1 Either a superior or a subordinate may invoke the C-RECOVER service after a failure. It is either a confirmed or optionally confirmed service depending on the requestor.

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

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

7.6.1.4 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.

7.6.1.5 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 superior and it issues a C-RECOVER request primitive as the reply to a C-RECOVER indication.

7.6.2 C-RECOVER parameters

Table 7 lists the C-RECOVER service parameters. Each parameter is discussed below.

Table 7 – 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(=)
User Data	U	C(=)	U	C(=)

7.6.2.1 Recovery state

7.6.2.1.1 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 8 lists the values for this parameter when the subordinate or the superior is the requestor.

Table 8 – Recover State parameter values

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

7.6.2.1.2 The values of this parameter have the following meanings:

a) "ready" is used by the subordinate to inform the superior that it had previously offered commitment. The following is true about the 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 7.6).

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

b) "commit" is used by the superior to inform the subordinate that it had previously ordered commitment. The following is true about the 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 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 subordinate receives a C-RECOVER indication primitive and it cannot establish an association with one or more of its subordinates.

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

7.6.2.2 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 7.1.2.1).

7.6.2.3 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 7.1.2.2).

7.6.2.4 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.

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 acting as a superior or as a subordinate. The sequences relate to 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 or C-ROLLBACK request primitive.

8.1.2 Tables 9 and 10 define the elements used in the state tables.

- Table 9 specifies the abbreviated name and a description for each state.
- Table 10 specifies the events.

8.1.3 Tables 11 to 14 specify the state tables. The tables are presented separately for convenience and clarity. They use the abbreviated names and identifiers of Tables 9 and 10.

- Table 11 shows the states and events for a CCR ASE that is used by the superior, up to the completion of the branch or the event of an application or communication failure, whichever happens first.
- Table 12 shows the states and events for a CCR ASE that is used by the subordinate, up to the completion of the branch or the event of an application or communication failure, whichever happens first.
- Table 13 shows the states and events for a CCR ASE that is used by a superior when it attempts the recovery of a branch.
- Table 14 shows the states and events for a CCR ASE that is used by a subordinate when it attempts the recovery of a branch.

Table 9 – States

State	Description
I	Idle State
A1	C-BEGIN request issued
A2	C-BEGIN request confirmed
A3	C-BEGIN request and C-PREPARE request issued
A4	C-BEGIN request confirmed and C-PREPARE request issued
A5	C-READY indication received
A6	C-COMMIT request issued
A7	C-READY indication not received, C-ROLLBACK request issued
A8	C-READY indication received, C-ROLLBACK request issued
A9	C-ROLLBACK indication received
A10	C-COMMIT request and C-BEGIN request issued
A11	C-READY indication not received, C-ROLLBACK and C-BEGIN requests issued
A12	C-ROLLBACK indication received after C-ROLLBACK and C-BEGIN requests issued
A13	C-READY indication received and C-ROLLBACK and C-BEGIN requests issued
B1	C-BEGIN indication received
B2	C-BEGIN indication received and C-BEGIN response issued
B3	C-BEGIN indication received and C-PREPARE indication received
B4	C-BEGIN indication and C-PREPARE indication received, C-BEGIN response issued
B5	C-READY request issued, C-PREPARE indication not received
B6	C-READY request issued, C-PREPARE indication received
B7	C-COMMIT indication received
B8	C-ROLLBACK indication received
B9	C-ROLLBACK request issued
B10	C-COMMIT indication and C-BEGIN indication received
B11	C-ROLLBACK and C-BEGIN indications received
X1	C-RECOVER(commit) request issued
X2	C-RECOVER(ready) indication received
Y1	C-RECOVER(commit) indication received
Y2	C-RECOVER(ready) request issued

Table 10 – Events

Abbreviated name	Source	Name and description
C-BEGIN req	Superior	C-BEGIN request primitive
C-BEGIN ind	CCR-sp	C-BEGIN indication primitive to subordinate
C-BEGIN rsp	Subordinate	C-BEGIN response primitive
C-BEGIN cnf	CCR-sp	C-BEGIN confirm primitive to superior
C-PREPARE req	Superior	C-PREPARE request primitive
C-PREPARE ind	CCR-sp	C-PREPARE indication primitive to subordinate
C-READY req	Subordinate	C-READY request primitive
C-READY ind	CCR-sp	C-READY indication primitive to superior
C-COMMIT req	Superior	C-COMMIT request primitive
C-COMMIT ind	CCR-sp	C-COMMIT indication primitive to subordinate
C-COMMIT rsp	Subordinate	C-COMMIT response primitive
C-COMMIT cnf	CCR-sp	C-COMMIT confirm primitive to superior
C-ROLLBACK req	Superior or subordinate	C-ROLLBACK request primitive
C-ROLLBACK ind	CCR-sp	C-ROLLBACK indication primitive to subordinate or superior
C-ROLLBACK rsp	Subordinate or superior	C-ROLLBACK response primitive
C-ROLLBACK cnf	CCR-sp	C-ROLLBACK confirm primitive to subordinate or superior
C-COMMIT + C-BEGIN req	Superior	C-COMMIT request primitive with a C-BEGIN request primitive
C-COMMIT + C-BEGIN ind	CCR-sp	C-COMMIT indication primitive with a C-BEGIN indication primitive to subordinate
C-ROLLBACK + C-BEGIN req	Superior	C-ROLLBACK request primitive with a C-BEGIN request primitive
C-ROLLBACK + C-BEGIN ind	CCR-sp	C-ROLLBACK indication primitive with a C-BEGIN indication primitive to subordinate or superior
C-RECOVER(commit) req	Superior	C-RECOVER(commit) request (Recovery State = "commit")
C-RECOVER(commit) ind	CCR-sp	C-RECOVER(commit) indication primitive to subordinate (Recovery State = "commit")
C-RECOVER(ready) req	Subordinate	C-RECOVER(ready) request primitive (Recovery State = "ready")
C-RECOVER(ready) ind	CCR-sp	C-RECOVER(ready) indication primitive to superior (Recovery State = "ready")
C-RECOVER(done) rsp	Subordinate	C-RECOVER(done) response primitive (Recovery State = "done")
C-RECOVER(done) cnf	CCR-sp	C-RECOVER(done) confirm primitive to superior (Recovery State = "done")
C-RECOVERY(retry-later) rsp	Superior or subordinate	C-RECOVER(retry-later) response primitive (Recovery State = "retry-later")
C-RECOVER(retry-later) cnf	CCR-sp	C-RECOVER(retry-later) confirm primitive to subordinate or superior (Recovery State = "retry-later")
C-RECOVER(unknown) rsp	Superior	C-RECOVER(unknown) response primitive (Recovery State = "unknown")
C-RECOVER(unknown) cnf	CCR-sp	C-RECOVER(unknown) confirm primitive to subordinate (Recovery State = "unknown")

Table 11 – State table for superior – Normal operation

Event	Preceding State													
	I	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
C-BEGIN req	A1													
C-BEGIN cnf		A2		A4										
C-PREPARE req		A3	A4											
C-READY ind		A5	A5	A5	A5									
C-COMMIT req						A6								
C-COMMIT cnf							I				A1			
C-ROLLBACK req		A7	A7	A7	A7	A8								
C-ROLLBACK cnf								I	I			A1		A1
C-ROLLBACK ind		A9	A9	A9	A9			A9				A12		
C-ROLLBACK rsp										I			A1	
C-COMMIT + C-BEGIN req						A10								
C-ROLLBACK + C-BEGIN req		A11	A11	A11	A11	A13								

Table 12 – State table for subordinate – Normal operation

Event	Preceding State											
	I	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
C-BEGIN ind	B1											
C-BEGIN rsp		B2		B4								
C-PREPARE ind		B3	B4			B6						
C-READY req		B5	B5	B6	B6							
C-COMMIT ind						B7	B7					
C-COMMIT rsp								I			B1	
C-ROLLBACK ind		B8	B8	B8	B8	B8	B8			B8		
C-ROLLBACK rsp									I			B1
C-ROLLBACK req		B9	B9	B9	B9							
C-ROLLBACK cnf										I		
C-COMMIT + C-BEGIN ind						B10	B10					
C-ROLLBACK + C-BEGIN ind		B11	B11	B11	B11	B11	B11			B11		

Table 13 – State table for superior – Recovery

Event	Preceding State		
	I	X1	X2
C-RECOVER (commit) req	X1		X1
C-RECOVER (done) cnf		I	
C-RECOVER (retry-later) cnf		I	
C-RECOVER (ready) ind	X2		
C-RECOVER (unknown) rsp			I
C-RECOVER (retry-later) rsp			I

Table 14 – State table for subordinate – Recovery

Event	Preceding State		
	I	Y1	Y2
C-RECOVER (commit) ind	Y1		Y1
C-RECOVER (done) rsp		I	
C-RECOVER (retry-later) rsp		I	
C-RECOVER (ready) req	Y2		
C-RECOVER (unknown) cnf			I
C-RECOVER (retry-later) cnf			I

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 10. 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 states used in the state tables are represented by the notation Zn assigned in Table 9, where Z is an upper-case letter (I, A, B, C, X or Y) and n is null or is an integer.

8.4 Interpretation of the state table

8.4.1 For both the superior and subordinate, the CCR ASE is initialized in the idle state (I). This occurs when the CCR ASE is initially used on the association. If the supporting association is normally or abnormally released, the CCR ASE ceases to exist.

8.4.2 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.

8.4.3 A blank cell represents the combination of an event and a state that is not defined for the CCR ASE. Any action taken by the CCR service-user is a local matter.

8.5 Completing the branch

8.5.1 For the superior, a branch is completed by any of following:

a) C-COMMIT confirm primitive; or

b) C-ROLLBACK confirm primitive; or

c) C-ROLLBACK response primitive; or

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

e) application or communication failure before it issues a C-COMMIT request primitive.

8.5.2 For a subordinate, a branch is completed by any of the following:

a) C-COMMIT response primitive; or

b) C-ROLLBACK response primitive; or

c) C-ROLLBACK confirm primitive; or

d) C-RECOVER(done) response primitive; or

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

f) application or communication failure before it issues a C-READY request.

8.6 Collisions and disruptive services

8.6.1 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.

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

8.6.3 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.

8.6.4 A 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.

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 by a cooperating main service

9.2.1 CCR may be incorporated into a referencing specification by providing a parameter in its service primitives to carry CCR semantics. Such a referencing specification is called a **cooperating main service**. Annex B in ITU-T Rec. X.852 | ISO/IEC 9805 defines the operating characteristics for the use of CCR by a cooperating main service.

9.2.2 Restrictions on token usage defined in this Recommendation | International Standard may not apply to a cooperating main service. However, the referencing specification is responsible for preventing collision situations that are not covered by CCR sequencing rules (see clause 8).

9.3 Use of resynchronization services with CCR Protocol Version 1

9.3.1 CCR services cannot be used by a referencing specification that uses session synchronization points and session resynchronization in a manner unrelated to CCR semantics.

9.3.2 A referencing specification may use session synchronization points and session resynchronization for a check pointing facility. This may or may not be visible in the service primitives of that referencing specification.

9.4 Use of session synchronization and resynchronization services with CCR Protocol Version 2

9.4.1 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.

9.4.2 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.

9.5 Use of CCR with session activities

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

9.6 Use of transport expedited service with CCR Protocol Version 1

9.6.1 If the transport-expedited service is used by the Session Layer, the CCR service-user shall

a) respond to a C-BEGIN indication primitive with a C-BEGIN response primitive; and

b) not issue C-ROLLBACK request primitive until after receiving the C-BEGIN confirmation primitive.

9.6.2 If the transport expedited service is not used by the Session Layer, these restrictions do not apply.

NOTE – The use of the session resynchronize service for C-ROLLBACK may cause purging of user-data outside the atomic-action. If the transport expedited service is used by session and the above restrictions are not followed, the C-BEGIN and user-data preceding it may be purged. It is expected that a future change to session will prevent this possibility and remove these restrictions.

9.7 Use of presentation services with CCR Protocol Version 2

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.

9.8 Starting a branch in CCR Protocol Version 1

If a request primitive of a referencing specification invokes a function whose completion (or failure) is signaled by a corresponding confirm (or indication) primitive, then a C-BEGIN request primitive cannot be issued between the referencing specification's request and confirm (or indication) primitives. This restriction prevents ambiguity about the precise point in time when the branch begins.

Annex A

CCR service-user rules

(This annex forms an integral part of this Recommendation I 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.1.2 Heuristic decision considerations

A.1.2.1 If a CCR service-user takes a heuristic decision (see 6.3), this Recommendation I International Standard relaxes the requirement to adhere to a number of CCR service-user rules necessary to guarantee atomicity.

A.1.2.2 The following bound data manipulation rules are relaxed in the presence of a heuristic decision:

Rule A.3.5-b;

Rule A.3.7; and

Rule A.3.10-b.

A.1.2.3 The following multi-branch sequence rules are relaxed in the presence of a heuristic decision:

Rule A.3.4.1-a;

Rule A.3.5-a;

Rule A.3.6.1-b;

Rule A.3.9.1-a; and

Rule A.3.10-a.

A.1.2.4 All rules not listed above shall always be followed even if atomicity is not required. The only exception is when CCR is used by a cooperating main service (see A.3).

A.2 Compliance

A.2.1 To be compliant with this Recommendation I 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 by a cooperating main service (see 9.2).

A.3.1 C-BEGIN request primitive

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

a) a C-READY request primitive has been issued to the superior if this is a leaf or intermediate (multi-branch sequence rule – Rule A.3.1-a); or

b) a C-ROLLBACK request primitive has been issued to the superior for a leaf or intermediate (multi-branch sequence rule – Rule A.3.1-b); or

c) a C-COMMIT request primitive has been issued to a subordinate for a master or intermediate (multi-branch sequence rule – Rule A.3.1-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.3.1-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.3.1-e).

A.3.2 C-PREPARE request primitive

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

A.3.3 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 all its subordinates in the atomic action tree that will not be rolled back (multi-branch sequence rule - Rule A.3.3-a); and

b) atomic action data have been recorded that are required for an offer of commitment as defined in A.4.1 (multi-branch and single branch recovery rule – Rule A.3.3-b); and

c) bound data can be released in either the initial or the final state (bound data rule – Rule A.3.3-c).

NOTE – The phrase "will not be rolled back" covers the situation in which a CCR service-user has determined that a subordinate will be rolled back, but the subordinate has not yet been ordered to roll back.

A.3.4 C-COMMIT request primitive

A.3.4.1 A C-COMMIT request primitive can only be issued if:

a) the CCR service-user has received a C-COMMIT or C-RECOVER(commit) indication primitive from the superior if it is an intermediate (multi-branch sequence rule – Rule A.3.4.1-a); and

b) the CCR service-user has received a C-READY or C-RECOVER(ready) indication primitive from all its subordinates that will not be rolled back for a master (multi-branch sequence rule – Rule A.3.4.1-b); and

c) 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.3.4.1-c); and

d) bound data can be released in the final state for a master (bound data rule – Rule A.3.4.1-d); and

e) the CCR service-user owns the major/activity token if CCR Protocol Version 1 is being used, or the synchronized-minor token if CCR Protocol Version 2 is being used (association use rule – Rule A.3.4.1-e).

A.3.4.2 If a C-COMMIT request primitive is issued by a master, the master shall release bound data in the final state (bound data rule – Rule A.3.4.2).

A.3.5 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 its subordinates in the atomic action tree 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.3.5-a); and

b) bound data have been released in the final state (bound data rule – Rule A.3.5-b); and

c) atomic action data for this branch have been forgotten (single branch recovery rule – Rule A.3.5-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.3.5-d).

A.3.6 C-ROLLBACK request primitive

A.3.6.1 A C-ROLLBACK request primitive can only be issued to a subordinate by an intermediate if:

a) a C-READY or C-RECOVER(ready) request primitive has not been issued to the superior (multi-branch sequence rule – Rule A.3.6.1-a); or

b) a C-ROLLBACK indication primitive or a C-RECOVER(unknown) confirmation primitive has been received from the superior (multi-branch sequence rule – Rule A.3.6.1-b).

A.3.6.2 If a C-ROLLBACK request primitive is issued by an intermediate or leaf to a superior, bound data shall be released in the initial state (bound data rule – Rule A.3.6.2).

A.3.7 C-ROLLBACK indication primitive

If C-ROLLBACK indication primitive is received from the superior, bound data shall be released in the initial state (bound data rule – Rule A.3.7).

A.3.8 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 all its subordinates in the atomic action tree that will not be rolled back (multi-branch sequence rule – Rule A.3.8-a); and

b) atomic action data have been recorded that are required for an offer of commitment as defined in A.4.1 (multi-branch and single branch recovery rule – Rule A.3.8-b); and

c) bound data can be released in either the initial or the final state (bound data rule – Rule A.3.8-c); and

d) the CCR service-user is the owner of the synchronize-minor token (association use rule – Rule A.3.8-d).

A.3.9 C-RECOVER(commit) request primitive

A.3.9.1 A C-RECOVER(commit) request primitive can only be issued if:

a) the CCR service-user has received a C-COMMIT or C-RECOVER(commit) indication primitive from the superior if it is an intermediate (multi-branch sequence rule – Rule A.3.9.1-a); and

b) the CCR service-user has received a C-READY or C-RECOVER(ready) indication primitive from all its subordinates that will not be rolled back for a master (multi-branch sequence rule – Rule A.3.9.1-b); and

c) atomic action data required to perform recovery have been recorded for all subordinate branches that will not be rolled back for a master (multi-branch recovery rule – Rule A.3.9.1-c); and

d) bound data can be released in the final state for a master (bound data rule – Rule A.3.9.1-d); and

e) the CCR service-user is the owner of the synchronize-minor token unless the primitive is in answer to a C-RECOVER(ready) indication from a subordinate (association use rule – Rule A.3.9.1-e).

A.3.9.2 If a C-RECOVER(commit) request primitive is issued by a master, the master shall release bound data in the final state (bound data rule – Rule A.3.9.2).

A.3.10 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 its subordinates in the atomic action tree 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.3.10-a); and

b) bound data have been released in the final state (bound data rule – Rule A.3.10-b); and

c) atomic action data for this branch have been forgotten (single branch recovery rule – Rule A.3.10-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.3.10-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.

A.4.1 Recording atomic action data that indicates an offer of commitment

Atomic action data indicating an offer of commitment can only be recorded if:

a) Rule A.3.3-a is satisfied; and

b) Rule A.3.3-c is satisfied.

The following atomic action data are recorded by a CCR service-user for an offer of commitment:

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

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

A.4.2 Forgetting atomic action data that indicates an offer of commitment

Atomic action data indicating an offer of commitment can only be forgotten if:

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

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

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.3.4.1-a is satisfied; and

b) Rule A.3.4.1-b is satisfied; and

c) Rule A.3.4.1-d is satisfied.

A.4.3.2 The following atomic action data are recorded by a master CCR service-user for an order of commitment:

– atomic action data required to perform recovery for all its 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 subordinate.

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 an offer of commitment 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 superior if an offer of commitment is recorded for an intermediate or leaf CCR service-user; and

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

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 superior if it is an intermediate or leaf; and

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

c) atomic action data reflect either an offer of commitment 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 a ready decision.

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 an offer of commitment.

Annex B

Relationship of CCR to the Application Layer Structure

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

The material in the annex is aligned with ITU-T Rec. X.207 | ISO/IEC 9545.

NOTE – Figure B.1 in this annex 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 tree

Figure B.1 is an example of an atomic action tree based on this architecture. It includes master, intermediate, and leaf CCR service-users.

NOTE – An intermediate CCR service-user has both the role of subordinate and one or more roles of superior. This is not visible to the CCR service-provider.

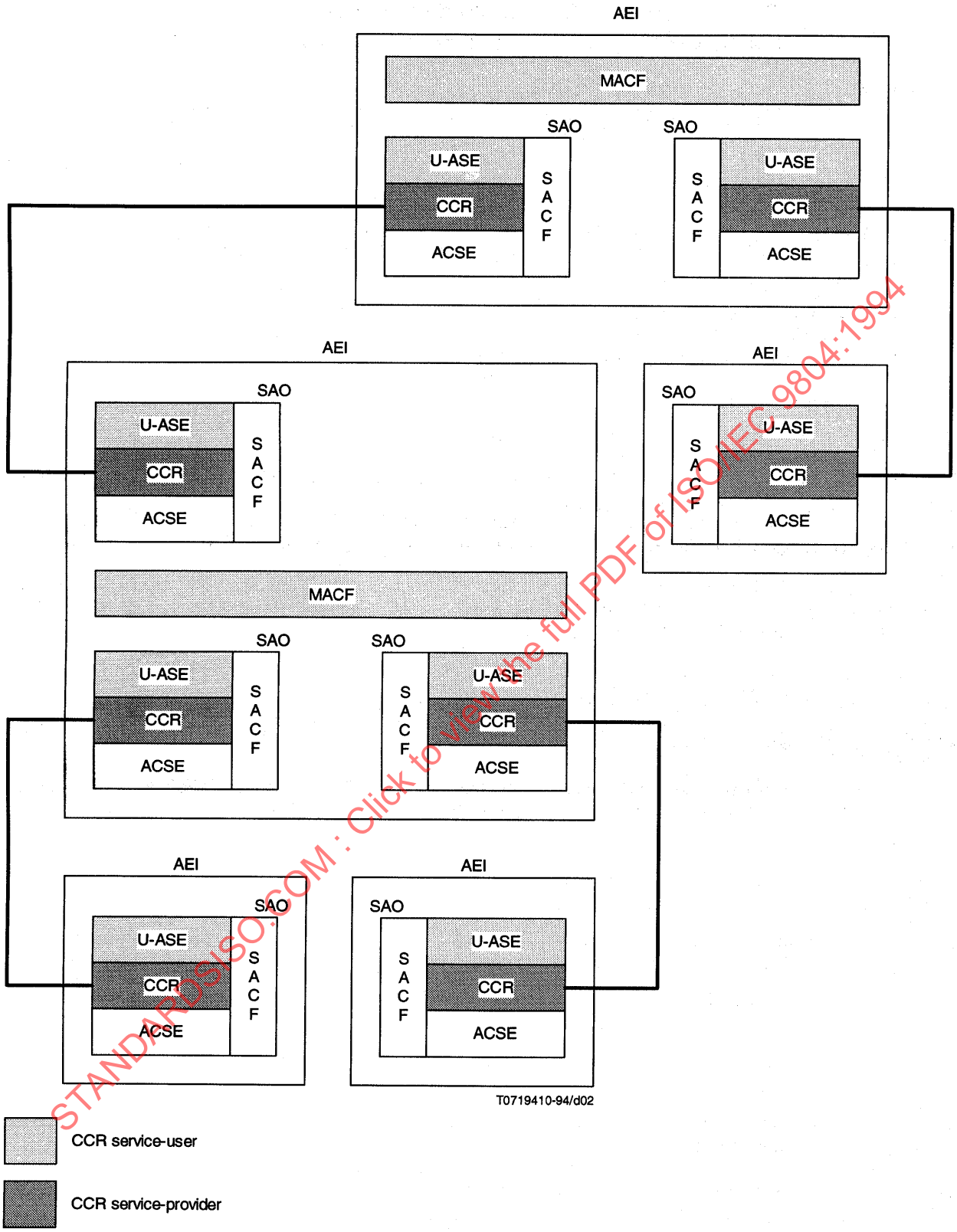

Figure B.1 – Atomic action tree architecture

Annex C

CCR tutorial

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

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.

All the terms defined in the normative part of this document 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.

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 (ITU-T Rec. X.851 | ISO/IEC 9804) and a protocol specification (ITU-T Rec. X.852 | ISO/IEC 9805). The usage rules (called CCR service-user rules) are 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) 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.

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.

The CCR Recommendations X.851 and X.852 | International Standards ISO/IEC 9804 and 9805 define the CCR ASE. CCR does not define the SACF (single association control function) or MACF (multiple association control function). 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

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.

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.

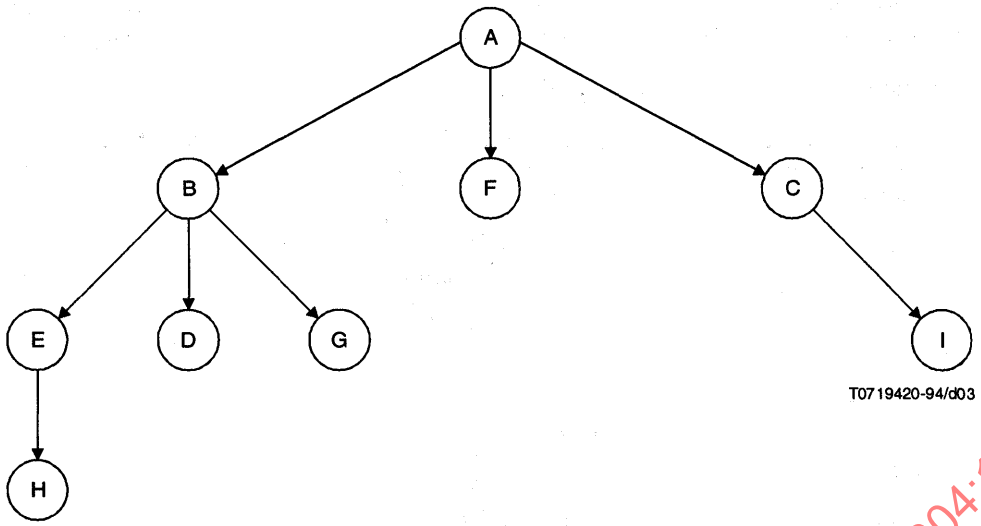

Figure C.1 – Atomic action tree

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 a master, 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.

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.

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 master of the atomic action. The branch identifier is assigned by the superior of the branch.

An atomic action identifier is made up of two parts: the AE title of the master and a suffix assigned by the master. 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 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 master 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.

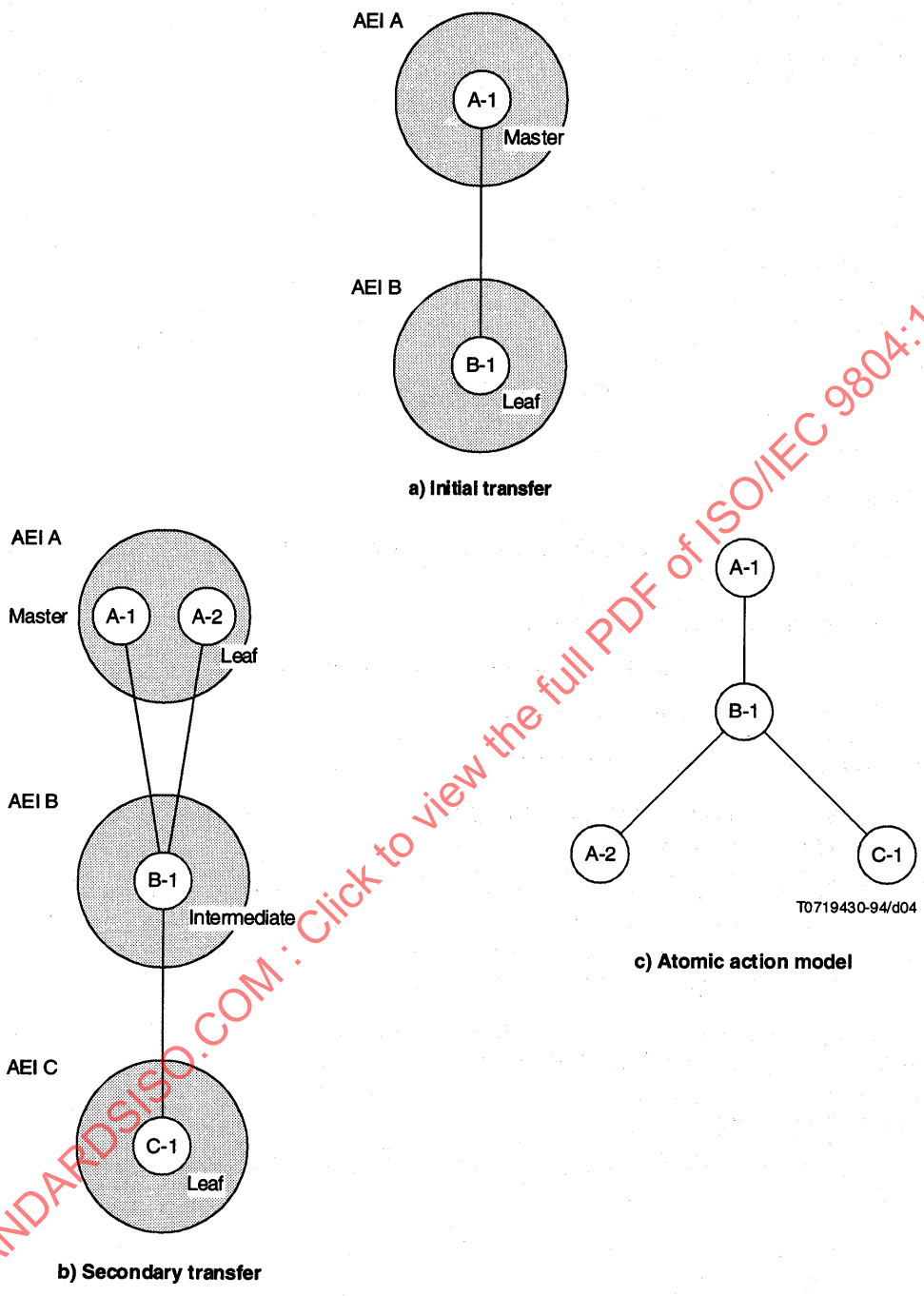

Figure C.2

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.

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. A master'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.

NOTE – 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.

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[1)] 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 document assumes that the loss of bound data occurs infrequently.

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

C.3.2 Atomic action data

Atomic action data is 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 document 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.

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.

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: – Master's AE title – Suffix
CCR service-user information: – AE title – API identifier – AEI identifier – Role [master / 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

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[2]. 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.

---

2) The branch to the superior is the only one on which it is 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 master. If, after it has received offers of commitment from all its subordinates (see A.3.4), the master decides to order commitment, 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 master'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.

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.