INTERNATIONAL STANDARD

ISO/IEC 9805

First edition
1990-11-15

Information technology — Open Systems Interconnection — Protocol specification for the Commitment, Concurrency and Recovery service element

Technologies de l'information — Interconnexion de systèmes ouverts — Spécification du protocole pour l'élément de service d'engagement, concurrence et reprise

Reference number
ISO/IEC 9805:1990(E)

Contents

Foreword

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

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

International Standard ISO/IEC 9805 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Annexes A and B form an integral part of this International Standard.

Introduction

This International Standard is one of a set of International Standards produced to facilitate the interconnection of information processing systems. It is related to other International Standards in the set as defined by the Reference Model for Open Systems Interconnection (ISO 7498). The Reference Model subdivides the area of standardization for interconnection into a series of layers of specification, each of manageable size.

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

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

This International Standard specifies the protocol for the application-service-element for commitment, concurrency, and recovery (CCR). These services are intended to be applicable to a wide range of application-process communication requirements.

The CCR protocol specification consists of the following main components:

a) the specification of the CCR APDUs using Abstract Syntax One (ASN.1, ISO 8824);

b) the elements of procedure for issuing CCR service indication and confirm primitives to the CCR service-user when CCR APDUs are received and for the sending of CCR APDUs when CCR service request and indication primitives are received from the CCR service-user;

c) the CCR protocol machine specified in terms of a state table; and

d) the presentation services (ISO 8822) used for sending and receiving CCR APDUs.

The CCR protocol shares the presentation-service with other application-service-elements.

The requirement to provide support for CCR together with other application-service-elements is satisfied by reference to this International Standard.

Annex A contains the definitions of the structure of the CCR APDUs.

Annex B describes the transfer of CCR APDUs as the values of a special parameter of another referencing application-service-element. Such an application-service-element is called a co-operating main service.

Information technology – Open Systems Interconnection – Protocol specification for the Commitment, Concurrency and Recovery service element

1 Scope

This International Standard is to be applied by reference from other specifications. This is done within such specifications by reference to the CCR services defined in ISO/IEC 9804. A reference to a CCR service invokes the procedures of this International Standard to cause external effects.

This International Standard applies whenever the use of CCR services does not encompass any communication activity which makes direct or indirect use of the session activity management services defined in ISO 8326. It can be used inside a session activity, and on a session-connection where the session activity functional unit is not in use. It can also be applied when the S-ACTIVITY service is used through the mechanisms of annex B.

This International Standard specifies the static and dynamic conformance requirements for systems implementing these procedures. It does not contain tests which can be used to demonstrate conformance.

2 Normative references

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

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

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

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

ISO 8326:1987/Add.2:—[1], *Information processing systems – Open Systems Interconnection – Basic connection oriented session service definition – Addendum 2: Unlimited user data.*

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

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

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

ISO 8824:1990, *Information processing systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1).*

ISO 8825:1990, *Information processing systems – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

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

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

3 Definitions

3.1 Reference model definitions

This International Standard makes use of the following terms defined in ISO 7498:

a) Application Layer;

b) application association; association;

c) application-process;

d) application-entity;

e) presentation-service;

1) To be published.

f) presentation-connection;

g) session-service; and

h) session-connection.

3.2 Naming and addressing definitions

This International Standard makes use of the following terms defined in ISO 7498-3:

a) application-process title;

b) application-entity qualifier;

c) application-entity title;

3.3 Service conventions definitions

This International Standard makes use of the following terms defined in ISO/TR 8509:

a) service-provider;

b) service-user;

c) confirmed service;

d) non-confirmed service;

e) primitive;

f) request (primitive);

g) indication (primitive);

h) response (primitive); and

i) confirm (primitive).

3.4 Presentation service definitions

This International Standard makes use of the following terms defined in ISO 8822:

a) abstract syntax;

b) abstract syntax name;

c) defined context set;

d) presentation context; and

e) presentation data value.

3.5 ACSE service definitions

This International Standard makes use of the following terms defined in ISO 8649:

a) association-initiator; and

b) association-responder.

3.6 Application Layer Structure definitions

This International Standard makes use of the following terms defined in ISO/IEC 9545:

a) application-entity-invocation;

b) application-service-element;

c) multiple association control function;

d) single association control function; and

e) single association object.

3.7 CCR service definitions

This International Standard makes use of the following terms defined in ISO/IEC 9804:

1) acceptor;

2) application failure;

3) atomic action;

4) atomic action branch; branch;

5) atomic action branch identifier; branch identifier;

6) atomic action data;

7) atomic action identifier;

8) atomic action tree;

9) atomicity;

10) bound data;

11) CCR service-provider;

12) CCR service-user;

13) commitment of an atomic action branch; commitment;

14) communication failure;

15) concurrency control;

16) cooperating main service;

17) distributed application;

18) doubt period;

19) durability;

20) final state;

21) heuristic decision;

22) initial state;

23) intermediate CCR service-user; intermediate;

24) intermediate state;

25) interrupted branch;

26) isolation;

27) leaf CCR service-user; leaf;

28) local commitment procedures;

29) local rollback procedures;

30) master CCR service-user; master;

31) offer of commitment of an atomic action branch; offer of commitment;

32) order of commitment of an atomic action branch; order of commitment;

33) phase I;

34) phase II;

35) presumed rollback;

36) recovery control;

37) recovery responsibility for an atomic action branch; recovery responsibility;

38) referencing specification;

39) requestor;

40) rollback of an atomic action branch; rollback;

41) subordinate of an atomic action branch; subordinate; and

42) superior of an atomic action branch; superior.

3.8 CCR protocol specification definitions

For the purpose of this International Standard, the following definitions apply.

3.8.1 accepting CCR protocol machine: The CCR protocol machine whose service-user is the acceptor for a particular CCR service.

3.8.2 CCR protocol machine: the protocol machine of the CCR application-service-element specified in this International Standard.

3.8.3 requesting CCR protocol machine: The CCR protocol machine whose service-user is the requestor for a particular CCR service.

4 Symbols and abbreviations

4.1 Data units

APDU	application-protocol-data-unit

4.2 Types of application-protocol-data-units

The following abbreviations have been given to the application-protocol-data-units defined in this International Standard.

C-BEGIN-RI

C-BEGIN-RC

C-PREPARE-RI

C-READY-RI

C-ROLLBACK-RI

C-ROLLBACK-RC

C-COMMIT-RI

C-COMMIT-RC

C-RECOVER-RI

C-RECOVER-RC

4.3 Other abbreviations

The following abbreviations are used in this International Standard.

ACSE	Association Control Service Element
AE	application-entity
AEI	application-entity invocation
AP	application-process
APDU	application-protocol-data-unit
ASE	application-service-element
ASN.1	Abstract Syntax Notation One
CCR	Commitment, concurrency, and recovery application-service-element
CCRPM	CCR protocol machine
cnf	confirm primitive
ind	indication primitive
OSI	Open Systems Interconnection
req	request primitive
rsp	response primitive

5 Conventions

5.1 This International Standard employs a tabular presentation of its APDU fields. In clause 7, tables are presented for each CCR APDU. Each field is summarized using the following notation:

M	presence is mandatory
O	presence is CCRPM option
U	presence is CCR service-user option
req	source is the related request primitive
ind	sink is the related indication primitive
rsp	source is the related response primitive
cnf	sink is the related confirm primitive
CCRPM	source or sink is the CCRPM

5.2 The structure of each CCR APDU is specified in annex A using the abstract syntax notation of ASN.1 (ISO 8824).

5.3 CCR allows the concatenation of some of its APDUs. In clause 9 an ASN.1-like notation is used to express the allowed concatenations.

6 Overview of the CCR protocol

6.1 Service support

The protocol specified in this International Standard supports the services defined in ISO/IEC 9804. These services are listed in table 1.

6.2 Constraints on ACSE services

6.2.1 An application-entity invocation (AEI) establishes an association to exchange CCR APDUs with another AEI by using the A-ASSOCIATE service of ACSE (ISO 8649).

6.2.2 When establishing the association, the following Presentation and Session Requirements must be specified on the A-ASSOCIATE service:

presentation kernel functional unit

session kernel functional unit

session typed data functional unit

session major synchronize functional unit

session minor synchronize functional unit

session resynchronize functional unit

6.2.3 When establishing the association, the following optional parameters of the ACSE A-ASSOCIATE service must be specified:

a) Calling AP title

b) Calling AE qualifier

c) Responding AP title

d) Responding AE qualifier

6.3 Use of the presentation service

6.3.1 CCR uses the following presentation (ISO 8822) services:

P-DATA
P-TYPED-DATA
P-SYNC-MAJOR
P-SYNC-MINOR
P-RESYNCHRONIZE(restart)

6.3.2 CCR APDUs are passed in the User Data parameters of the above presentation services as one or more presentation data values. The value of the ASN.1 data type for each CCR APDU is specified in annex A. If more than one ASN.1 data type is sent, a corresponding number of presentation data values are included.

6.3.3 If other presentation data values are present on a presentation service primitive, the referencing specification shall specify sequencing rules. These rules ensure that the CCR semantics are maintained and comply with the concatenation and mapping rules specified in clauses 9 and 10.

NOTE – The use of presentation-service parameters other than User Data is specified in clause 9.

6.3.4 It is the responsibility of the CCR service-user to control the presentation contexts available in the defined context set of the underlying presentation-connection.

6.4 Relationship to the session-service and the transport-service.

6.4.1 The session functional units required for the session-connection that supports the presentation-connection (that in turn supports the association) are determined by the A-ASSOCIATE service requestor and acceptor. They accomplish this using the Session Requirements parameter on the A-ASSOCIATE primitives. The required session functional units are given in 6.2.

6.4.2 The rules of the session-service affect the operation of the CCRPM and its service-user. The CCR service-user must be aware of these constraints. This International Standard assumes that a local mechanism enforces them. For example, it is the responsibility of the CCR service-user to control the possession of the available session tokens.

6.4.3 If the Transport-expedited service is used by the session layer, the CCR service-user:

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

b) following a C-BEGIN request, shall not issue C-ROLLBACK request until after receipt of a C-BEGIN confirmation.

If the Transport-expedited service is not used by the session layer, these restrictions do not apply.

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 or Optionally confirmed	Superior Subordinate

NOTE — The use of the session resynchronize service for C-ROLLBACK is liable to 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 can be purged and user-data preceeding it. It is expected that a future change to session will prevent this possibility and thus allow the removal of this requirement.

6.4.4 CCR requires use of session unlimited user data (see ISO 8326 : 1987/Add.2).

6.5 Operation of the CCRPM

6.5.1 The protocol specification for CCR is presented in this International Standard as a protocol machine. This protocol *machine is referred to as the CCR protocol machine* (CCRPM).

6.5.2 A CCRPM is used for a protocol exchange sequence for one atomic action branch on an existing association. A CCRPM is also used for a sequence of atomic action branches in which the completion (commitment or rollback) of one overlaps with the beginning of the next one. The procedures of a CCRPM are performed in co-operation with the overall CCR service-user. The CCRPM shares the presentation-connection that supports the association with other ASEs.

6.5.3 A CCR service primitive is issued by a CCR service-user within a sequence of application or presentation service primitives on a single association, as defined in ISO/IEC 9804.

6.5.4 The procedures specified in clause 7 are carried out as a result of the request and response primitives issued in conformance with the CCRPM State Table defined in clause 8 and as a result of the receipt of presentation service primitives carrying data values in the CCR presentation context. The parameters of the CCR service primitives are structured according to annex A to produce CCR APDUs. These APDUs are transferred using the presentation-service according to the specification given in clauses 7, 9, and 10.

6.5.5 The value of a CCR APDU is transferred as a presentation data value from the CCR presentation context. The abstract syntax for data types transferred in this context are defined in annex A by specifying the complete set of CCR APDUs using Abstract Syntax Notation One (ASN.1, ISO 8824).

7 Elements of procedures

The CCR protocol consists of the following procedures:

a) begin branch;

b) prepare subordinate;

c) offer commitment;

d) order commitment;

e) rollback;

f) branch recovery;

g) order commitment and begin new branch; and

h) rollback and begin new branch.

The following subclauses describe these procedures. The descriptions include the specification of presentation service primitives normally used to carry CCR APDUs. However, for concatenated CCR APDUs, the presentation service mapping specified in clause 10 applies.

Figures 1 to 6 show the ASN.1 structure of the CCR APDUs. The complete ASN.1 module, containing these definitions and those of the supporting datatypes, is in annex A.

7.1 Begin branch procedure

7.1.1 Purpose

This procedure is used to begin a new atomic action branch between two CCR-service users. It supports the C-BEGIN service defined in ISO/IEC 9804.

7.1.2 APDUs used

The procedure uses the following CCR APDUs:

C-BEGIN-RI
C-BEGIN-RC

The structure of these APDUs is shown in figure 1.

The C-BEGIN-RI APDU fields are listed in table 2. The C-BEGIN-RC APDU field is listed in table 3.

7.1.3 Prerequisite requirements

7.1.3.1 For the requestor, the use of this procedure requires that no other atomic action branch is active on the association.

7.1.3.2 The requestor of the C-BEGIN request primitive shall be the owner of the session synchronize-minor token.

Table 2 — C-BEGIN-RI fields

Field name	Presence	Source	Sink
atomic-action-identifier	M	req	ind
branch-suffix	M	req	ind
user-data	U	req	ind

Table 3 — C-BEGIN-RC field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.1.4 Procedure operation

This procedure is driven by the following events:

a) C-BEGIN request primitive from the requestor;

b) C-BEGIN-RI APDU received by the accepting CCRPM;

c) C-BEGIN response primitive from the acceptor; and

d) C-BEGIN-RC APDU received by the requesting CCRPM.

The events c) and d) are optional and may occur later.

7.1.4.1 C-BEGIN request primitive

The requesting CCRPM forms a C-BEGIN-RI APDU from parameter values of the C-BEGIN request primitive. If the C-BEGIN-RI is not concatenated with other CCR APDUs, the CCRPM issues a P-SYNC-MINOR request primitive with the APDU as a data value of the primitive's User Data parameter. If the CCRPM concatenates the C-BEGIN-RI APDU with another CCR APDU, it issues the appropriate presentation service primitive as specified in clause 10, with the C-BEGIN-RI APDU as a data value in the user data parameter.

7.1.4.2 C-BEGIN-RI APDU

The accepting CCRPM receives a C-BEGIN-RI APDU from its peer as user data on a P-SYNC-MINOR indication primitive, if the APDU is unconcatenated. If the APDU is concatenated with other CCR APDUs, the C-BEGIN-RI APDU will be received as user data on the appropriate presentation primitive as specified in clause 10. In either case, the CCRPM issues a C-BEGIN indication primitive with parameter values derived from the APDU.

```
C-BEGIN-RI ::= [1] SEQUENCE
     {  atomic-action-identifier       [0] ATOMIC-ACTION-IDENTIFIER,
        branch-suffix                  [1] BRANCH-SUFFIX,
        user-data                          User-data     OPTIONAL
     }

C-BEGIN-RC ::= [2] SEQUENCE
     {  user-data                          User-data     OPTIONAL }
```

Figure 1 — C-BEGIN APDUs

7.1.4.3 C-BEGIN response primitive

The accepting CCRPM forms a C-BEGIN-RC APDU from the parameter value of the C-BEGIN response primitive. If the C-BEGIN-RC is not concatenated with other CCR APDUs, the CCRPM issues a P-SYNC-MINOR response primitive with the APDU as a data value of the primitive's User Data parameter. If the CCRPM concatenates the C-BEGIN-RC APDU with another CCR APDU, it issues the appropriate presentation service primitive as specified in clause 10, with the C-BEGIN-RC APDU as a data value in the user data parameter.

7.1.4.4 C-BEGIN-RC APDU

The requesting CCRPM receives a C-BEGIN-RC APDU from its peer as user data on a P-SYNC-MINOR confirm primitive, if the APDU is unconcatenated. If the APDU is concatenated with other CCR APDUs, the C-BEGIN-RC APDU will be received as user data on the appropriate presentation primitive as specified in clause 10. In either case, the CCRPM issues a C-BEGIN confirm primitive with the parameter value derived from the APDU.

7.1.5 Use of the C-BEGIN-RI APDU fields

For the requesting CCRPM: the fields of the C-BEGIN-RI APDU are directly mapped from the corresponding parameters on the C-BEGIN request primitive as specified in table 4.

The C-BEGIN request includes the Branch Identifier - Superior's Name parameter on the request primitive. The parameter value is the requestor's AE title which was passed in the A-ASSOCIATE service used to establish the supporting association and is not mapped to a field of the C-BEGIN-RI APDU.

For the accepting CCRPM: the fields of the C-BEGIN-RI APDU are directly mapped to the corresponding parameters on the C-BEGIN indication primitive as specified in table 4.

The accepting CCRPM also includes the Branch Identifier - Superior's Name parameter on the indication primitive. The parameter value is the requestor's AE title passed in the A-ASSOCIATE service used to establish the supporting association.

7.1.6 Use of the C-BEGIN-RC APDU field

For the accepting and requesting CCRPM: the C-BEGIN-RC APDU field is directly mapped to and from the corresponding parameter on the C-BEGIN response and confirm primitives as specified in table 5.

Table 4 — Mapping of C-BEGIN req/ind parameters

APDU field name	Parameter name
atomic-action-identifier {masters-name}	Atomic Action Identifier - Master's Name
atomic-action-identifier {atomic-action-suffix}	Atomic Action Identifier - Suffix
—	Branch Identifier - Superior's Name
branch-suffix	Branch Identifier - Suffix
user-data	User Data

Table 5 —Mapping of C-BEGIN rsp/cnf parameter

APDU field name	Parameter name
user-data	User Data

7.1.7 Collisions

A collision of a C-BEGIN-RI APDU with another CCR APDU cannot occur.

NOTE — Collisions between two C-BEGIN-RI APDUs cannot occur because the CCR service-user must own the synchronize-minor token when issuing C-BEGIN request (except when issued with C-ROLLBACK or C-COMMIT). The requirement to own the token before issuing C-RECOVER request (except when replying to a C-RECOVER indication) makes collisions of C-BEGIN-RI APDUs and C-RECOVER-RI APDUs impossible.

7.2 Prepare subordinate procedure

7.2.1 Purpose

The prepare subordinate procedure is used by the superior to request that the subordinate complete processing for the atomic action branch and use the offer commitment procedure (see 7.3) to complete the atomic action branch. If offering commitment is not possible, the subordinate uses the rollback procedure (see 7.5) to force completion of the atomic action branch. The prepare subordinate procedure supports the C-PREPARE service defined in ISO/IEC 9804.

7.2.2 APDU used

The procedure uses the following CCR APDU.

C-PREPARE-RI

The structure of this APDU is shown in figure 2.

The C-PREPARE-RI APDU field is listed in table 6.

Table 6 — C-PREPARE-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.2.3 Prerequisite requirements

None.

7.2.4 Prepare subordinate procedure

This procedure is driven by the following events:

a) C-PREPARE request primitive from the requestor; and

```
C-PREPARE-RI ::= [3] SEQUENCE
    {  user-data                  User-data    OPTIONAL }
```

Figure 2 — C-PREPARE APDU

b) C-PREPARE-RI APDU received by the accepting CCRPM.

7.2.4.1 C-PREPARE request primitive

The requesting CCRPM forms a C-PREPARE-RI APDU from the parameter value of the C-PREPARE request primitive. If the C-PREPARE-RI is not concatenated with other CCR APDUs, the CCRPM issues a P-TYPED-DATA request primitive with the APDU as a data value of the primitive's User Data parameter. If the CCRPM concatenates the C-PREPARE-RI APDU with another CCR APDU, it issues the appropriate presentation service primitive as specified in clause 10, with the C-PREPARE-RI APDU as a data value in the user data parameter.

7.2.4.2 C-PREPARE-RI APDU

The accepting CCRPM receives a C-PREPARE-RI APDU from its peer as user data on a P-TYPED-DATA indication primitive, if the APDU is unconcatenated. If the APDU is concatenated with other CCR APDUs, the C-PREPARE-RI APDU will be received as user data on the appropriate presentation primitive as specified in clause 10. In either case, the CCRPM issues a C-PREPARE indication primitive with the parameter value derived from the APDU.

7.2.5 Use of the C-PREPARE-RI APDU field

For the requesting and accepting CCRPM: the field of the C-PREPARE-RI APDU is directly mapped to and from the corresponding parameter on the C-PREPARE request and indication primitives as specified in table 7.

Table 7 — Mapping of C-PREPARE req/ind parameter

APDU Field Name	Parameter name
user-data	User Data

7.2.6 Collisions

7.2.6.1 The prepare subordinate procedure and the commitment offer procedure may be used simultaneously by the superior and the subordinate, respectively. This results in a collision of a C-PREPARE-RI APDU and a C-READY-RI APDU. Both events are treated normally, resulting in the issue of the appropriate indication primitives.

7.2.6.2 The prepare procedure and the rollback procedure can be used simultaneously by the CCR service-users. The collision is resolved in favour of the rollback procedure.

7.2.6.3 A CCR service-user can initiate the rollback procedure immediately after initiating the prepare procedure. In this case the rollback can disrupt the prepare procedure.

7.3 Offer commitment procedure

7.3.1 Purpose

The offer commitment procedure is used by the subordinate to inform its superior that it is offering commitment. The procedure supports the C-READY service defined in ISO/IEC 9804.

7.3.2 APDU used

This procedure uses the following CCR-APDU.

C-READY-RI

The structure of this APDU is shown in figure 3.

The C-READY-RI APDU field is listed in table 8

Table 8 — C-READY-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.3.3 Prerequisite requirements

For the requestor, the use of this procedure requires that atomic action data for this branch are accessible in stable storage.

7.3.4 Offer commitment procedure

This procedure is driven by the following events:

a) C-READY request primitive from the requestor; and

b) C-READY-RI APDU received by the accepting CCRPM.

7.3.4.1 C-READY request primitive

The requesting CCRPM forms the C-READY-RI APDU from the parameter value of the C-READY request primitive. If the C-READY-RI is not concatenated with other CCR APDUs, the CCRPM issues a P-TYPED-DATA request primitive with the APDU as a value of the primitive's User Data parameter. If the CCRPM concatenates the C-READY-RI APDU with another CCR APDU, it issues the appropriate

```
C-READY-RI ::= [4] SEQUENCE
    {  user-data                  User-data    OPTIONAL }
```

Figure 3 — C-READY APDU

presentation service primitive as specified in clause 10, with the C-READY-RI APDU as a data value in the user data parameter.

7.3.4.2 C-READY-RI APDU

The accepting CCRPM receives a C-READY-RI APDU from its peer as user data on a P-TYPED-DATA indication primitive, if the APDU is unconcatenated. If the APDU is concatenated with other CCR APDUs, the C-READY-RI APDU will be received as user data on the appropriate presentation primitive as specified in clause 10. In either case, the CCRPM issues a C-READY indication primitive with the parameter value derived from the APDU.

7.3.5 Use of the C-READY-RI APDU field

For the requesting and accepting CCRPM: the field of the C-READY-RI APDU is directly mapped to and from the corresponding parameter on the C-READY request and indication primitives as specified in table 9.

Table 9 — Mapping of C-READY req/ind parameter

APDU Field name	Parameter name
user-data	User Data

7.3.6 Collisions

The commitment offer procedure and the prepare subordinate procedure may be used simultaneously by the subordinate and the superior, respectively. This results in a collision of a C-READY-RI APDU and a C-PREPARE-RI APDU. Both events are treated normally, resulting in the issue of the appropriate indication primitives.

7.4 Order commitment

7.4.1 Purpose

The order commitment procedure is used by a superior to request its subordinate to release its bound data in their final state. It supports the C-COMMIT service defined in ISO/IEC 9804.

7.4.2 APDUs used

The procedure uses the following CCR APDUs.

C-COMMIT-RI

C-COMMIT-RC

The structure of these APDUs is shown in figure 4

The C-COMMIT-RI APDU and the C-COMMIT-RC APDU field are listed in table 10 and table 11, respectively.

Table 10 — C-COMMIT-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

Table 11 — C-COMMIT-RC field

Field name	Presence	Source	Sink
user-data	U	rsp	cnf

7.4.3 Prerequisite requirements

For the requestor to issue the C-COMMIT request primitive, it is required that atomic action data for this branch are accessible in stable storage. The requestor shall also be the owner of the session major/activity token.

For the acceptor to issue the C-COMMIT response primitive, it is required that no atomic action data for this branch are accessible in stable storage.

7.4.4 Order commitment procedure

This procedure is driven by the following events:

a) C-COMMIT request primitive from the requestor; and

b) C-COMMIT-RI APDU received by the accepting CCRPM; and

c) C-COMMIT response primitive from the acceptor; and

d) C-COMMIT-RC APDU received by the requesting CCRPM.

7.4.4.1 C-COMMIT request primitive

The requesting CCRPM forms a C-COMMIT-RI APDU from the parameter value of the C-COMMIT request primitive. It issues a P-SYNC-MAJOR request primitive with the APDU as a data value of the primitive's User Data parameter.

7.4.4.2 C-COMMIT-RI APDU

The accepting CCRPM receives a C-COMMIT-RI APDU from its peer as user data on a P-SYNC-MAJOR indication

```
C-COMMIT-RI ::= [5] SEQUENCE
     {  user-data                  User-data    OPTIONAL }


C-COMMIT-RC ::= [6] SEQUENCE
     {  user-data                  User-data    OPTIONAL }
```

Figure 4 — C-COMMIT APDUs

primitive. It issues a C-COMMIT indication primitive with the parameter value derived from the APDU.

7.4.4.3 C-COMMIT response primitive

The accepting CCRPM forms a C-COMMIT-RC APDU from the parameter value of the C-COMMIT response primitive. It issues a P-SYNC-MAJOR response primitive with the APDU as a data value of the primitive's User Data parameter.

7.4.4.4 C-COMMIT-RC APDU

The requesting CCRPM receives a C-COMMIT-RC APDU from its peer as user data on a P-SYNC-MAJOR confirm primitive. It issues a C-COMMIT confirm primitive with the parameter value derived from the APDU.

7.4.5 Use of the C-COMMIT-RI APDU fields

For the requesting and accepting CCRPM: the C-COMMIT-RI APDU field is directly mapped to and from the corresponding parameter on the C-COMMIT request and indication primitives as specified in table 12.

Table 12 — Mapping of C-COMMIT req/ind parameter

APDU Field name	Parameter name
user-data	User Data

7.4.6 Use of the C-COMMIT-RC APDU field

For the accepting and requesting CCRPM: the C-COMMIT-RC APDU field is directly mapped to and from the corresponding parameter on the C-COMMIT response and confirm primitives as specified in table 13.

Table 13 — Mapping of C-COMMIT rsp/cnf parameter

APDU Field Name	Parameter name
user-data	User Data

7.4.7 Collision

None.

7.5 Rollback

7.5.1 Purpose

The rollback procedure is used to force completion of an atomic action branch. It supports the C-ROLLBACK service defined in ISO/IEC 9804.

7.5.2 APDUs used

The procedure uses the following CCR APDUs.

C-ROLLBACK-RI

C-ROLLBACK-RC

The structure of these APDUs is shown in figure 5.

The C-ROLLBACK-RI APDU field is listed in table 14. The C-ROLLBACK-RC APDU field is listed in table 15.

Table 14 — C-ROLLBACK-RI fields

Field name	Presence	Source	Sink
user-data	U	req	ind

Table 15 — C-ROLLBACK-RC fields

Field name	Presence	Source	Sink
user-data	U	rsp	cnf

7.5.3 Prerequisite requirements

For the requestor, the use of this procedure requires either

a) no atomic action data for this branch are accessible in stable storage; or

b) the CCR service-user has been ordered to rollback by its superior.

7.5.4 Rollback procedure

This procedure is driven by the following events:

a) C-ROLLBACK request primitive from the requestor;

b) C-ROLLBACK-RI APDU received by the accepting CCRPM;

c) C-ROLLBACK response primitive from the acceptor; and

```
C-ROLLBACK-RI ::= [7] SEQUENCE
      {  user-data                    User-data    OPTIONAL  }


C-ROLLBACK-RC ::= [8] SEQUENCE
      {  user-data                    User-data    OPTIONAL  }
```

Figure 5 — C-ROLLBACK APDUs

d) C-ROLLBACK-RC APDU received by the requesting CCRPM.

7.5.4.1 C-ROLLBACK request primitive

The requesting CCRPM forms a C-ROLLBACK-RI APDU from the parameter value of the C-ROLLBACK request primitive. It issues a P-RESYNCHRONIZE(restart) request primitive with the APDU as a data value of the primitive's User Data parameter.

7.5.4.2 C-ROLLBACK-RI APDU

The accepting CCRPM receives a C-ROLLBACK-RI APDU from its peer as user data on a P-RESYNCHRONIZE(restart) indication primitive. It issues a C-ROLLBACK indication primitive with the parameter value derived from the APDU.

For the acceptor, if atomic action data for this branch are accessible in stable storage, then it is required that these data will be forgotten.

7.5.4.3 C-ROLLBACK response primitive

The accepting CCRPM forms a C-ROLLBACK-RC APDU from the parameter value of the C-ROLLBACK response primitive. It issues a P-RESYNCHRONIZE(restart) response primitive with the APDU as a data value of the primitive's User Data parameter.

7.5.4.4 C-ROLLBACK-RC APDU

The requesting CCRPM receives a C-ROLLBACK-RC APDU from its peer as user data on a P-RESYNCHRONIZE(restart) confirm primitive. It issues a C-ROLLBACK confirm primitive with the parameter value derived from the APDU.

7.5.5 Use of the C-ROLLBACK-RI APDU fields

For the accepting and requesting CCRPM: the C-ROLLBACK-RI APDU field is directly mapped to and from the corresponding parameter on the C-ROLLBACK request and indication primitives as specified in table 16.

Table 16 — Mapping of C-ROLLBACK req/ind parameters

APDU Field name	Parameter name
user-data	User Data

7.5.6 Use of the C-ROLLBACK-RC APDU field

For the accepting and requesting CCRPM: the C-ROLLBACK-RC APDU field is mapped to and from the corresponding parameter on the C-ROLLBACK response and confirm primitives as specified in table 17.

7.5.7 Disruptive effects

Because the C-ROLLBACK service is mapped on the P-RESYNCHRONIZE(restart) service, CCR APDUs other than a C-ROLLBACK-RI from the association-initiator are discarded (by the underlying session service-provider). This mapping guarantees that rollback takes precedence over all other allowed CCR protocol procedures.

Table 17 — Mapping of C-ROLLBACK rsp/cnf parameter

APDU Field name	Parameter name
user-data	User Data

7.5.8 Collision with a C-ROLLBACK-RI APDU

If two C-ROLLBACK-RI APDUs collide, the C-ROLLBACK-RI APDU from the association-responder is discarded by the underlying session service-provider. That is, the association-initiator wins. Therefore, for the association-responder, the delivery of its user data to the peer is not guaranteed.

7.6 Branch recovery procedure

7.6.1 Purpose

7.6.1.1 The branch recovery procedure is used to recover an atomic action branch after the branch was disrupted by an application or communication failure. The procedure supports the C-RECOVER service defined in ISO/IEC 9804.

7.6.1.2 This procedure either

a) makes a specific, previously disrupted branch active on the association; or

b) is used by the superior of a branch that is in the process of recovery on the association.

Case b) occurs when the superior uses the procedure to send a C-RECOVER-RI(commit) APDU to the subordinate in response to a C-RECOVER-RI(ready) from the subordinate.

7.6.2 APDUs used

The procedure uses the following CCR APDUs.

C-RECOVER-RI

C-RECOVER-RC

The structure of these APDUs is shown in figure 6.

The fields of the C-RECOVER-RI APDU are listed in table 18. The fields of the C-RECOVER-RC APDU are listed in table 19.

7.6.3 Prerequisite requirements

For the requestor, atomic action data for this branch is required to be accessible in stable storage. If this procedure is being used to make a specific, previously disrupted branch active (see 7.6.1), the following is also required:

```
C-RECOVER-RI ::= [9] SEQUENCE
     {  atomic-action-identifier       [0] ATOMIC-ACTION-IDENTIFIER,
        branch-identifier              [1] BRANCH-IDENTIFIER,
        recovery-state                 [2] CHOICE
           {  commit                      [1]  NULL,
              ready                       [2]  NULL} ,
        user-data                          User-data OPTIONAL
     }

C-RECOVER-RC ::= [10] SEQUENCE
     {  atomic-action-identifier       [0] ATOMIC-ACTION-IDENTIFIER,
        branch-identifier              [1] BRANCH-IDENTIFIER,
        recovery-state                 [2] CHOICE
           {  done                        [1]  NULL,
              unknown                     [2]  NULL,
              retry-later                 [3]  NULL} ,
        user-data                          User-data     OPTIONAL
     }
```

Figure 6 — C-RECOVER APDUs

Table 18 — C-RECOVER-RI fields

Field name	Presence	Source	Sink
atomic-action-identifier	M	req	ind
branch-identifier	M	req	ind
recovery-state	M	req	ind
user-data	U	req	ind

Table 19 — C-RECOVER-RC fields

Field name	Presence	Source	Sink
atomic-action-identifier	M	rsp	cnf
branch-identifier	M	rsp	cnf
recovery-state	M	rsp	cnf
user-data	U	rsp	cnf

a) No other atomic action branch is active on this association.

b) The requestor is the owner of the session synchronize-minor token.

NOTE – The branch recovery procedure is mapped on the presentation P-TYPED-DATA service. This ownership requirement is made to avoid a collision of a C-RECOVER-RI APDU with a C-BEGIN-RI APDU.

7.6.4 Branch recovery procedure

This procedure is driven by the following events:

a) C-RECOVER request primitive from the requestor;

b) C-RECOVER-RI APDU received by the accepting CCRPM;

c) C-RECOVER response primitive from the acceptor; and

d) C-RECOVER-RC APDU received by the requesting CCRPM

If the requestor is the superior, all four events occur. If the requestor is the subordinate, the acceptor (i.e., the superior) has two options:

e) it may reply with a C-RECOVER response primitive, thus continuing this procedure; or

f) it may reply with a C-RECOVER request primitive, thus ending this procedure and initiating a new branch recovery procedure (as the requestor).

7.6.4.1 C-RECOVER request primitive

The requesting CCRPM forms a C-RECOVER-RI APDU from parameter values of the C-RECOVER request primitive. The value of the Recovery State parameter is derived by the CCR service-user from the atomic action data. The requesting CCRPM issues a P-TYPED-DATA request primitive with the APDU as a data value of the primitive's User Data parameter.

7.6.4.2 C-RECOVER-RI APDU

The accepting CCRPM receives a C-RECOVER-RI APDU from its peer as user data on a P-TYPED-DATA indication primitive. It issues a C-RECOVER indication primitive with parameter values derived from the APDU.

7.6.4.3 C-RECOVER response primitive

The accepting CCRPM forms a C-RECOVER-RC APDU from parameter values of the C-RECOVER response primitive. The value of the Recovery State parameter is derived by the CCR service user from the atomic action data. The accepting CCRPM issues a P-TYPED-DATA request primitive with the APDU as a data value of the primitive's User Data parameter.

7.6.4.4 C-RECOVER-RC APDU

The requesting CCRPM receives a C-RECOVER-RC APDU from its peer as user data on a P-TYPED-DATA indication primitive. It issues a C-RECOVER confirm primitive with the parameter values derived from the APDU.

7.6.5 Use of the C-RECOVER-RI APDU fields

For the requesting and accepting CCRPM: the fields of the C-RECOVER-RI APDU are directly mapped to and from the corresponding parameters on the C-RECOVER request and indication primitives as specified in table 20.

Table 20 — Mapping of C-RECOVER req/ind parameters

APDU Field name	Parameter name
atomic-action-identifier	Atomic Action Identifier
branch-identifier	Branch Identifier
recovery-state	Recovery state
user-data	User Data

7.6.6 Use of the C-RECOVER-RC APDU fields

For the accepting and requesting CCRPM: the fields of the C-RECOVER-RC APDU are directly mapped to and from the corresponding parameters on the C-RECOVER response and confirm primitives as specified in table 21.

Table 21 — Mapping of C-RECOVER rsp/cnf parameters

APDU Field name	Parameter name
atomic-action-identifier	Atomic Action Identifier
branch-identifier	Branch Identifier
recovery-state	Recovery state
user-data	User Data

7.6.7 Collisions

None.

7.7 Order commitment and begin branch procedure

7.7.1 Purpose

This procedure is used by a superior to request its subordinate to release its bound data in the final state on one atomic action branch, while beginning a new atomic action branch between the two CCR-service users. It supports the C-COMMIT and C-BEGIN services defined in ISO/IEC 9804.

7.7.2 APDUs used

The procedure uses the CCR APDUs specified in 7.1.2 and 7.4.2.

7.7.3 Prerequisite requirements

The prerequisite requirements for this procedure are the same as those for the order commitment procedure, specified in 7.4.3.

7.7.4 Procedure operation

This procedure is driven by the following events:

a) C-COMMIT request primitive + C-BEGIN request primitive from the requestor;

b) C-COMMIT-RI APDU + C-BEGIN-RI APDU received by the accepting CCRPM;

c) C-COMMIT response primitive from the acceptor; and

d) C-COMMIT-RC APDU received by the requesting CCRPM.

NOTE — The C-BEGIN response primitive and the C-BEGIN-RC APDU may optionally occur with c) and d) respectively.

7.7.4.1 C-COMMIT request primitive + C-BEGIN request primitive

The requesting CCRPM forms a C-COMMIT-RI APDU and a C-BEGIN-RI APDU from parameter values of the C-COMMIT request primitive and C-BEGIN request primitive, respectively. It issues a P-SYNC-MAJOR request primitive with the APDUs as data values of the primitive's User Data parameter.

7.7.4.2 C-COMMIT-RI APDU + C-BEGIN-RI APDU

The accepting CCRPM receives a C-COMMIT-RI and a C-BEGIN-RI APDU from its peer as user data on a P-SYNC-MAJOR indication primitive. It issues a C-COMMIT indication primtive + a C-BEGIN indication primitive with parameter values derived from the APDUs.

7.7.4.3 C-COMMIT response primitive

The accepting CCRPM forms a C-COMMIT-RC APDU from the parameter value of the C-COMMIT response primitive. It issues a P-SYNC-MAJOR response primitive with the APDU as a data value of the primitive's User Data parameter.

7.7.4.4 C-COMMIT-RC APDU

The requesting CCRPM receives a C-COMMIT-RC APDU from its peer as user data on a P-SYNC-MAJOR confirm primitive. It issues a C-COMMIT confirm primitive with the parameter value derived from the APDU.

7.7.5 Use of the C-COMMIT-RI APDU and C-BEGIN-RI APDU fields

The procedures of 7.4.5 are followed for the C-COMMIT-RI APDUs fields and of 7.1.5 for the C-BEGIN-RI APDU fields.

7.7.6 Use of the C-COMMIT-RC APDU field

The procedures of 7.4.6 are followed.

7.7.7 Collisions

A collision of a C-COMMIT-RI APDU + C-BEGIN-RI APDU with another CCR APDU cannot occur.

7.8 Rollback and begin branch procedure

7.8.1 Purpose

This procedure is used by a superior to request its subordinate to rollback one atomic action branch, while beginning a new atomic action branch between the two CCR-service users. It supports the C-ROLLBACK and C-BEGIN services defined in ISO/IEC 9804.

7.8.2 APDUs used

The procedure uses the CCR APDUs specified in 7.1.2 and 7.5.2.

7.8.3 Prerequisite requirements

The prerequisite requirements for this procedure are the same as those for the rollback procedure, specified in 7.5.3.

7.8.4 Procedure operation

This procedure is driven by the following events:

a) C-ROLLBACK request primitive + C-BEGIN request primitive from the requestor;

b) C-ROLLBACK-RI APDU + C-BEGIN-RI APDU received by the accepting CCRPM;

c) C-ROLLBACK response primitive from the acceptor; and

d) C-ROLLBACK-RC APDU received by the requesting CCRPM.

NOTE — The C-BEGIN response primitive and the C-BEGIN-RC APDU may optionally occur with c) and d) respectively.

7.8.4.1 C-ROLLBACK request primitive + C-BEGIN request primitive

The requesting CCRPM forms a C-ROLLBACK-RI APDU and a C-BEGIN-RI APDU from parameter values of the C-ROLLBACK request primitive and C-BEGIN request primitive, respectively. It issues a P-RESYNCHRONIZE(restart) request primitive with the APDUs as data values of the primitive's User Data parameter.

7.8.4.2 C-ROLLBACK-RI APDU + C-BEGIN-RI APDU

The accepting CCRPM receives a C-ROLLBACK-RI and a C-BEGIN-RI APDU from its peer as user data on a P-RESYNCHRONIZE(restart) indication primitive. It issues a C-ROLLBACK indication primtive + a C-BEGIN indication primitive with parameter values derived from the APDUs.

7.8.4.3 C-ROLLBACK response primitive

The accepting CCRPM forms a C-ROLLBACK-RC APDU from the parameter value of the C-ROLLBACK response primitive. It issues a P-RESYNCHRONIZE(restart) response primitive with the APDU as a data value of the primitive's User Data parameter.

7.8.4.4 C-ROLLBACK-RC APDU

The requesting CCRPM receives a C-ROLLBACK-RC APDU from its peer as user data on a P-RESYNCHRONIZE(restart) confirm primitive. It issues a C-ROLLBACK confirm primitive with the parameter value derived from the APDU.

7.8.5 Use of the C-ROLLBACK-RI APDU and C-BEGIN-RI APDU fields

The procedures of 7.5.5 are followed for the C-ROLLBACK-RI APDUs fields and of 7.1.5 for the C-BEGIN-RI APDU fields.

7.8.6 Use of the C-ROLLBACK-RC APDU field

The procedures of 7.5.6 are followed.

7.8.7 Disruptive effects

The rollback and begin branch procedure has the same disruptive effects as the rollback procedure, as described in 7.5.7.

7.8.8 Collisions

7.8.8.1 Collisions with CCR APDUs other than C-ROLLBACK-RI APDU are resolved in favour of the C-ROLLBACK-RI APDU + C-BEGIN-RI APDU.

7.8.8.2 Resolution of the collision of C-ROLLBACK-RI APDU + C-BEGIN-RI APDU with C-ROLLBACK-RI APDU [both mapped to P-RESYNCHRONIZE(restart)] depends on which side was the association-initiator. Figures 7 and 8 show the two cases - in both, CCRPM A's service-user initiates the rollback and begin branch procedure.

In figure 7, CCRPM A's service-user was the association-initiator. The C-ROLLBACK-RI APDU from CCRPM B is discarded (by the underlying session-service) and is not received by CCRPM A. The C-ROLLBACK-RI + C-BEGIN-RI APDU from CCRPM A is received by CCRPM B. As a result, the rollback procedure initiated by CCRPM B is disrupted. The rollback and begin branch procedure initiated by CCRPM A is processed normally.

In figure 8, CCRPM B's service-user was the association-initiator. The C-ROLLBACK-RI + C-BEGIN-RI APDU from CCRPM A is discarded (by the underlying session-service) and is not received by CCRPM B. The C-ROLLBACK-RI APDU is received by CCRPM A. As a result, the rollback and begin branch procedure initiated by CCRPM A is disrupted.

The rollback procedure initiated by CCRPM B is processed normally with addition that CCRPM A sends a copy of the discarded C-BEGIN-RI APDU concatenated with the C-ROLLBACK-RC APDU.

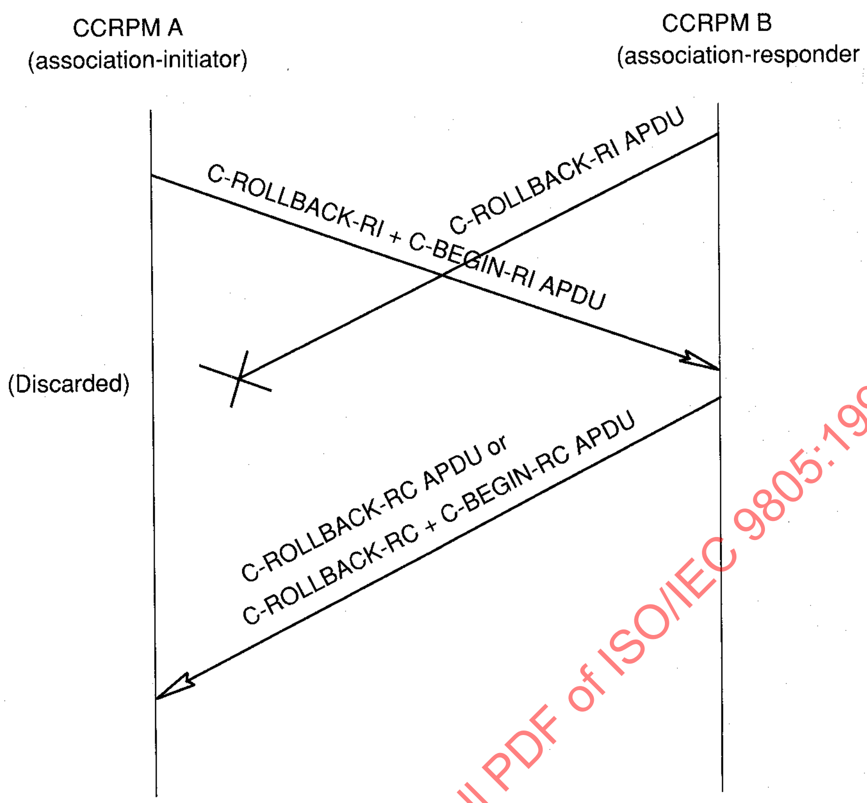

Figure 7 — CCRPM A is associaton-initiator

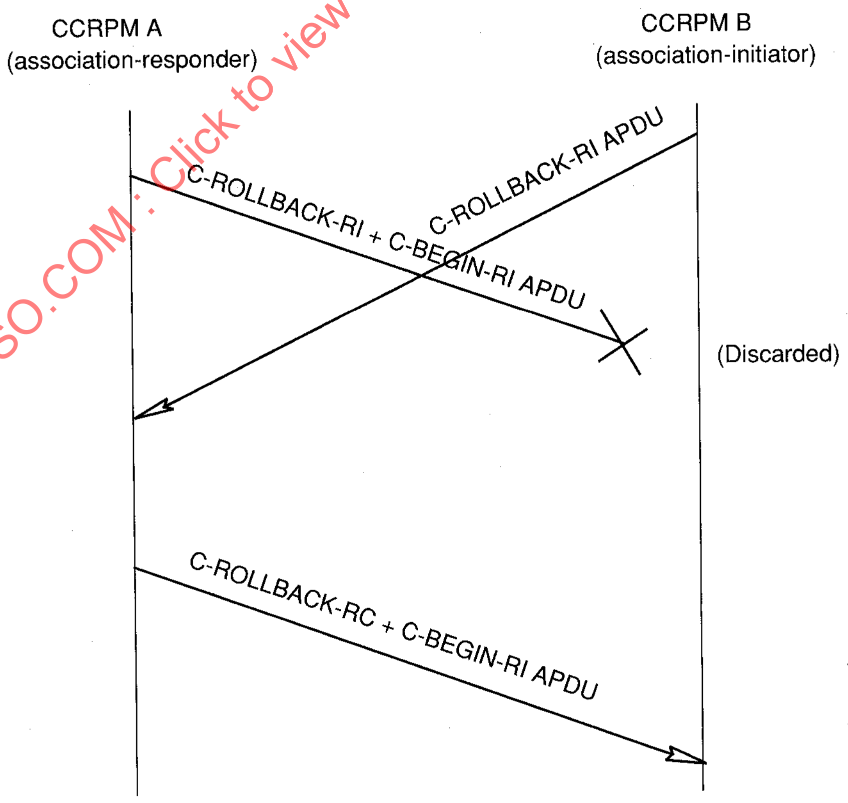

Figure 8 — CCRPM B is associaton-initiator

8 CCRPM State Table

This clause defines a single CCR Protocol Machine (CCRPM) in terms of a state table. The CCR State Table specifies the interrelationship between the current state of a CCRPM, the incoming events that occur, preconditions, enablements, the actions taken, outgoing events, and, finally, the resultant state of the CCRPM.

8.1 General

8.1.1 A CCRPM usually handles at most one atomic action branch at any one time. An overlap of two branches occurs only when a C-BEGIN request primitive is processed jointly with a C-COMMIT or C-ROLLBACK request primitive.

8.1.2 Tables 22 to 27 define the elements used in the State Table.

— Table 22 specifies the abbreviated name and description for each state.

— Table 23 specifies the abbreviated name, source, and description for each incoming event.

— Table 24 specifies the identifier and description for each specific action.

— Table 25 specifies the identifier and description for each precondition.

— Table 26 specifies the identifier and description for each enablement.

— Table 27 specifies the identifier and description for each outgoing event.

8.1.3 The overall CCR State Table is divided into individual tables (tables 28 to 31) for convenience and clarity. The individual state tables utilize the abbreviated names and identifiers of tables 22 to 27.

— Table 28 specifies the states and events that occur in a CCRPM that is being used by the superior for an atomic action branch, up to the completion of the branch or a failure, whichever happens first.

— Table 29 specifies the states and events that occur in a CCRPM that is being used by the subordinate up to completion or failure.

— Table 30 specifies the states and events that occur in a CCRPM that is used by the superior when recovery of a branch is attempted.

— Table 31 specifies the states and events that occur in a CCRPM that is used by the subordinate when recovery of a branch is attempted.

8.2 Incoming events

8.2.1 The types of incoming events specified in table 23 are:

a) the occurrence of a CCR service primitive request; or

b) the occurrence of a CCR service primitive response; or

c) the receipt of a CCR APDU as a presentation data value; or

d) the joint occurrence of two CCR service primitive requests; or

e) the receipt of two CCR APDUs as presentation data values on the same presentation primitive.

8.2.2 Clause 10 specifies the allowed sequences of concatenated CCR APDUs that may be sent in a single presentation primitive. The joint occurrence of allowed CCR service primitives or the receipt of allowed concatenated APDUs not shown as an incoming event in table 23 are treated as the consecutive occurrence of individual incoming events.

8.3 Outgoing events

The types of outgoing events specified in table 26 are:

a) the occurrence of a CCR service primitive indication; or

b) the occurrence of a CCR service primitive confirm; or

c) a CCR APDU as a presentation data value being sent; or

d) the joint occurrence of two CCR service primitive indications; or

e) the sending of two CCR APDUs as presentation data values on the same presentation primitive.

8.4 Specific actions

The specific actions specified in table 24 are performed internally by the CCRPM. They specify the values to be assigned to the variables specified in 8.7. The actions also state when the atomic action branch is completed.

8.5 Predicates

A predicate is a precondition that has either a "true" or "false" value. The CCRPM predicates specified in table 25 include the following:

a) whether or not atomic action data for a particular atomic action branch is accessible in stable storage;

b) the possession of the major/activity sync token;

c) the possession of the minor sync tokens;

d) whether a CCR request primitive is issued for the branch that is active on the association.

8.6 Enablements

The enablements permit changes to the accessibility of atomic action data in stable storage. An enablement does not require that changes be made. The enablements for the CCRPM (defined in table 26) are:

a) atomic action data for the branch can be made accessible in stable storage; or

b) atomic action data for the branch can cease to be accessible in stable storage.

The accessibility of atomic action data in stable storage controls some of the predicates defined in table 25.

8.7 Variables

Two variables are specified for the CCRPM:

a) Current-Branch; and

b) Next-Branch.

At any time, each variable contains either a value of "null" or a value that identifies a particular branch of an atomic action. This value consists of an atomic action identifier plus a branch identifier.

In this clause, the branch that is identified by the Current-Branch variable is called the current branch.

The Next-Branch variable is used to hold a value which can subsequently be assigned to the Current-Branch variable.

8.8 Notation

The following notation is used in the CCR State Table (tables 28 to 31).

— The states specified in table 22 are represented by the notation "Zn", where Z is an upper-case letter and n is null or an integer.

— Incoming events are represented by the names assigned in table 23.

— Specific actions are represented by the notation "[n]", where n is the action number assigned in table 24.

— Predicates are represented by the notation "pn" as assigned in table 25, where n is an integer.

— Enablements are represented by the notation "en" as assigned in table 26, where n is an integer.

— The outgoing events are represented by the identifier assigned for the event in table 27.

8.9 Conventions

8.9.1 In the CCR state tables, the intersection of an incoming event (row) and a state (column) forms a cell. A blank cell represents an event/state combination that is not defined for the CCRPM (see 8.10.2).

8.9.2 A non-blank cell represents an event/state combination that is defined for the CCRPM. Such a cell has an entry which contains the following:

a) a predicate expression (optional);

b) a specific action (optional);

c) an outgoing event; and

d) a resultant state.

8.9.3 If the intersection of an incoming event and the column "Preconditions" is blank, there is no precondition for the event. If a precondition is shown in the cell, the precondition applies to the incoming event.

8.9.4 If the intersection of the "Enablements" row and a state column is blank, no specific enablement exists for that state. If an enablement is shown, the enablement applies if the CCRPM is in that state.

8.10 Actions to be taken by the CCRPM

The CCR State Table defines the actions to be taken by the CCRPM.

8.10.1 General

8.10.1.1 The CCRPM is initialized in the idle state "I" and the variables are set to "null." This occurs when the CCRPM is initially used on the association.

8.10.1.2 The state of the CCRPM is only changed as specified in 8.10.2 and 8.10.3. When the association is normally or abnormally released, the CCRPM ceases to exist.

8.10.2 Invalid intersections

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken.

a) If the incoming event corresponds to the receipt of one or more CCR service primitives from the CCR service-user, any action taken by the CCRPM is a local matter. However, the CCRPM shall not send invalid protocol (i.e. one or more CCR APDUs) to its peer.

b) If the incoming event corresponds to receipt of one or more CCR APDUs from the peer CCRPM, the action, future state and subsequent outgoing events (if any outgoing events do occur) are not determined by this International Standard. However, as long as it persists, the CCRPM shall not send protocol (i.e. one or more CCR APDUs) to its peer.

8.10.3 Valid intersections

Non-blank cells indicate a valid intersection of an incoming event and state. If such an intersection occurs and

a) the predicate expression (if any) under the Predicate column for the row corresponding to the incoming event is true and

b) the predicate expression (if any) in the cell is true

the following actions are taken:

c) The CCRPM performs the specific action (if any) shown in the cell.

d) The specified output event is performed.

e) The state of the CCRPM is changed to the specified resultant state.

If one or both of the predicate expressions is false, the CCRPM follows the procedure for an invalid intersection, specified in 8.10.2.

8.11 Changes to atomic action data

8.11.1 Atomic action data for a particular atomic action branch is not made accessible in stable storage unless a CCRPM is in a state for which an enablement in table 26 permits the change.

8.11.2 Atomic action data for a particular branch of an atomic action remains accessible in stable storage unless:

a) a CCRPM is in a state for which the enablement permits the change; or

b) a CCRPM has performed the specific action in table 24 of determining that the atomic action branch is completed.

Table 22 - CCRPM States

Abbreviated name	Description
I	Idle
A1	C-BEGIN req received
A2	C-BEGIN-RC APDU received
A3	C-BEGIN req and C-PREPARE req received
A4	C-BEGIN-RC APDU received and C-PREPARE req received
A5	C-READY-RI received
A6	C-COMMIT req received
A7	C-READY-RI APDU not received and C-ROLLBACK req received
A8	C-READY-RI APDU received and C-ROLLBACK req received
A9	C-ROLLBACK-RI APDU received
A10	C-COMMIT req and C-BEGIN req received
A11	C-READY-RI APDU not received and C-ROLLBACK req and C-BEGIN req received
A13	C-READY-RI APDU received and C-ROLLBACK req and C-BEGIN req received
B1	C-BEGIN-RI APDU received
B2	C-BEGIN-RI APDU and C-BEGIN rsp received
B3	C-BEGIN-RI APDU and C-PREPARE-RI APDU received
B4	C-BEGIN rsp and C-PREPARE-RI APDU received
B5	C-READY req received, C-PREPARE-RI APDU not received
B6	C-READY req and C-PREPARE-RI APDU received
B7	C-COMMIT-RI APDU received
B8	C-ROLLBACK-RI APDU received
B9	C-ROLLBACK req received
B10	C-COMMIT-RI and C-BEGIN-RI APDUs received
B11	C-ROLLBACK-RI and C-BEGIN-RI APDUs received
X1	C-RECOVER (commit) req received
X2	C-RECOVER (ready)-RI APDU received
Y1	C-RECOVER (commit)-RI APDU received
Y2	C-RECOVER (READY) req received

Table 23 – Incoming event list

Abbreviated name	Source	Name and description
C-BEGIN req	CCR-user	C-BEGIN request primitive issued by requestor
C-BEGIN-RI	CCR-peer	C-BEGIN-RI APDU received by accepting CCRPM
C-BEGIN rsp	CCR-user	C-BEGIN response primitive issued by acceptor
C-BEGIN-RC	CCR-peer	C-BEGIN-RC APDU received by requesting CCRPM
C-PREPARE req	CCR-user	C-PREPARE request primitive issued by requestor
C-PREPARE-RI	CCR-peer	C-PREPARE-RI APDU received by accepting CCRPM
C-READY req	CCR-user	C-READY request primitive issued by requestor
C-READY-RI	CCR-peer	C-READY-RI APDU received by accepting CCRPM
C-COMMIT req	CCR-user	C-COMMIT request primitive issued by requestor
C-COMMIT-RI	CCR-peer	C-COMMIT-RI APDU received by accepting CCRPM
C-COMMIT rsp	CCR-user	C-COMMIT response primitive issued by acceptor
C-COMIT-RC	CCR-peer	C-COMMIT-RC APDU received by requesting CCRPM
C-ROLLBACK req	CCR-user	C-ROLLBACK request primitive issued by requestor
C-ROLLBACK-RI	CCR-peer	C-ROLLBACK-RI APDU received by accepting CCRPM
C-ROLLBACK rsp	CCR-user	C-ROLLBACK response primitive issued by acceptor
C-ROLLBACK-RC	CCR-peer	C-ROLLBACK-RC APDU received by requesting CCRPM
C-COMMIT + C-BEGIN req	CCR-user	C-COMMIT request primitive with a C-BEGIN request primitive
C-COMMIT-RI + C-BEGIN-RI	CCR-peer	C-COMMIT-RI+ C-BEGIN-RI concatenated APDU received by accepting CCRPM
C-ROLLBACK + C-BEGIN req	CCR-user	C-ROLLBACK request primitive with a C-BEGIN request primitive issued by requestor
C-ROLLBACK-RI + C-BEGIN-RI	CCR-peer	C-ROLLBACK-RI + C-BEGIN-RI concatenated APDU received by accepting CCRPM
C-RECOVER(commit) req	CCR-user	C-RECOVER(commit)request primitive issued by requestor (Recovery State = "commit")
C-RECOVER-RI(commit)	CCR-peer	C-RECOVER-RI(commit) APDU received by accepting CCRPM (recovery-state = "commit")
C-RECOVER(ready) req	CCR-user	C-RECOVER(ready) request primitive issued by requestor (Recovery State = "ready")
C-RECOVER-RI(ready)	CCR-peer	C-RECOVER-RI(ready) APDU received by accepting CCRPM (recovery-state = "ready")
C-RECOVER(done) rsp	CCR-user	C-RECOVER(done) response primitive issued by acceptor (Recovery State = "done")
C-RECOVER-RC(done)	CCR-peer	C-RECOVER-RC(done) APDU received by requesting CCRPM (recovery-state = "done")
C-RECOVER(retry-later) rsp	CCR-user	C-RECOVER(retry-later) response primitive issued by acceptor (Recovery State = "retry-later")
C-RECOVER-RC(retry-later)	CCR-peer	C-RECOVER-RC(retry-later) APDU received by requesting CCRPM (recovery-state = "retry-later")

Table 23 - Incoming event list (concluded)

Abbreviated name	Source	Name and description
C-RECOVER(unknown) rsp	CCR-user	C-RECOVER(unknown) response primitive issued by acceptor (Recovery State = "unknown")
C-RECOVER-RC(unknown)	CCR-peer	C-RECOVER-RC(unknown) APDU received by requesting CCRPM (recovery-state = "unknown")

Table 24 – Actions

Action	Description
1	Current-Branch variable is set to the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-BEGIN request.
2	The current branch is completed. Current-Branch variable is set to "null".
3	Next-Branch variable is set to the atomic action branch identified by the Atomic Action identifier and Branch Identifier on the C-BEGIN request. Save the parameter values of the C-BEGIN request.
4	The current branch is completed. Current-Branch variable is set to the value of Next-Branch variable. Next-Branch variable is set to "null".
5	Current-Branch variable is set to the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-BEGIN-RI APDU.
6	Next-Branch variable is set to the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-BEGIN-RI APDU.
7	Current-Branch variable is set to the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-RECOVER request.
8	Current-Branch variable is set to the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-RECOVER-RI APDU.
9	Current-Branch variable is set to "null".

Table 25– Predicates

Predicate	Description
p1	Atomic action data for the superior of the current branch is accessible in stable storage. The major/activity token is in the possession of this requestor.
p2	Either no atomic action data for the superior of the current branch is accessible in stable storage, or the CCR service-user using this association has been ordered to rollback by its superior.
p3	Atomic action data for the subordinate of the current branch is accessible in stable storage.
p4	No atomic action data for the subordinate of the current branch is accessible in stable storage.
p5	The atomic action data for the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-RECOVER (COMMIT) request is accessible in stable storage; and the minor sync token is in the possession of the requestor.
p6	The Current Branch variable is the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on C-RECOVER(COMMIT) request; and atomic action data for the superior of the current branch is accessible in stable storage.
p7	The minor sync token is in the possession of the requestor.

Table 26 – Enablements

Enablement	Description
e1	Atomic action data for the superior of the current branch can be made accessible in stable storage.
e2	Atomic action data for the subordinate of the current branch can be made accessible in stable storage.
e3	Atomic action data for the subordinate of the current branch can cease to be accessible in stable storage.

Table 27 – Outgoing event list

Code	Description
pa	Send C-BEGIN-RI APDU
pb	Send C-BEGIN-RC APDU
pc	Send C-PREPARE-RI APDU
pd	Send C-READY-RI APDU
pe	Send C-COMMIT-RI APDU
pf	Send C-COMMIT-RC APDU
pg	Send C-ROLLBACK-RI APDU
ph	Send C-ROLLBACK-RC APDU
pi	Send C-RECOVER(commit)-RI APDU
pj	Send C-RECOVER(done)-RC APDU
pk	Send C-RECOVER(ready)-RI APDU
pl	Send C-RECOVER(unknown)-RC APDU
pm	Send C-RECOVER(retry-later)-RC APDU
pea	Send C-COMMIT-RI APDU and C-BEGIN-RI APDU on the same presentation primitive
pga	Send C-ROLLBACK-RI APDU and C-BEGIN-RI APDU on the same presentation primitive
pha	Send C-ROLLBACK-RC APDU, and resend the discarded C-BEGIN-RI APDU on the same presentation primitive
sa	Issue C-BEGIN ind
sb	Issue C-BEGIN cnf
sc	Issue C-PREPARE ind
sd	Issue C-READY ind
se	Issue C-COMMIT ind
sf	Issue C-COMMIT cnf
sg	Issue C-ROLLBACK ind
sh	Issue C-ROLLBACK cnf
si	Issue C-RECOVER(commit) ind
sj	Issue C-RECOVER(done) cnf
sk	Issue C-RECOVER(ready) ind
sl	Issue C-RECOVER(unknown) cnf
sm	Issue C-RECOVER(retry-later) cnf
sea	Issue C-COMMIT ind and C-BEGIN ind
sga	Issue C-ROLLBACK ind and C-BEGIN ind

Table 28 – State table for superior: normal

Incoming Event	Precondition	Preceding State													
		I	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
C-BEGIN req	p7	[1] pa A1													
C-BEGIN-RC			sb A2		sb A4										
C-PREPARE req			pc A3	pc A4											
C-READY-RI			sd A5	sd A5	sd A5	sd A5									
C-COMMIT req	p1						pe A6								
C-COMMIT-RC								[2] sf I				[4] sf A1			
C-ROLLBACK req	p2		pg A7	pg A7	pg A7	pg A7	pg A8								
C-ROLLBACK-RC									[2] sh I	[2] sh I			[4] sh A1		[4] sh A1
C-ROLLBACK-RI			sg A9	sg A9	sg A9	sg A9			sg A9				sg A12		
C-ROLLBACK rsp											[2] ph I			[4] pha A1	
C-COMMIT req + C-BEGIN req	p1						[3] pea A10								
C-ROLLBACK + C-BEGIN req	p2		[3] pga A11	[3] pga A11	[3] pga A11	[3] pga A11	[3] pga A13								
Enablement							e1								

Table 29– State table for subordinate: normal

Incoming Event	Precondition	Preceding State											
		I	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
C-BEGIN RI		[5] sa B1											
C-BEGIN rsp			pb B2		pb B4								
C-PREPARE-RI			sc B3	sc B4			sc B6						
C-READY req	p3		pd B5	pd B5	pd B6	pd B6							
C-COMMIT-RI							se B7	se B7					
C-COMMIT rsp	p4								[2] pf I			[4] pf B1	
C-ROLLBACK-RI			sg B8	sg B8	sg B8	sg B8	sg B8	sg B8			sg B8		
C-ROLLBACK rsp	p4									[2] ph I			[4] ph B1
C-ROLLBACK req	p4		pg B9	pg B9	pg B9	pg B9							
C-ROLLBACK-RC											sh I		
C-COMMIT-RI + C-BEGIN-RI	p1						[5] sea B10	[5] sea B10					
C-ROLLBACK-RI + C-BEGIN-RI			[6] sga B11	[6] sga B11	[6] sga B11	[6] sga B11	[6] sga B11	[6] sga B11			[6] sga B11		
Enablement			e2	e2	e2	e2			e3	e3	e3	e3	e3

Table 30 - State table for superior: recovery

Incoming Event	Precondition	Preceding State		
		I	X1	X2
C-RECOVER(commit) req		p5 [7] pi X1		p6 pi X1
C-RECOVER(done)-RC			[2] sj I	
C-RECOVER(retry-later)-RC			sm I	
C-RECOVER(ready)-RI		[8] sk X2		
C-RECOVER(retry-later) rsp				pm I
C-RECOVER(unknown) rsp	p2			[9] pl I
Enablement				

Table 31 - State table for subordinate: recovery

Incoming event	Precondition	Preceding state		
		I	Y1	Y2
C-RECOVER(commit)-RI		[8] si Y1		si Y1
C-RECOVER(done) rsp	p4		[2] pj I	
C-RECOVER(retry-later) rsp			pm I	
C-RECOVER(ready) req	p3 & p7	[7] pk Y2		
C-RECOVER(retry-later)-RC				sm I
C-RECOVER(unknown)-RC				[2] sl I
Enablement			e3	

9 Mapping to the presentation service

Clauses 7 and 8 specify the behaviour of a CCRPM in relation to CCR input events. Some events result in sending or receiving one or more (concatenated) CCR APDUs. This clause specifies how the presentation-service primitives are used by the CCRPM. Table 32 summarizes the mapping of CCR primitives and their related APDUs to the presentation primitives used.

9.1 Begin branch

The begin branch procedure uses the P-SYNC-MINOR service. For the C-BEGIN response primitive, the begin branch procedure can also use the P-TYPED-DATA service.

9.1.1 Use of the P-SYNC-MINOR req/ind parameters

9.1.1.1 Type: This mandatory parameter is set to the value of "optional."

9.1.1.2 Synchronization Point Serial Number: The value is kept by the CCRPM but its use is not otherwise determined by this International Standard.

9.1.1.3 User data: The User Data parameter is used to carry the C-BEGIN-RI APDU. User Data (if any) on the C-BEGIN request primitive is included in the C-BEGIN-RI APDU and is expressed using one or more presentation contexts specified by the requestor on the C-BEGIN request primitive.

Table 32 - Mapping overview

CCR primitive	CCR APDU	Presentation primitive
C-BEGIN req/ind	C-BEGIN-RI	P-SYNC-MINOR req/ind
C-BEGIN rsp/cnf	C-BEGIN-RC	P-SYNC-MINOR rsp/cnf
C-BEGIN rsp/cnf where C-BEGIN req was given with C-ROLLBACK req or C-COMMIT req	C-BEGIN-RC	P-TYPED-DATA req/ind
C-PREPARE req/ind	C-PREPARE-RI	P-TYPED-DATA req/ind
C-READY req/ind	C-READY-RI	P-TYPED-DATA req/ind
C-ROLLBACK req/ind	C-ROLLBACK-RI	P-RESYNC(restart) req/ind
C-ROLLBACK rsp/cnf	C-ROLLBACK-RC	P-RESYNC(restart) rsp/cnf
C-ROLLBACK req/ind + C-BEGIN req/ind	C-ROLLBACK-RI followed by C-BEGIN-RI	P-RESYNC(restart) req/ind
C-ROLLBACK rsp/cnf + C-BEGIN rsp/cnf	C-ROLLBACK-RC followed by C-BEGIN-RC	P-RESYNC(restart) rsp/cnf
(see 7.8.8.1)	C-ROLLBACK-RC followed by C-BEGIN-RI	P-RESYNC(restart) rsp/cnf
C-COMMIT req/ind	C-COMMIT-RI	P-SYNC-MAJOR req/ind
C-COMMIT rsp/cnf	C-COMMIT-RC	P-SYNC-MAJOR rsp/cnf
C-COMMIT req/ind + C-BEGIN req/ind	C-COMMIT-RI followed by C-BEGIN-RI	P-SYNC-MAJOR req/ind
C-COMMIT rsp/cnf + C-BEGIN rsp/cnf	C-COMMIT-RC followed by C-BEGIN-RC	P-SYNC-MAJOR rsp/cnf
C-RECOVER req/ind	C-RECOVER-RI	P-TYPED-DATA req/ind
C-RECOVER rsp/cnf	C-RECOVER-RC	P-TYPED DATA req/ind