INTERNATIONAL STANDARD

ISO/IEC 9805-1

Third edition
1998-12-15

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

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

Reference number
ISO/IEC 9805-1:1998(E)

Contents

Page

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 9805-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 33, *Distributed application services*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.852.

This third edition cancels and replaces the second edition (ISO/IEC 9805-1:1994), which has been technically revised.

ISO/IEC 9805 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — Protocol for the Commitment, Concurrency and Recovery service element*:

— *Part 1: Protocol specification*

— *Part 2: Protocol Implementation Conformance Statement (PICS) proforma*

Annexes A and B form an integral part of this part of ISO/IEC 9805.

Introduction

This Recommendation | International Standard is one of a set of Recommendations | International Standards produced to facilitate the interconnection of information processing systems. It is related to other Recommendation | International Standards in the set as defined by the Reference Model for Open Systems Interconnection (see ITU-T Rec. X.200 | ISO/IEC 7498-1). 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 Recommendation | 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.

This Recommendation | International Standard specifies CCR Protocol Version 2. The 1990 edition of ISO/IEC 9805 specified CCR protocol Version 1. The 1993/1994 editions of this Recommendation | International Standard specified both CCR protocol Version 1 and CCR protocol Version 2, for the static commitment functional unit only.

The CCR protocol specification consists of the following main components:

a) the specification of the CCR APDUs using Abstract Syntax One (ASN.1, ITU-T Rec. X.680 | ISO/IEC 8824-1);

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 (see ITU-T Rec. X.216 | ISO/IEC 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 Recommendation | International Standard.

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

Annex B describes how the CCR can be used in combination with other ASEs that use the presentation service in a way that is not compatible with the use of the presentation service by CCR as specified in the body of this Recommendation | International Standard.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – PROTOCOL FOR THE COMMITMENT, CONCURRENCY AND RECOVERY SERVICE ELEMENT: PROTOCOL SPECIFICATION

1 Scope

This Recommendation | 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 ITU-T Rec. X.851 | ISO/IEC 9804. A reference to a CCR service invokes the procedures of this Recommendation | International Standard to cause external effects.

This Recommendation | International Standard specifies, in clause 9, a use of the ACSE, Presentation and Session services to carry the CCR semantics. This "reference" mapping can be used whenever the use of these services does not conflict with the user made by other ASEs or ASOs that are using the same association. Annex B defines how a different use of supporting services may be specified, for use where the reference mapping is inappropriate.

This Recommendation | 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.

This edition of this Recommendation | International Standard specifies CCR protocol Version 2, which makes use of the Session Data Separation functional unit to protect data not belonging to the CCR atomic action. The 1990 edition of ISO/IEC 9805 specified CCR protocol Version 1. The 1993/1994 edition of this Recommendation | International Standard specified both CCR protocol Version 1 and CCR protocol Version 2, for the static commitment functional unit only.

This International Standard specifies the protocol elements that support the following functional units:

a) static commitment;

b) dynamic commitment;

c) read only;

d) one-phase commitment;

e) cancel; and

f) overlapped recovery.

2 Normative references

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

2.1 Identical Recommendations | International Standards

– ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*

– ITU-T Recommendation X.207 (1993) | ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application layer structure.*

– ITU-T Recommendation X.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model : Conventions for the definition of OSI services.*

- ITU-T Recommendation X.215 (1995) | ISO/IEC 8326:1996, *Information technology – Open Systems Interconnection – Session service definition.*
- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition.*
- ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the association control service element.*
- ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the association control service element: Protocol specification.*
- ITU-T Recommendation X.650 (1996) | ISO/IEC 7498-3:1997, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing.*
- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.680 (1994)/Amd. 1 (1995) | ISO/IEC 8824-1:1995/Amd. 1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation – Amendment 1: Rules of extensibility.*
- Technical Corrigendum (1997) to ITU-T Recommendation X.680 (1994)/Amd. 1 (1995) | ISO/IEC 8824-1:1995/Amd. 1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation – Amendment 1: Rules of extensibility.*
- ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*
- ITU-T Recommendation X.851 (1997) | ISO/IEC 9804:1998, *Information technology – Open Systems Interconnection – Service definition for the commitment, concurrency and recovery service element.*

3 Definitions

3.1 Reference model definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.200 | ISO/IEC 7498-1:

a) Application Layer;

b) application association; association;

c) application-process;

d) application-entity;

e) presentation-service;

f) presentation-connection;

g) session-service; and

h) session-connection.

3.2 Naming and addressing definitions

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

a) application-process title;

b) application-entity qualifier;

c) application-entity title.

3.3 Service conventions definitions

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

a) service-provider;

b) service-user;

c) confirmed service;

d) non-confirmed service;

e) primitive;

f) request (primitive);

g) indication (primitive);

h) response (primitive); and

i) confirm (primitive).

3.4 Presentation service definitions

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

a) abstract syntax;

b) abstract syntax name;

c) defined context set;

d) presentation context; and

e) presentation data value.

3.5 ACSE service definitions

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

a) association-initiator; and

b) association-responder.

3.6 Application Layer Structure definitions

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

a) application-entity-invocation;

b) application-service-element;

3.7 CCR service definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.851 | ISO/IEC 9804:

1) acceptor;
2) application failure;
3) atomic action;
4) atomic action branch identifier; branch identifier;
5) atomic action branch; branch;
6) atomic action data;
7) atomic action graph;
8) atomic action identifier;

9) atomic action initiator;
10) atomic action owner;
11) atomicity;
12) begin-tree; atomic action begin-tree
13) bound data;
14) branch-initiator; atomic action branch-initiator;
15) branch-responder; atomic action branch-responder;
16) CCR service-provider;
17) CCR service-user;
18) commit coordinator;
19) commit-decider;
20) commit-subordinate;
21) commit-superior;
22) commit-tree; atomic action commit-tree;
23) communication failure;
24) compensating action;
25) concurrency control;
26) confirmation of commitment;
27) connected graph;
28) consistency;
29) continuing two-phase branch; continuing two-phase neighbour;
30) distributed application;
31) doubt period;
32) durability;
33) final state;
34) graph;
35) heuristic decision;
36) initial state;
37) intermediate;
38) intermediate state;
39) interrupted branch;
40) isolation;
41) leaf;
42) local commitment procedures;
43) local rollback procedures;
44) mixed heuristic situation; mixed situation;
45) neighbour (of a node in a graph);
46) neighbourhood (of a node);
47) node; CCR node;
48) non-reference mapping;
49) order of commitment of an atomic action branch; order of commitment;
50) phase I;
51) phase II;
52) presumed-rollback;
53) ready signal;
54) ready-to-commit state;
55) recovery of an atomic action branch; recovery;
56) recovery responsibility for an atomic action branch; recovery responsibility;
57) reference mapping;
58) referencing specification;
59) requesting neighbourhood;
60) requestor;
61) rollback of an atomic action branch; rollback;
62) root;
63) subordinate of a node;
64) subordinate of an arc;
65) subordinate subtree;
66) subtree (of a node);
67) superior of a node (in a rooted tree);
68) superior of an arc (in a rooted tree);
69) tree;
70) user-ASE.

3.8 CCR protocol specification definitions

For the purposes of this Recommendation | 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 Recommendation | 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 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 Protocol Specification:

C-INITIALIZE-RI

C-INITIALIZE-RC

C-BEGIN-RI

C-BEGIN-RC

C-PREPARE-RI

C-READY-RI

C-COMMIT-RI

C-COMMIT-RC

C-ROLLBACK-RI

C-ROLLBACK-RC

C-CANCEL-RI

C-NOCHANGE-RI

C-NOCHANGE-RC

C-RECOVER-RI

C-RECOVER-RC

4.3 Other abbreviations

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

ACSE Association Control Service Element

AE application-entity

AEI application-entity invocation

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 Protocol Specification 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 (see ITU-T Rec. X.680 | ISO/IEC 8824-1).

5.3 CCR allows the concatenation of some of its APDUs. In clause 11 the allowed concatenations are specified.

6 Overview of the CCR protocol

6.1 Service support

The protocol specified in this Recommendation | International Standard supports the services defined in ITU-T Rec. X.851 | ISO/IEC 9804. These services are listed in Table 1.

Table 1 – CCR services

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

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 (see ITU-T Rec. X.217 | ISO/IEC 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 minor synchronize functional unit;
- Session resynchronize functional unit;
- Session data separation 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.

6.2.4 The ACSE User information on an A-ASSOCIATE request shall contain the C-INITIALIZE-RI APDU. The ACSE User information on an A-ASSOCIATE response shall contain the C-INITIALIZE-RC APDU.

6.3 Use of the presentation service

6.3.1 CCR uses the following presentation (see ITU-T Rec. X.216 | ISO/IEC 8822) services:

- P-DATA;
- P-TYPED-DATA;
- P-SYNC-MINOR;
- P-RESYNCHRONIZE(abandon).

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 shall 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 Protocol Specification 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 CCR requires use of session unlimited user data (see ITU-T Rec. X.215 | ISO/IEC 8326).

6.5 Operation of the CCRPM

6.5.1 The protocol specification for CCR is presented in this Recommendation | 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 cooperation 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 ITU-T Rec. X.851 | 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 11.

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 (see ITU-T Rec. X.680 | ISO/IEC 8824-1).

6.6 Rules of extensibility

For the C-INITIALIZE-RI APDU, a receiving CCRPM shall

a) ignore any undefined element;

b) where named bits are used, treat any bit as insignificant when no name is assigned to it.

7 Elements of procedures

The CCR protocol consists of the following procedures:

a) initialization;

b) begin branch;

c) prepare ;

d) signal readiness;

e) order commitment;

f) rollback;

g) cancel;

h) nochange completion;

i) branch recovery;

j) order commitment and begin new branch;

k) rollback and begin new branch; and

l) error.

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 11 applies.

Figures 1 to 9 show the ASN.1 structure of the CCR APDUs. The complete ASN.1 module, containing these definition and those of the supporting datatypes, is in Annex A.

7.1 Initialization procedure

7.1.1 Purpose

This procedure is used to negotiate the CCR version and which CCR functional units are available on the association. It supports the C-INITIALIZE service defined in X.851 | ISO/IEC 9804.

7.1.2 APDUs used

This procedure uses the following CCR APDUs:

C-INITIALIZE-RI

C-INITIALIZE-RC

The structure of these APDUs is shown in Figure 1.

```
C-INITIALIZE-RI  ::= [11]          SEQUENCE
      { version-number             [0] BIT STRING
                                       { version1(0), version2(1) } DEFAULT { version2 },
       ...,
      ccr-requirements             [1] Ccr-requirements DEFAULT { static-commitment},
       ready-collision-reservation [2] BOOLEAN DEFAULT TRUE,
      ...,
       user-data                       User-data OPTIONAL
      }
C-INITIALIZE-RC  ::= [12]          SEQUENCE
      { version-number             [0] BIT STRING
                                       { version1(0), version2(1) } DEFAULT { version2 },
      ...,
       ccr-requirements            [1] Ccr-requirements DEFAULT { static-commitment},
       ready-collision-reservation [2] BOOLEAN DEFAULT TRUE,
      ...,
       user-data                       User-data OPTIONAL
      }
```

Figure 1 – C-INITIALIZE APDUs

The C-INITIALIZE-RI APDU fields are listed in Table 2. The C-INITIALIZE-RC APDU fields are listed in Table 3.

Table 2 – C-INITIALIZE-RI field

Field name	Presence	Source	Sink
version-number	M	CCRPM	CCRPM
ccr-requirements	M	req	ind
ready-collision-reservation	U	req	ind
user-data	U	req	ind

Table 3 – C-INITIALIZE-RC field

Field name	Presence	Source	Sink
version-number	M	CCRPM	CCRPM
ccr-requirements	M	req	ind
ready-collision-reservation	U	req	ind
user-data	U	req	ind

7.1.3 Procedure operation

The procedure is performed concurrently with the A-ASSOCIATE procedure (see ITU-T Rec. X.217 | ISO/IEC 8649) when the association will be used for CCR. The procedure is driven by the following events:

a) C-INITIALIZE request primitive from the requestor;

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

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

d) C-INITIALIZE-RC APDU received by the CCRPM of the requestor.

7.1.3.1 C-INITIALIZE request primitive

The requesting CCRPM forms a C-INITIALIZE-RI APDU from parameter values of the C-INITIALIZE request primitive. The C-INITIALIZE APDU is a data value of the A-ASSOCIATE request User Information parameter.

7.1.3.2 C-INITIALIZE-RI APDU

The accepting CCRPM receives a C-INITIALIZE-RI APDU from its peer as user information on an A-ASSOCIATE indication primitive. It issues a C-INITIALIZE indication primitive with parameter values derived from the APDU.

7.1.3.3 C-INITIALIZE response primitive

The accepting CCRPM forms a C-INITIALIZE-RC APDU from the parameter value of the C-INITIALIZE response primitive. The APDU is a data value of the A-ASSOCIATE response User Information parameter.

7.1.3.4 C-INITIALIZE-RC APDU

The requesting CCRPM receives a C-INITIALIZE-RC APDU from its peer as user information on an A-ASSOCIATE confirm primitive.

7.1.4 Use of the C-INITIALIZE-RI APDU fields

The fields of the C-INITIALIZE-RI APDU fields are used by the requesting and accepting CCRPM as specified below.

7.1.4.1 Versions

For the requesting CCRPM: The value assigned to this field is determined within the implementation of the CCRPM. It is a bit string where each bit that is set to one indicates a version of CCR protocol that this CCRPM supports. The ASN.1 definition of the “Versions” datatype in Annex A specifies which bit represents which version. Multiple bits may be set, indicating support of multiple versions.

The version of the protocol defined in this Protocol Specification shall be Version 2.

NOTE – An implementation that can support multiple versions is not required to indicate support for all of these on every C-INITIALIZE-RI APDU. In a particular instance, it may choose to offer fewer versions than it is capable of.

For the accepting CCRPM: The CCRPM sets the Versions parameter of the C-INITIALIZE indication primitive to indicate the version of the CCR protocol selected for use on the association. This shall be the highest protocol version which is supported by this CCRPM and for which the corresponding bit in the “versions” field was set. Bits representing protocol versions that are not supported by the CCRPM are ignored.

7.1.4.2 CCR requirements

For the requesting CCRPM: This value is directly mapped from the CCR Requirements parameter on the C-INITIALIZE request primitive.

For the accepting CCRPM: The CCR Requirements parameter of the C-INITIALIZE indication primitive shall be set from the ccr-requirements field of the C-INITIALIZE-RI APDU. Only those CCR functional units that are both proposed in the ccr-requirements field of the APDU and are supported by the responding CCRPM shall be appear in the CCR Requirements parameter of the indication.

7.1.4.3 Ready-collision-reservation

For the requesting CCRPM: This value is directly mapped from the Ready-Collision-Reservation parameter on the C-INITIALIZE request primitive.

For the accepting CCRPM: This value is used to determine the value of the Ready-Collision-Reservation parameter of the C-INITIALIZE indication primitive.

7.1.4.4 User data

For the requesting CCRPM: This value is directly mapped from the User Data parameter on the C-INITIALIZE request primitive.

For the accepting CCRPM: This value is used to determine the value of the User Data parameter of the C-INITIALIZE indication primitive.

7.1.5 Use of the C-INITIALIZE-RC APDU field

The C-INITIALIZE-RC APDU fields are used by the accepting and requesting CCRPMs as specified below.

7.1.5.1 Versions

For the accepting CCRPM: The "versions" field shall be a bit string that indicates the version of the CCR protocol that has been selected for use on the association as specified in 7.1.6.1. The bit specified in the ASN.1 definition of the "Versions" datatype as representing the selected version, shall be set to one. Any leading bits, representing lower versions, shall be zero. There shall be no trailing zero bits.

For the requesting CCRPM: The version represented by the only set bit (the last) in the "versions" field is the version of the CCR protocol selected for use on the association. This version number is the value of the Version parameter of the C-INITIALIZE confirm primitive.

7.1.5.2 CCR requirements

For the accepting CCRPM: This value is directly mapped from the CCR Requirements parameter on the C-INITIALIZE response primitive. Only functional units that were proposed on the C-INITIALIZE indication shall be included on the response.

For the requesting CCRPM: This value is used to determine the value of the CCR Requirements parameter of the C-INITIALIZE confirm.

7.1.5.3 User data

For the accepting CCRPM: This value is directly mapped from the User Data parameter on the C-INITIALIZE indication primitive.

For the requesting CCRPM: This value is used to determine the value of the User Data parameter of the C-INITIALIZE confirm.

7.1.6 Collisions

None.

7.2 Begin branch procedure

7.2.1 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 ITU-T Rec. X.851 | ISO/IEC 9804.

7.2.2 APDUs used

The procedure uses the following CCR APDUs:

C-BEGIN-RI

C-BEGIN-RC

The structure of these APDUs is shown in Figure 2.

```
C-BEGIN-RI ::= [1]                    SEQUENCE
    { atomic-action-identifier        [0] ATOMIC-ACTION-IDENTIFIER,
      branch-suffix      CHOICE       {
                                      [2] OCTET STRING,
                                      [3] INTEGER,
                                      ...
                                      },
    ...,...,
      user-data                       User-data          OPTIONAL
    }

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

Figure 2 – C-BEGIN APDUs

The C-BEGIN-RI APDU fields are listed in Table 4. The C-BEGIN-RC APDU field is listed in Table 5.

Table 4 – C-BEGIN-RI field

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

Table 5 – C-BEGIN-RC field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.2.3 Prerequisite requirements

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

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

7.2.4 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.

7.2.4.1 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 11, with the C-BEGIN-RI APDU as a data value in the user data parameter.

7.2.4.2 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 11. In either case, the CCRPM issues a C-BEGIN indication primitive with parameter values derived from the APDU.

7.2.4.3 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 11, with the C-BEGIN-RC APDU as a data value in the user data parameter.

7.2.4.4 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 11. In either case, the CCRPM issues a C-BEGIN confirm primitive with the parameter value derived from the APDU.

7.2.5 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 6.

The CCRPM shall represent the "Atomic Action Identifier – Owner's Name" parameter of the C-BEGIN request in the abstract syntax by using either the "name" form or the "sender" value of the "side" form of the "owners-name" field. The latter form may only be used if the Owner's Name is the AE-title of the requestor, as passed on the A-ASSOCIATE service used to establish the supporting association.

The C-BEGIN request includes the Branch Identifier – Initiator'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 6. If the "owners-name" field in the "atomic-action-identifier" field is the "sender" value of the "side" form, the "Atomic Action Identifier – Owner's Name" parameter of the C-BEGIN indication shall be the requestor's AE-title that was passed in the A-ASSOCIATE service used to establish the supporting association.

The accepting CCRPM also includes the Branch Identifier – Initiator'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.

Table 6 – Mapping of C-BEGIN req/ind parameters

APDU Field name	Parameter name
atomic-action-identifier {owners-name}	Atomic Action Identifier – Owner's Name
atomic-action-identifier {atomic-action-suffix}	Atomic Action Identifier – Suffix
–	Branch Identifier – Branch-initiator'sName
branch-suffix	Branch Identifier – Suffix
user-data	User Data

7.2.6 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 7.

Table 7 – Mapping of C-BEGIN rsp/cnf parameter

APDU Field name	Parameter name
user-data	User Data

7.2.7 Collisions

A collision of a C-BEGIN-RI APDU with another CCR APDU cannot occur unless the overlapped recovery functional unit is selected. The specification of the CCR service-user shall ensure that C-BEGIN request and C-RECOVER request shall not collide.

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). The requirement, when overlapped recovery is not selected, 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. When overlapped recovery is selected, there is no protection *in CCR* against collision between C-BEGIN-RI and C-RECOVER-RI APDUs.

7.3 Prepare procedure

7.3.1 Purpose

The prepare procedure is used by CCR service-user to inform its neighbour that no more application semantics that affect bound data will be sent. The prepare procedure supports the C-PREPARE service defined in ITU-T Rec. X.851 | ISO/IEC 9804.

7.3.2 APDU used

The procedure uses the following CCR APDU:

C-PREPARE-RI

The structure of this APDU is shown in Figure 3.

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

Figure 3 – C-PREPARE APDU

The C-PREPARE-RI APDU field is listed in Table 8.

Table 8 – C-PREPARE-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.3.3 Prerequisite requirements

None.

7.3.4 Prepare procedure

This procedure is driven by the following events:

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

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

7.3.4.1 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 neither concatenated with other CCR APDUs, nor with APDUs from other ASEs, 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 C-PREPARE-RI APDU is concatenated with another CCR APDU or with an APDU from another ASE, the appropriate presentation service primitive is issued as specified in clause 11, with the C-PREPARE-RI APDU as a data value in the user data parameter.

7.3.4.2 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 or with APDUs from other ASEs, the C-PREPARE-RI APDU will be received as user data on the appropriate presentation primitive as specified in clause 11. In either case, the CCRPM issues a C-PREPARE indication primitive with the parameter value derived from the APDU.

7.3.5 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 9.

Table 9 – Mapping of C-PREPARE req/ind parameter

APDU Field name	Parameter name
user-data	User Data

7.3.6 Collisions

7.3.6.1 The prepare procedure and the signal readiness procedure may be used simultaneously by both CCR service-users. 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.

7.3.6.2 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.

7.3.6.3 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.

7.4 Signal readiness procedure

7.4.1 Purpose

The signal readiness procedure is used by a CCR service-user to inform its neighbour that it is in the ready-to-commit state. The procedure supports the C-READY service defined in ITU-T Rec. X.851 | ISO/IEC 9804.

7.4.2 APDU used

This procedure uses the following CCR-APDU:

C-READY-RI

The structure of this APDU is shown in Figure 4.

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

Figure 4 – C-READY APDU

The C-READY-RI APDU field is listed in Table 10.

Table 10 – C-READY-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.4.3 Prerequisite requirements

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

7.4.4 Signal readiness 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.

7.4.4.1 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 neither concatenated with other CCR APDUs, nor with APDUs from other ASEs, the CCRPM issues a P-TYPED-DATA request primitive with the APDU as a value of the primitive's User Data parameter. If the C-READY-RI APDU is concatenated with anther CCR APDU or with an APDU from another ASE, the appropriate presentation service primitive is issued as specified in clause 11, with the C-READY-RI APDU as a data value in the user data parameter.

7.4.4.2 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 or with APDUs from other ASEs, the C-READY-RI APDU will be received as user data on the appropriate presentation primitive as specified in clause 11. In either case, the CCRPM issues a C-READY indication primitive with the parameter value derived from the APDU.

7.4.5 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 11.

Table 11 – Mapping of C-READY req/ind parameter

APDU Field name	Parameter name
user-data	User Data

7.4.6 Collisions

The signal readiness procedure and the prepare procedure may be used simultaneously by both CCR service-users. 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.

A collision between two C-READY-RI APDUs can occur. Both events are treated normally, resulting in the issue of the appropriate indication primitives.

NOTE – A C-READY-RI APDU may collide with application data. It is the CCR service-user's responsibility to police this situation, which could lead to inconsistency in the bound data of the atomic action (see ITU-T Rec. X.851 | ISO/IEC 9804, C.9.3).

7.5 Order commitment

7.5.1 Purpose

The order commitment procedure is used by a commit-superior to request its commit-subordinate to release its bound data in their final state. It supports the C-COMMIT service defined in ITU-T Rec. X.851 | ISO/IEC 9804.

7.5.2 APDUs used

The procedure uses the following CCR APDUs:

C-COMMIT-RI

C-COMMIT-RC

The structure of these APDUs is shown in Figure 5.

```
C-COMMIT-RI ::= [5] SEQUENCE
      {                              ...,...,
     user-data                       User-data          OPTIONAL }
C-COMMIT-RC ::= [6] SEQUENCE
     { ...,...,
      user-data                      User-data          OPTIONAL }
```

Figure 5 – C-COMMIT APDUs

The C-COMMIT-RI APDU and the C-COMMIT-RC APDU fields are listed in Tables 12 and 13, respectively.

Table 12 – C-COMMIT-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

Table 13 – C-COMMIT-RC field

Field name	Presence	Source	Sink
user-data	U	rsp	cnf

7.5.3 Prerequisite requirements

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

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

7.5.4 Order commitment procedure

This procedure is driven by the following events:

a) C-COMMIT request primitive from the requestor;

b) C-COMMIT-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.

7.5.4.1 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-MINOR request primitive with the APDU as a data value of the primitive's User Data parameter.

7.5.4.2 C-COMMIT-RI APDU

The accepting CCRPM receives a C-COMMIT-RI APDU from its peer as user data on a P-SYNC-MINOR indication primitive. It issues a C-COMMIT indication primitive with the parameter value derived from the APDU.

7.5.4.3 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-MINOR response primitive with the APDU as a data value of the primitive's User Data parameter.

7.5.4.4 C-COMMIT-RC APDU

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

7.5.5 Use of the C-COMMIT-RI APDU field

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 14.

Table 14 – Mapping of C-COMMIT req/ind parameter

APDU Field name	Parameter name
user-data	User Data

7.5.6 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 15.

Table 15 – Mapping of C-COMMIT rsp/cnf parameter

APDU Field name	Parameter name
user-data	User Data

7.5.7 Collision

None.

7.6 Rollback procedure

7.6.1 Purpose

The rollback procedure is used to force completion of an atomic action branch. It supports the C-ROLLBACK service defined in ITU-T Rec. X.851 | ISO/IEC 9804.

7.6.2 APDUs used

The procedure uses the following CCR APDUs:

C-ROLLBACK-RI

C-ROLLBACK-RC

The structure of these APDUs is shown in Figure 6.

```
C-ROLLBACK-RI  ::=  [7]          SEQUENCE
      {  ....,...,
      user-data                  User-data       OPTIONAL }
C-ROLLBACK-RC  ::=  [8]          SEQUENCE
      {  ...,...,
      user-data                  User-data       OPTIONAL }
```

Figure 6 – C-ROLLBACK APDUs

The C-ROLLBACK-RI APDU field is listed in Table 16. The C-ROLLBACK-RC APDU field is listed in Table 17.

Table 16 – C-ROLLBACK-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

Table 17 – C-ROLLBACK-RC field

Field name	Presence	Source	Sink
user-data	U	rsp	cnf

7.6.3 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 commit-superior.

7.6.4 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

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

7.6.4.1 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(abandon) request primitive with the APDU as a data value of the primitive's User Data parameter.

7.6.4.2 C-ROLLBACK-RI APDU

The accepting CCRPM receives a C-ROLLBACK-RI APDU from its peer as user data on a P-RESYNCHRONIZE(abandon) 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 be forgotten.

7.6.4.3 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(abandon) response primitive with the APDU as a data value of the primitive's User Data parameter.

7.6.4.4 C-ROLLBACK-RC APDU

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

7.6.5 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 18.

Table 18 – Mapping of C-ROLLBACK req/ind parameters

APDU Field name	Parameter name
user-data	User Data

7.6.6 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 19.

Table 19 – Mapping of C-ROLLBACK rsp parameter

APDU Field name	Parameter name
user-data	User Data

7.6.7 Disruptive effects

Because the C-ROLLBACK service is mapped on the P-RESYNCHRONIZE 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.

7.6.8 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.

7.7 No-change completion procedure

7.7.1 Purpose

This procedure is used to complete an atomic action branch when the requestor has made no changes to bound data and does not wish to be involved in any two-phase commitment procedures. The requestor may optionally ask to receive an indication of the commitment or rollback of the atomic action. The procedure supports the C-NOCHANGE service defined in ITU-T Rec. X.851 | ISO/IEC 9804.

This procedure is only used when the no-change functional unit has been selected.

7.7.2 APDUs used

The procedure uses the following CCR APDUs:

C-NOCHANGE-RI

C-NOCHANGE-RC

The structure of these APDUs is shown in Figure 7.

```
C-NOCHANGE-RI  ::=  [13]`              SEQUENCE
      { confirmation                   [0] ENUMERATED
                                           { not-required(0), result-requested(1), ...}
          DEFAULT with-result,
        ...,...,
      user-data                        User-data          OPTIONAL
      }

C-NOCHANGE-RC  ::=  [14]               SEQUENCE
      { outcome                        [0] ENUMERATED
                                           { not-determined(0), committed(1), rolled-back(2),
                                         no-change(3), ...}
                                         DEFAULT not-determined ,
        ...,...,
      user-data                        User-data          OPTIONAL
      }
```

Figure 7 – C-NOCHANGE APDUs

The C-NOCHANGE-RI APDU fields are listed in Table 20. The C-NOCHANGE-RC APDU field is listed in table 21.

Table 20 – C-NOCHANGE-RI fields

Field name	Presence	Source	Sink
confirmation	M	req	ind
user-data	U	req	ind

Table 21 – C-NOCHANGE-RC fields

Field name	Presence	Source	Sink
outcome	M	rsp	cnf
user-data	U	rsp	cnf

7.7.3 Prerequisite requirements

For the requestor to issue the C-NOCHANGE request primitive:

– the requestor shall have no bound data for the atomic action.

7.7.4 Procedure operation

This procedure is driven by the following events:

a) C-NOCHANGE request primitive from the requestor;

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

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

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

Events c) and d) are optional and may not occur if the Confirmation parameter on the C-NOCHANGE request has the value “not-required”.

7.7.4.1 C-NOCHANGE request primitive

The requesting CCRPM forms a C-NOCHANGE-RI APDU from parameter values of the C-NOCHANGE request primitive. It issues a P-DATA request primitive with the APDU as a data value of the primitive’s User Data parameter.

7.7.4.2 C-NOCHANGE-RI APDU

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

7.7.4.3 C-NOCHANGE response primitive

The accepting CCRPM forms a C-NOCHANGE-RC APDU from the parameter value of the C-NOCHANGE response primitive. It issues a P-DATA request primitive with the APDU as a data value of the primitive’s User Data parameter.

7.7.4.4 C-NOCHANGE-RC APDU

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

7.7.5 Use of the C-NOCHANGE-RI APDU field

For the requesting and accepting CCRPM: The fields of the C-NOCHANGE-RI APDU are directly mapped from and to the corresponding parameters on the C-NOCHANGE request and indication primitives as specified in Table 22.

Table 22 – Mapping of C-NOCHANGE req/ind parameters

APDU Field name	Parameter name
confirmation	Confirmation
user-data	User Data

7.7.6 Use of the C-NOCHANGE-RC APDU fields

For the accepting and requesting CCRPM: The C-NOCHANGE-RC APDU fields are directly mapped to and from the corresponding parameters on the C-NOCHANGE response and confirm primitives as specified in Table 23.

Table 23 – Mapping of C-NOCHANGE rsp/cnf parameters

APDU Field name	Parameter name
outcome	Outcome
user-data	User Data

7.7.7 Collisions

7.7.7.1 Collision with C-BEGIN-RC APDU, C-PREPARE-RI APDU, C-READY-RI APDU

Collisions are possible with any of these APDUs. The event is treated normally and the corresponding indication primitive is issued by the receiving CCRPM.

7.7.7.2 Collision with C-NOCHANGE-RI APDU

If the NOCHANGE commitment procedure is used simultaneously by both CCR service-users, the C-NOCHANGE-RIs confirm each other. The CCR service-users, on receiving a C-NOCHANGE-RI APDU after sending one, and regardless of the value of the Confirmation parameter, issues a C-NOCHANCE confirm with the Outcome parameter having the value “no-change”.

7.7.7.3 Collision with C-ROLLBACK-RI APDU

A collision with a C-ROLLBACK-RI APDU is handled by session services (see 7.6.7 on the disruptive effects of rollback) – the rollback procedure disrupts the NOCHANGE procedure.

7.8 Cancel procedure

7.8.1 Purpose

This procedure is used to initiate rollback of the atomic action branch. The procedure differs from the Rollback procedure in that the service is non-confirmed, and can be blocked by flow-control. It supports the C-CANCEL service defined in ITU-T Rec. X.851 | ISO/IEC 9804.

7.8.2 APDUs used

The procedure uses the following CCR APDU:

C-CANCEL-RI

The structure of this APDU is shown in Figure 8.

```
C-CANCEL-RI  ::=  [15]                SEQUENCE
      { ...,...,
      user-data                       User-data          OPTIONAL }
```

Figure 8 – C-CANCEL APDU

The C-CANCEL-RI APDU fields are listed in Table 24.

Table 24 – C-CANCEL-RI field

Field name	Presence	Source	Sink
user-data	U	req	ind

7.8.3 Prerequisite requirements

7.8.3.1 For the requestor to issue the C-CANCEL request primitive, it is required that 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.

7.8.3.2 The cancel functional unit shall be selected on the association.

7.8.4 Procedure operation

This procedure is driven by the following events:

a) C-CANCEL request primitive from the requestor;

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

7.8.4.1 C-CANCEL request primitive

The requesting CCRPM forms a C-CANCEL-RI APDU from parameter values of the C-CANCEL request primitive. It issues a P-TYPED-DATA request primitive with the APDU as a data value of the primitive's User Data parameter.

7.8.4.2 C-CANCEL-RI APDU

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

7.8.5 Use of the C-CANCEL-RI APDU fields

For the requesting CCRPM: The fields of the C-CANCEL-RI APDU are directly mapped from the corresponding parameters on the C-CANCEL request primitive as specified in Table 25.

For the accepting CCRPM: The fields of the C-CANCEL-RI APDU are directly mapped to the corresponding parameters on the C-CANCEL indication primitive as specified in Table 25.

Table 25 – Mapping of C-CANCEL req/ind parameters

APDU Field name	Parameter name
user-data	User Data

7.8.6 Collisions

7.8.6.1 Collision with C-BEGIN-RC APDU or C-PREPARE-RI APDU or C-NOCHANGE-RI APDU or C-READY-RI APDU.

Collisions are possible with these APDUs. In such a case, the CCRPM that issued the C-CANCEL-RI APDU discards the incoming APDU and does not issue a CCR service primitive. The CCRPM that receives the C-CANCEL-RI APDU treats the event normally and issues a C-CANCEL indication primitive.

7.8.6.2 Collision with C-CANCEL-RI APDU

A CCRPM that receives a C-CANCEL-RI APDU after issuing C-CANCEL-RI APDU treats the event normally and issues a C-CANCEL indication primitive.

7.8.6.3 Collision with C-ROLLBACK-RI APDU

A C-CANCEL-RI APDU that collides with a C-ROLLBACK-RI APDU will be purged by the session services (see 7.6.7 on the disruptive effects of rollback). The Cancel procedure is disrupted.

7.9 Branch recovery procedure

7.9.1 Purpose

7.9.1.1 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 ITU-T Rec. X.851 | ISO/IEC 9804.

7.9.1.2 This procedure can be used to recover more than one branch at a time on a single association. The procedure is used independently for each branch.

7.9.2 APDUs used

The procedure uses the following CCR APDUs:

C-RECOVER-RI

C-RECOVER-RC

The structure of these APDUs is shown in Figure 9.

```
C-RECOVER-RI ::= [9]              SEQUENCE
    { atomic-action-identifier    [0]     ATOMIC-ACTION-IDENTIFIER,
      branch-identifier           [1]     BRANCH-IDENTIFIER,
      recovery-state              [2]     ENUMERATED
            { commit(0), ready(1), done(2), unknown(3), retry-later(5),... }
      ...,
      reversed-branch             [3]     BOOLEAN DEFAULT FALSE
                                          -- shall be absent if FALSE,
      ...,
      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]     ENUMERATED
    { commit(0), ready(1), done(2), unknown(3), retry-later(5) ...},
    ...,
      reversed-branch             [3]     BOOLEAN DEFAULT FALSE,
                                          -- shall be absent if FALSE,
    ...,
      user-data                   User-data   OPTIONAL

}
```

Figure 9 – C-RECOVER APDUs

The fields of the C-RECOVER-RI APDU are listed in Table 26. The fields of the C-RECOVER-RC APDU are listed in Table 27.

Table 26 – 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
reversed-branch	C	req	ind
user-data	U	req	ind

Table 27 – 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
reversed-branch	C	rsp	cnf
user-data	U	rsp	cnf

7.9.3 Prerequisite requirements

For the requestor, atomic action data for this branch is required to be accessible in stable storage. If the overlapped-recovery functional unit is not selected, then no other atomic action branch shall be active or in recovery on this association. If the branch is not already in recovery on this association, the requestor shall be the owner of the session synchronize-minor token.

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

If the overlapped-recovery functional unit is selected, there shall not be an undisrupted atomic action branch active on the association. The CCR service-user shall ensure that this procedure does not collide with a use of the begin branch procedure on the same association.

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

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

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

7.9.4.1 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.

7.9.4.2 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.

7.9.4.3 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.

7.9.4.4 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.

7.9.5 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 28.

For the requesting CCRPM: If the Atomic Action Identifier or Branch Identifier parameters of the C-RECOVER request contain the AE-title of the requestor, as passed on the A-ASSOCIATE service used to establish the supporting association, the CCRPM shall represent this in the abstract syntax by using either the "name" form or the "sender" value of the "side" form of the corresponding APDU field. Similarly, if the parameters contain the AE-title of the acceptor passed on the A-ASSOCIATE service used to establish the supporting association, the CCRPM shall represent this in the abstract syntax by using either the "name" form or the "receiver" value of the "side" form of the corresponding APDU field.

For the accepting CCRPM: If the "owners-name" field in the "atomic-action-identifier" or the "initiators-name" field in the "branch-identifier" is the "sender" value of the "side" form, the corresponding parameter value shall be the C-RECOVER requestor's AE-title passed on the A-ASSOCIATE service used to establish the supporting association. Similarly, if the "receiver" value of the "side" form is used, the corresponding parameter shall be the C-RECOVER acceptor's AE-title passed in the A-ASSOCIATE service used to establish the supporting association.

Table 28 – Mapping of C-RECOVER req/ind parameters

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

7.9.6 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 29.

For the accepting CCRPM: If the Atomic Action Identifier or Branch Identifier parameters of the C-RECOVER response contain the AE-title of the acceptor, as passed on the A-ASSOCIATE service used to establish the supporting association, the CCRPM shall represent this in the abstract syntax by using either the "name" form or the "sender" value for the "side" form of the corresponding APDU field. Similarly, if the parameters contain the AE-title of the requestor, as passed on the A-ASSOCIATE service used to establish the supporting association, the CCRPM shall represent this in the abstract syntax by using either the "name" form or the "receiver" value of the "side" form of the corresponding APDU field.

For the requesting CCRPM: If the "owners-name" field in the "atomic-action-identifier" or the "initiators-name" field in the "branch-identifier" is the "sender" value of the "side" form, the corresponding parameter value shall be the C-RECOVER acceptor's AE-title passed on the A-ASSOCIATE service used to establish the supporting association. Similarly, if the "receiver" value of the "side" form is used, the corresponding parameter shall be the C-RECOVER requestor's AE-title passed in the A-ASSOCIATE service used to establish the supporting association.

NOTE – The "sender" and "receiver" values identify the peer's by their roles in the transmission of a particular APDU, not the procedure. Thus, a value of "sender" on a C-RECOVER-RI corresponds to a value of "receiver" on the replying C-RECOVER-RC.

Table 29 – 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
was-initiator	Was Initiator
user-data	User Data

7.9.7 Collisions

None.

7.10 Order commitment and begin branch procedure

7.10.1 Purpose

This procedure is used by a commit-superior to request its commit-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 ITU-T Rec. X.851 | ISO/IEC 9804.

7.10.2 APDUs used

The procedure uses the CCR APDUs specified in 7.2.2 and 7.5.2.

7.10.3 Prerequisite requirements

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

7.10.4 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.

7.10.4.1 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-MINOR request primitive with the APDUs as data values of the primitive's User Data parameter.

7.10.4.2 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-MINOR indication primitive. It issues a C-COMMIT indication primitive + a C-BEGIN indication primitive with parameter values derived from the APDUs.

7.10.4.3 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-MINOR response primitive with the APDU as a data value of the primitive's User Data parameter.

7.10.4.4 C-COMMIT-RC APDU

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

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

The procedures of 7.5.5 are followed for the C-COMMIT-RI APDUs fields and of 7.2.5 for the C-BEGIN-RI APDU fields.

7.10.6 Use of the C-COMMIT-RC APDU field

The procedures of 7.5.6 are followed.

7.10.7 Collisions

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

7.11 Error procedure

7.11.1 Purpose

This procedure is used to signal a CCR service provider error condition (e.g. protocol error). It supports the C-P-ERROR service defined in ITU-T Rec. X.851 | ISO/IEC 9804/AM1.

7.11.2 APDUs used

None.

7.11.3 Prerequisite requirements

None.

7.11.4 Procedure operation

This procedure is driven when an error is detected by the CCRPM. It issues a C-P-ERROR indication primitive with the parameter value set by the CCRPM that detects the error.

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, predicates enablements, the actions taken, outgoing events, and, finally, the resultant state of the CCRPM.

8.1 General

8.1.1 Except when the overlapped recovery functional unit is selected, 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 request primitive.

8.1.2 Tables 30 to 35 define the elements used in the State Table:

- Table 30 specifies the abbreviated name and description for each state of the CCRPM.
- Table 31 specifies the abbreviated name, source and description for each incoming event.
- Table 32 specifies the identifier and description for each specific action.
- Table 33 specifies the identifier and description for each predicate.
- Table 34 specifies the identifier and description for each enablement.
- Table 35 specifies the identifier and description for each outgoing event.

8.1.3 The overall CCR State Table is divided into individual tables (see Tables 36 to 43) for convenience and clarity. The individual state tables utilize the abbreviated names and identifiers of Tables 30 to 35. Each state occurs in only individual table.

- Table 36 specifies the states and events that occur in a CCRPM during the initialization procedure (concurrent association establishment) and when it is idle.
- Table 37 specifies the states and events that occur in a CCRPM in phase I, up to rollback, a ready signal, the beginning of a no-change completion procedure or a failure.
- Table 38 specifies the states and events that occur in a CCRPM from sending a ready signal to the receipt of a rollback or commit order.
- Table 39 specifies the states and events that occur in a CCRPM from receipt of a ready signal to the ordering of commitment or rollback.
- Table 40 specifies the states and events that occur in a CCRPM following a rollback or cancel order.
- Table 41 specifies the states and events that occur in a CCRPM following a commit order.
- Table 42 specifies the states and events that occur in a CCRPM during the no-change completion procedure.
- Table 43 specifies the states and events that occur in a CCRPM during the recovery procedure.

8.2 Incoming events

8.2.1 The types of incoming events specified in Table 31 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;

f) the receipt of an A-ASSOCIATE confirm with no C-INITIALIZE-RC APDU in its User Data, when the previous A-ASSOCIATE request contained a C-INITIALIZE-RI APDU.

NOTE – This event occurs when an association is being established with an implementation that supports only the CCR kernel functional unit (see Annex C).

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 31 is treated as the consecutive occurrence of individual incoming events.

8.3 Outgoing events

The types of outgoing events specified in Table 32 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 33 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 an expression that has either a “true” or “false” value. The CCRPM predicates specified in Table 34 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 minor sync tokens;

c) whether particular CCR functional units have been selected on the association.

Predicates determine what outgoing events, specific actions and resultant states apply for particular incoming events and state combinations. If all applicable predicates are false, the incoming event is invalid.

8.6 Enablements

Enablements permit changes data related to an atomic action. An enablement is required in order for changes to be made to:

a) atomic action data; or

b) bound data.

An enablement does not require that changes be made. However, some of the predicates defined in Table 37 test whether changes allowed by particular enablements have been made. Enablements for the CCRPM are defined in Table 35 and have names of the form e*N, where * represents the letter a or b and indicates whether the enablement concerns atomic action data or bound data, and N is an identifying integer.

8.7 Variables

The following variables are specified for the CCRPM:

a) *Current-Branch*;

b) *Next-Branch*.

At any time, each of the *Current-Branch* and *Next-Branch* variables 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 subclause, 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 (see Tables 36 to 43).

– The CCRPM states specified in Table 30 are represented by the notation “Zn”, where Z is an upper-case letter and n is null or an integer. The letter indicates the general state reached in the progress of the branch. In the labels for the columns of the state table, each state is also identified by an abbreviated indication of the immediately previous events. These abbreviations are informative only.

– Incoming events are represented by the names assigned in Table 31.

– Specific actions are represented by the notation “[n]”, where n is the action number assigned in Table 32.

– Predicates are represented by the notation “p*”, “pf...” as assigned in Table 33, where n is an integer and ... is a mnemonic.

– The Boolean operator AND is represented by the notation “&”.

– The Boolean operator OR is represented by the notation “|”.

– The Boolean operator NOT is represented by the notation “~”.

– Enablements are represented by the notation “ean” or “ebn” as assigned in Table 34, where n is an integer. Where more than one enablement of the same kind is shown for a CCRPM state, both apply.

– The outgoing events are represented by the identifier assigned for the event in Table 35.

The informative abbreviations that summarize the previous events in the column headings for each state generally use the following conventions:

- three-letter abbreviations refer to CCR primitives and -RI APDUs;
- the character ">" following such an abbreviation refers to the sending of the CCR APDU;
- the character "<" preceding a three-letter abbreviation refers to the issue of the indication;
- the character "*" following a three-letter abbreviation refers to the completion of the corresponding procedure.

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 one or more action lists, each consisting of :

a) a predicate expression (optional);

b) a specific action (optional);

c) an outgoing event (optional); and

d) a resultant state.

8.9.3 If the intersection of an incoming event and the column "Predicate" is blank, there is no predicate for the event. If a predicate expression is shown in the "Predicate" cell for the event, it is to be read as if the predicate was present at the head of every action list in every non-empty cell for the row, linked to predicates in a particular cell by the AND Boolean operator.

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 when 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 its state "S0" when the association establishment procedure begins.

The *Current-Branch* and *Next-Branch* variables are set to "null".

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 in the CCRPM tables 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 not determined by this Protocol Specification. 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 the receipt of one or more CCR APDUs from the peer CCRPM, the CCRPM shall:

1) issue a C-P-ERROR indication; and

2) change its state to X.

NOTE – No further CCRPM events can take place. However, if the association is released or aborted, the CCRPM will cease to exist.

8.10.3 Valid intersections

Non-blank cells in the CCRPM tables 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) at the head of (one of) the action list(s) in the cell is true,

the following actions are taken:

c) the CCRPM performs the specific action or actions (if any) in the action list;

d) the specified output event or events (if any) in the action list are performed;

e) the state of the CCRPM is changed to the resultant state specified in the action list.

If the predicate expression in the Predicate column is false, the CCRPM follows the procedure for an invalid intersection as specified in 8.10.2.

NOTE – Where a cell contains multiple actions lists, it will be found that the predicate expressions are complementary.

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 recovering the branch is in a state for which an enablement in Table 34 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 32 of determining that the atomic action branch is completed.

8.12 Changes to bound data

8.12.1 Bound data can be manipulated in the following ways:

a) changes to produce the final state of the bound data through the normal progression of the atomic action; or

b) release of all data in the initial state as the result of a rollback decision; or

c) release of all data in the final state as the result of a commitment decision.

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

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

b) atomic action data does not reflect a ready decision; and

c) atomic action data does not reflect a commit decision.

8.12.3 Release of all data in the initial state as the result of a rollback decision requires that:

a) a CCRPM is in a state for which an enablement permits the release; and

b) atomic action data does not reflect a commit decision.

8.12.4 Release of all data in the final state as the result of a commitment decision requires that:

a) a CCRPM is in a state for which an enablement permits the release; and

b) atomic action data reflects either a ready or a commit decision.

Table 30 – CCRPM States

Name	Abbreviation	Description
A1	bgn>	C-BEGIN-RI sent
A13	bgn> <bga (stat)	With static commitment, initiator has completed begin branch procedure.
A2	<bgn	C-BEGIN ind issued
A23	<bgn bga> (stat)	With static commitment, responder has completed begin branch procedure.
A3	bgn*	With dynamic commitment, begin branch procedure completed.
A4	bgn> prp>	C-BEGIN-RI and C-PREPARE-RI sent
A5	bgn* prp>	Begin branch procedure completed and C-PREPARE-RI sent
A6	<bgn <prp	C-BEGIN ind and C-PREPARE ind issued
A7	bgn* <prp	Begin branch procedure completed and C-PREPARE ind issued
A8	prp> <prp	C-PREPARE ind issued and C-PREPARE-RI sent
B1	bgn> rdy>	C-BEGIN-RI and C-READY-RI sent
B2	bgn> prp> rdy>	C-BEGIN-RI, C-PREPARE-RI and C-READY-RI sent
B3	bgn* rdy>	Begin branch procedure completed, C-READY-RI sent.
B4	bgn* prp> rdy>	Begin branch procedure completed, C-PREPARE-RI and C-READY-RI sent.
B5	<prp rdy>	C-PREPARE ind issued and C-READY-RI sent
B6	<prp prp> rdy>	C-READY-RI sent and C-PREPARE collision
C1	<rdy	C-READY-RI received
D1	rdy> <rdy	C-READY-RI sent and received
E1	<cmt	C-COMMIT ind issued
E2	<cmtbg	C-COMMIT ind with C-BEGIN ind issued
F1	rbk>	C-ROLLBACK-RI sent
F2	<rbk	C-ROLLBACK ind issued
F3	<rdy rbk>	C-READY-RI received and C-ROLLBACK-RI sent
G1	cmt>	C-COMMIT-RI sent
G2	cmtbg>	C-COMMIT-RI with C-BEGIN-RI sent
I	idle	No action in progress
J1	bgn> nch>	C-BEGIN-RI and C-NOCHANGE-RI sent
J2	bgn* nch>	Begin branch procedure completed and C-NOCHANGE-RI sent
J3	nch> <prp	C-NOCHANGE-RI sent and C-PREPARE-RI received
J4	nch> <rdy	C-NOCHANGE-RI sent and C-READY-RI received
K1	<nch	C-NOCHANGE ind issued
M1	can>	C-CANCEL-RI issued
M2	<can	C-CANCEL ind issued
R1	Rcmt>	C-RECOVER(commit)-RI sent
R2	<Rrdy	C-RECOVER(ready)-RI received
R3	Rrdy>	C-RECOVER(ready)-RI sent
R4	<Rcmt	C-RECOVER(commit)-RI received
S0	void	No association, CCRPM void
S1	ini>	C-INITIALIZE-RI sent
S2	<ini	C-INITIALIZE ind issued
X	error	Protocol error has been detected.

Table 31 – Incoming events

Name	Source	Description
BEGIN-RI	CCR-peer	C-BEGIN-RI APDU received by accepting CCRPM
BEGIN-RC	CCR-peer	C-BEGIN-RC APDU received by requesting CCRPM
BEGINreq	CCR-user	C-BEGIN request primitive issued by requestor
BEGINrsp	CCR-user	C-BEGIN primitive issued by acceptor
CMT+BGN-RI	CCR user	C-COMMIT-RI+ C-BEGIN-RI concatenated APDU received by accepting CCRPM
CMT+BGNreq	CCR-user	C-COMMIT request primitive with a C-BEGIN request primitive issued by requestor
COMMIT-RC	CCR-peer	C-COMMIT-RC APDU received by requesting CCRPM
COMMIT-RI	CCR-peer	C-COMMIT-RI APDU received by accepting CCRPM
COMMITreq	CCR-user	C-COMMIT request primitive issued by requestor
COMMITrsp	CCR-user	C-COMMIT primitive issued by acceptor
DISRUPT	ACSE provider or user	(provider) A-ABORT indication or A-P-ABORT indication received; (user) A-ABORT request.
INIT-RI	CCR-peer	C-INITIALIZE-RI APDU received by accepting CCRPM
INIT-RC	CCR-peer	C-INITIALIZE-RC APDU received by requesting CCRPM
INITreq	CCR-su	C-INITIALIZE request primitive issued by requestor
INITrsp	CCR-su	C-INITIALIZE primitive issued by acceptor
NOCHANGE-RC	CCR-peer	C-NOCHANGE-RC APDU received by requesting CCRPM
NOCHANGE-RI	CCR-peer	C-NOCHANGE-RI APDU received by accepting CCRPM
NOCHANGEreq	CCR-user	C-NOCHANGE request primitive issued by requestor
NOCHANGErsp	CCR-user	C-NOCHANGE response primitive issued by acceptor
PREPARE-RI	CCR-peer	C-PREPARE-RI APDU received by accepting CCRPM
PREPAREreq	CCR-user	C-PREPARE request primitive issued by requestor
RCV-RC(done)	CCR-peer	C-RECOVER-RC(done) APDU with recovery-state = “done” received by requesting CCRPM
RCV-RC(retry-later)	CCR-peer	C-RECOVER-RC(retry-later) APDU with recovery-state = “retry-later” received by requesting CCRPM
RCV-RI(ready)	CCR-peer	C-RECOVER-RI(ready) APDU with recovery-state = “ready” received by accepting CCRPM
RCV-RI(commit)	CCR-peer	C-RECOVER-RI(commit) APDU with recovery-state = “commit” received by accepting CCRPM
RCV-RC(unknown)	CCR-peer	C-RECOVER-RC(unknown) APDU with recovery-state = “unknown” received by requesting CCRPM
RCV(commit)req	CCR-user	C-RECOVER(commit) request primitive with Recovery State = “commit” issued by requestor
RCV(done)rsp	CCR-user	C-RECOVER(done) primitive with Recovery State = “done” issued by acceptor
RCV(ready)req	CCR-user	C-RECOVER(ready) request primitive with Recovery State = “ready” issued by requestor
RCV(retry-later)rsp	CCR-user	C-RECOVER(retry-later) primitive with Recovery State = “retry-later” issued by acceptor
RCV(unknown)rsp	CCR-user	C-RECOVER(unknown) primitive with Recovery State = “unknown” issued by acceptor
READY-RI	CCR-peer	C-READY-RI APDU received by accepting CCRPM
READYreq	CCR-user	C-READY request primitive issued by requestor
ROLLBACK-RC	CCR-peer	C-ROLLBACK-RC APDU received by requesting CCRPM
ROLLBACK-RI	CCR-peer	C-ROLLBACK-RI APDU received by accepting CCRPM
ROLLBACKreq	CCR-user	C-ROLLBACK request primitive issued by requestor
ROLLBACKrsp	CCR-user	C-ROLLBACK primitive issued by acceptor

Table 32 – 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.
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.
8	The *Current-Branch* variable is set to the atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-RECOVER primitive or APDU.
9	*Current-Branch* variable is set to "null".

Table 33 – Predicates

Predicate	Description
p1	Atomic action data for the commit-superior of the current branch is accessible in stable storage and either the atomic action data reflects a commit decision or the CCR service-user has been ordered to commit by its commit-superior on another branch.
p2	Either p4 is true or the CCR service-user has been ordered to rollback by its commit-superior on another branch.
p3	Atomic action data for the commit-subordinate of the current branch is accessible in stable storage.
p4	No atomic action data for the current branch is accessible in stable storage.
p7	The minor synchronize token is in the possession of the requestor.
p9	The atomic action branch identified by the Atomic Action Identifier and Branch Identifier on the C-RECOVER request primitive or C-RECOVER-RI APDU is the branch identified by the *Current-Branch* variable.
pdy	The dynamic commitment functional unit is selected.
pnc	The no-change functional unit is selected.
pcn	The cancel functional unit is selected.
prcl	The Ready-collision-reservation field on whichever of C-INITIALIZE-RI or C-INITIALIZE-RC was sent had the value "true" or was absent. NOTE 1 – An implementation can ensure that prcl is always true if it never issues a C-INITIALIZE request or response with the Ready-collision-reservation parameter set to "false".
prcr	The Ready-collision-reservation field on whichever of C-INITIALIZE-RI or C-INITIALIZE-RC was received had the value "true" or was absent. NOTE 2 – A referencing specification can ensure that prcr is always true if the implementation is only used to support a referencing specification which requires that a C-INITIALIZE request or response is never issued with the Ready-collision-reservation parameter set to "false".

Table 34 – Enablements

Code	Description
ea1	Atomic action data reflecting a commit decision or the receipt of a ready signal on of the current branch can be made accessible in stable storage.
ea2	Atomic action data reflecting a ready signal one of the current branch can be made accessible in stable storage.
ea3	Atomic action data reflecting a ready signal of the current branch can cease to be accessible in stable storage.
ea4	ea1 AND IF prcl THEN ea3
eb5	Bound data can be changed to produce the final state.
eb6	Bound data can be released in the initial state as part of the normal rollback procedures.
eb7	Bound data can be released in the final state as part of the normal commitment procedures.
eb8	Bound data can be released as part of a heuristic decision.

Table 35 – Outgoing events

Code	Description
sbga	Issue C-BEGIN cnf.
sbgn	Issue C-BEGIN ind.
scan	Issue C-CANCEL ind.
scma	Issue C-COMMIT cnf.
scmt	Issue C-COMMIT ind.
scmtbg	Issue C-COMMIT ind with C-BEGIN ind.
serr	Issue C-P-ERROR ind.
sina	Issue C-INITIALIZE cnf.
sini	Issue C-INITIALIZE ind.
snca	Issue C-NOCHANGE cnf.
snci	Issue C-NOCHANGE ind.
sncc	Issue C-NOCHANGE cnf with Outcome parameter “nochange”.
sprp	Issue C-PREPARE ind.
srba	Issue C-ROLLBACK cnf.
srbk	Issue C-ROLLBACK ind.
srca	Issue C-RECOVER cnf.
srcv	Issue C-RECOVER ind.
srdy	Issue C-READY ind.
tbga	Send C-BEGIN-RC APDU.
tbgn	Send C-BEGIN-RI APDU.
tcan	Send C-CANCEL-RI APDU.
tcma	Send C-COMMIT-RC APDU.
tcmt	Send C-COMMIT-RI APDU.
tcmtbg	Send C-COMMIT-RI APDU and C-BEGIN-RI APDU on the same presentation primitive.
tina	Send C-INITIALIZE-RC APDU.
tini	Send C-INITIALIZE-RI APDU.
tnca	Send C-NOCHANGE-RC APDU.
tnci	Send C-NOCHANGE-RI APDU.
tprp	Send C-PREPARE-RI APDU.
trba	Send C-ROLLBACK-RC APDU.
trbk	Send C-ROLLBACK-RI APDU.
trca	Send C-RECOVER-RC APDU.
trcv	Send C-RECOVER-RI APDU.
trdy	Send C-READY-RI APDU.

Table 36 – CCRPM State Table – Initialisation, idle and error

Incoming event	Predicate	Preceding state				
		S0 void	S1 ini>	S2 <ini	I idle	X error
INITreq		tini S1				
INIT-RI		sini S2				
INITrsp			sina I			
INIT-RC				tina I		
BEGINreq	p7				[1] tbgn A1	
BEGIN-RI					[5] sbgn A2	
RCV(commit)req					p7 [8] trcv R1	
RCV(ready)req					p7 [8] trcv R3	
RCV-RI(commit)					[8] srcv R4	
RCV-RI(ready)					[8] srcv R2	
DISRUPT		S0	S0	S0	S0	S0
Enablements atomic action data bound data						

Table 37 – CCRPM State Table – Phase I

Incoming event	Predicate	Preceding state									
		A1 bgn>	A2 <bgn	A13 bgn> <bga (stat)	A23 <bgn bga> (stat)	A3 bgn*	A4 bgn> prp>	A5 bgn* prp>	A6 <bgn <prp	A7 bgn* <prp	A8 prp> <prp
BEGINrsp			pdy tbga A3 ~pdy tbga A23						tbga A7		
BEGIN-RC		pdy sbga A3 ~pdy sbga A13					sbga A5				
PREPAREreq		tprp A4	pdy tprp A5	tprp A5		tprp A5			pdy tprp A8	pdy tprp A8	
PREPARE-RI		pdy sprp A7	sprp A6		sprp A7	sprp A7	pdy sprp A8	pdy sprp A8			
READYreq	p3	pdy trdy B1	trdy B3		trdy B3	trdy B3	pdy trdy B2	pdy trdy B4	trdy B5	trdy B5	trdy B6
READY-RI		srdy C1	pdy srdy C1	srdy C1		srdy C1	srdy C1	srdy C1	pdy srdy C1	pdy srdy C1	srdy C1
ROLLBACKreq	p2	trbk F1	trbk F1	trbk F1	trbk F1	trbk F1	trbk F1	trbk F1	trbk F1	trbk F1	trbk F1
ROLLBACK-RI		srbk F2	srbk F2	srbk F2	srbk F2	srbk F2	srbk F2	srbk F2	srbk F2	srbk F2	srbk F2
CANCELreq	p2&pcn	tcan M1	tcan M1	tcan M1	tcan M1	tcan M1	tcan M1	tcan M1	tcan M1	tcan M1	tcan M1
CANCEL-RI	pcn	scan M2	scan M2	scan M2	scan M2	scan M2	scan M2	scan M2	scan M2	scan M2	scan M2
NOCHANGEreq	pnc&p4	tnci J1	tnci J2	tnci J2	tnci J2	tnci J2	tnci J1	tnci J2	tnci J3	tnci J3	tnci J3
NOCHANGE-RI	pnc	snci K1	snci K1	snci K1	snci K1	snci K1	snci K1	snci K1	snci K1	snci K1	snci K1
DISRUPT		S0	S0	S0	S0	S0	S0	S0	S0	S0	S0
Enablements atomic action data bound data		 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5	 ea2 eb5

Table 38 – CCRPM State Table – After sending a ready signal

Incoming event	Predicate	Preceding state					
		B1 bgn> rdy>	B2 bgn> prp> rdy>	B3 bgn* rdy>	B4 bgn* prp> rdy>	B5 <prp rdy>	B6 <prp prp> rdy>
BEGIN-RC		B3	B4				
PREPARE-RI		sprp B5	pdy sprp B6	sprp B5	pdy sprp B6		
READY-RI		pdy srdy D1	srdy D1	pdy srdy D1	srdy D1	pdy srdy D1	srdy D1
ROLLBACK-RI		srbk F2	srbk F2	srbk F2	srbk F2	srbk F2	srbk F2
CANCEL-RI	pcn	scan M2	scan M2	scan M2	scan M2	scan M2	scan M2
COMMIT-RI		scmt E1	scmt E1	scmt E1	scmt E1	scmt E1	scmt E1
CMT+BGN-RI		[6] scmtbg E2	[6] scmtbg E2	[6] scmtbg E2	[6] scmtbg E2	[6] scmtbg E2	[6] scmtbg E2
NOCHANGE-RI	pnc	snci K1	snci K1	snci K1	snci K1	snci K1	snci K1
DISRUPT		S0	S0	S0	S0	S0	S0
Enablements atomic action data bound data		eb8	eb8	eb8	eb8	eb8	eb8

Table 39 – CCRPM State Table – After receiving a ready signal

Incoming event	Predicate	Preceding state	
		C1 <rdy	D1 rdy> <rdy
ROLLBACKreq	p2	trbk F3	prcl trbk F3
ROLLBACKreq			prcr srbk F2
CANCELreq	p2&pcn	tcan M1	
COMMITreq	p1 & p7	tcmt G1	tcmt G1
COMMIT-RI			scmt E1
CMT+BGNreq	p1 & p7	[3] tcmtbg G2	[3] tcmtbg G2
CMT+BGN-RI			[6] scmtbg E2
NOCHANGEreq	pnc&p4	tnci J4	
DISRUPT		S0	S0
Enablements atomic action data bound data		ea1	ea4 eb8

Table 40 – CCRPM State Table – After cancel or rollback

Incoming event	Predicate	Preceding state				
		M1 can>	M2 <can	F1 rbk>	F2 <rbk	F3 <rdy rbk>
BEGIN-RC		M1				
PREPARE-RI		M1				
READY-RI		M1				
ROLLBACKreq	p2	trbk F1	trbk F1			
ROLLBACK-RI		srbk F2	srbk F2	srbk F2		
ROLLBACKrsp	p4				[2] trba I	
ROLLBACK-RC				[2] srba I		[2] srba I
CANCEL-RI	pcn	scan M2				
NOCHANGE-RI	pnc	M1				
DISRUPT		S0	S0	S0	S0	S0
Enablements atomic action data bound data					ea3 eb6	

Table 41 – CCRPM State Table – After commit order

Incoming event	Predicate	Preceding state			
		E1 <cmt	E2 <cmtbg	G1 cmt>	G2 cmtbg>
COMMITrsp	p4	[2] tcma I	[4] tcma A2		
COMMIT-RC				[2] scma I	[4] scma A1
DISRUPT		S0	S0	S0	S0
Enablements atomic action data bound data		ea3 eb7	ea3 eb7		

Table 42 – CCRPM State Table – No-change completion

Incoming event	Predicate	Preceding state				
		J1 bgn> nch>	J2 bgn* nch>	J3 nch> <prp	J4 nch> <rdy	K1 <nch
BEGINreq	p7					[1] tbgn A1
BEGIN-RI		[5] sbgn A2	[5] sbgn A2	[5] sbgn A2	[5] sbgn A2	
BEGIN-RC		J2				
PREPARE-RI		J3	J3			
READY-RI		J4	J4	J4		
ROLLBACKreq	p2					trbk F1
ROLLBACK-RI		srbk F2	srbk F2	srbk F2		
CANCEL-RI	pcn	scan M2	scan M2	scan M2		
NOCHANGE-RI	pnc	sncc I	sncc I	sncc I	sncc I	
NOCHANGErsp						[2] tnca I
NOCHANGE-RC		[2] snca I	[2] snca I	[2] snca I	[2] snca I	
RCV(commit)req						p7 [8] trcv R1
RCV(ready)req						p7 [8] trcv R4
RCV-RI (commit)		[8] srcv R3	[8] srcv R3	[8] srcv R3	[8] srcv R3	
RCV-RI (ready)		[8] srcv R2	[8] srcv R2	[8] srcv R2	[8] srcv R2	
DISRUPT		S0	S0	S0	S0	S0
Enablements atomic action data bound data						 ea3 eb6 eb7

Table 43 – CCRPM State Table – Completing recovery

Incoming event	Predicate	Preceding state			
		R1 Rcmt>	R2 <Rrdy	R3 Rrdy>	R4 <Rcmt
RCV(commit)req	p1		p9 trcv R1		
RCV(commit)-RI				p9 srcv R4	
RCV(done)rsp	p4				[2] srcv I
RCV(done)-RC		[2] trcv I			
RCV(unknown)rsp	p2		[9] trcv I		
RCV(unknown)-RC				[2] srcv I	
RCV(retry-later)rsp			[9] trcv I		[9] trcvId
RCV(retry-later)-RC		[9] srcv I		[9] srcv I	
DISRUPT		S0	S0	S0	S0
Enablements atomic action data bound data					 ea3 eb7

9 Reference mapping to the ACSE and presentation services

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 when this reference mapping is used. Table 44 summarizes the mapping of CCR primitives and their related APDUs to the presentation primitives used.

NOTE – Annex B defines the rules and constraints for specifications of alternative mappings to support communication services.

9.1 Initialize

The initialize procedure uses the A-ASSOCIATE service.

9.1.1 Use of the A-ASSOCIATE parameters

9.1.1.1 User Data: This parameter is used to carry the C-INITIALIZE-RI and C-INITIALIZE-RC APDUs. User Data (if any) on the C-INITIALIZE request and response primitives are included in the APDU and are expressed using one or more presentation contexts specified by the requestor on the C-INITIALIZE primitive.

Table 44 – Mapping overview

CCR primitive or primitive combination	CCR APDU or APDUs	Presentation primitive
C-INITIALIZE req/ind	C-INITIALIZE-RI	A-ASSOCIATE req/ind
C-INITIALIZE rsp/cnf	C-INITIALIZE-RC	A-ASSOCIATE rsp/cnf
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 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(abandon) req/ind
C-ROLLBACK rsp/cnf	C-ROLLBACK-RC	P-RESYNC(abandon) rsp/cnf
C-COMMIT req/ind	C-COMMIT-RI	P-SYNC-MINOR req/ind
C-COMMIT rsp/cnf	C-COMMIT-RC	P-SYNC-MINOR rsp/cnf
C-COMMIT req/ind + C-BEGIN req/ind	C-COMMIT-RI followed by C-BEGIN-RI	P-SYNC-MINOR req/ind
C-COMMIT rsp/cnf + C-BEGIN rsp/cnf	C-COMMIT-RC followed by C-BEGIN-RC	P-SYNC-MINOR rsp/cnf
C-NOCHANGE req/ind	C-NOCHANGE-RI	P-TYPED-DATA req/ind
C-NOCHANGE rsp/cnf	C-NOCHANGE-RC	P-TYPED-DATA req/ind
C-CANCEL req/ind	C-CANCEL-RI	P-TYPED-DATA req/ind
C-RECOVER req/ind	C-RECOVER-RI	P-TYPED-DATA req/ind
C-RECOVER rsp/cnf	C-RECOVER-RC	P-TYPED-DATA req/ind

9.1.1.2 Presentation context definition list: This shall include at least the abstract syntax “ccr-syntax-apdus-2” (see Annex A).

9.1.1.3 Presentation requirements: See 6.2.

9.1.1.4 Calling AE qualifier: See 6.2.

9.1.1.5 Responding AP title: See 6.2.

9.1.1.6 Responding AE qualifier: See 6.2.

9.1.1.7 Session requirements: See 6.2.

9.1.1.8 All other parameters: The setting of the other parameters is not specified in this Protocol Specification.

9.2 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.

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

9.2.1.1 Type: This mandatory parameter is set to the value of “optional”.

9.2.1.2 Synchronization Point Serial Number: The use of this value is not determined by in this Protocol Specification.

9.2.1.3 Data Separation: This parameter is set TRUE by the CCRPM on the request primitive.