Standard ECMA-294

December 1999

ECMA

Standardizing Information and Communication Systems

Broadband Integrated Services Digital Network (B-ISDN) and Broadband Private Integrated Services Network (B-PISN) - Digital Subscriber Signalling System No. two (DSS2), Broadband Inter-Exchange Signalling (B-QSIG), and Signalling System No. 7 (SS7) - Call Control in a Separated Call and Bearer Control Environment - Part 1: Protocol Specification

Standard ECMA-294
December 1999

ECMA

Standardizing Information and Communication Systems

Broadband Integrated Services Digital Network (B-ISDN) and Broadband Private Integrated Services Network (B-PISN) - Digital Subscriber Signalling System No. two (DSS2), Broadband Inter-Exchange Signalling (B-QSIG), and Signalling System No. 7 (SS7) - Call Control in a Separated Call and Bearer Control Environment - Part 1: Protocol Specification

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:** http://www.ecma.ch - **Internet:** helpdesk@ecma.ch
IW ECMA-294.DOC 06-01-00 09,01

Brief History

This Standard is one of a series of ECMA Standards defining services and signalling protocols applicable to Broadband Private Integrated Services Networks (B-PISNs). The series uses B-ISDN concepts as developed by ITU-T and conforms to the framework of International Standards for Open Systems Interconnection as defined by ISO/IEC.

This Standard has been produced by ECMA TC32-TG15 in collaboration with ETSI Technical Committee Signalling Protocols and Switching (SPS) under ETSI work item DEN/SPS-05132-1.

The Standard is part 1 of a multi-part standard covering the Digital Subscriber Signalling System No. 2 (DSS2), Broadband Inter-Exchange Signalling (B-QSIG), and Signalling System No. 7 (SS7) protocol specification for the Broadband Integrated Services Digital Network (B-ISDN) and Broadband Private Integrated Services Network (B-PISN) Call Control, as described below:

Part 1: "Protocol specification";

Part 2: "Protocol Implementation Conformance Statement (PICS) proforma specification";

Part 3: "Test Suite Structure and Test Purposes (TSS&TP) specification";

Part 4: "Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma specification".

Part 3 and part 4 will only be produced by ETSI as EN 302 092-3 and EN 302 092-4 respectively.

The Standard is based upon the practical experience of ECMA member companies and the results of their active and continuous participation in the work of ISO/IEC JTC1, ITU-T, ETSI and other international and national standardization bodies. It represents a pragmatic and widely based consensus.

This ECMA Standard is technically aligned with EN 302 092-1 published by ETSI in November 1999.

This Standard has been adopted by the ECMA General Assembly of December 1999.

Table of contents

1 Scope

This Standard specifies a signalling protocol for the purpose of call control at the Q_B, S_B, T_B, and co-incident S_B/T_B reference points within, between, and at the access to Broadband Private Integrated Services Networks and within, between, and at the access to European Broadband Integrated Services Digital Networks. The protocol operates between two adjacent call control entities. The protocol is applicable to a terminal or network node in a separated call and bearer (connection) control environment for the support of calls having none, a single bearer or multiple bearers. The protocol is applicable to a two-party call. The protocol also provides forward compatibility to the extent that an implementation can also operate within a multi-party call with other implementations that use additional capabilities, provided the implementation is deployed where it does not need to be aware of more than two parties.

This Standard is related to other Standards in this series which will describe the architecture of a separated call and bearer control environment and scenarios in which such an architecture can be applied.

The protocol specified in this Standard is independent of the supporting transport service.

The protocol specified in this Standard is independent of the protocol used for bearer establishment.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of this Standard.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

ECMA-254	Broadband Private Integrated Services Network (B-PISN) - Inter-Exchange Signalling Protocol - Generic Functional Protocol (B-QSIG-GF)
ETS 300 796-1	Broadband Integrated Services Digital Network (B-ISDN); Digital Subscriber Signalling System No. two (DSS2) protocol; Generic functional protocol; Core aspects; Part 1: Protocol specification [ITU-T Rec. Q.2932.1 (1996), modified]
ITU-T Rec. X.680	Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation
ITU-T Rec. X.681	Information Technology - Abstract Syntax Notation One (ASN.1): Information object specification
ITU-T Rec. X.682	Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification
ITU-T Rec. X.683	Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications
ITU-T Rec. X.690	Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rues (CER) and Distinguished Encoding Rules (DER)
ITU-T Rec. X.880	Information technology - Remote operations: concepts, model and notation
CCITT Rec. Z.100	CCITT specification and description language (SDL)

3 Definitions

For the purposes of this Standard the following definitions apply.

3.1 Adjacent call control entities (adjacent CC entities)

Within the context of a single call, two CC entities that signal directly to each other with no intervening CC entity.

3.2 Bearer

A connection for the transport of user plane information between users involved in a call.

3.3 Bearer control

Functionality and signalling in and between networks and terminals to effect the control of a bearer, that bearer being part of a call.

3.4 Bearer control entity (BC entity)

An entity that is located in a terminal or a network and that participates in bearer control.

3.5 Call

An association between two or more users using a telecommunication service to communicate through one or more networks.

3.6 Call control

Functionality and signalling in and between networks and terminals to effect the control of a call, excluding the control of individual bearers.

3.7 Call control entity (CC entity)

An entity that is located in a terminal or a network and that participates in call control.

3.8 Call control signalling service provider

An entity which provides the signalling services of call control.

3.9 Call control signalling service user

An entity within the CC entity to which the signalling services of call control are provided.

NOTE

The call control signalling service user performs the Call Description handling, provides the interactions with bearer control, and in a network node, co-ordinates the incoming and outgoing side of the CC entity (figure 6).

3.10 Call segment

A part of a call located between two adjacent call control entities.

3.11 Called party

Any party in a call other than the calling party.

3.12 Calling party

The party which initiates the call establishment.

3.13 Information model

A representation of the service and abstract communications configuration using an object oriented technique.

3.14 Originating CC entity

The CC entity that initiates call establishment and is located in a terminal or equipment that functions like a terminal (e.g. a server in the network).

3.15 Party

An addressable signalling endpoint.

3.16 Preceding CC entity

The CC entity that initiates call establishment across a given call segment.

3.17 Succeeding CC entity

The CC entity at the opposite end of a call segment from the preceding CC entity.

3.18 Terminating CC entity

The CC entity to which call establishment is directed and that is located in a terminal or equipment that functions like a terminal.

3.19 Transit CC entity

A CC entity through which a call passes, excluding the originating and terminating CC entity.

4 Abbreviations

For the purposes of this Standard, the following abbreviations apply:

APDU	Application Protocol Data Unit
ASN.1	Abstract Syntax Notation One
BC	Bearer Control
B-PISN	Broadband Private Integrated Services Network
CC	Call Control
M	Mandatory
O	Optional

5 Basic model

5.1 Separation of Call Control (CC) and Bearer control (BC)

The protocol specified in this Standard is applicable to an environment in which the control of a call is separate from the control of the bearer or bearers that exist within the context of that call.

In order for two users to communicate using a telecommunication service, an association, or call, is established between the two users. Within the context of the call, one or more bearers can be established for transporting user plane information between the users. However, bearers are not normally established until the call has been accepted by the called terminal, and therefore resources required by bearers are not occupied unnecessarily if the call cannot be established, e.g., if the called terminal is unable to accept the call because resources are not available. During the life-time of the call, bearers can be added or cleared down as required. The call is terminated by one of the two users when there is no further need for communication. Termination of the call implies that all bearers (if any) are cleared down.

Although for some telecommunication services a single bearer is sufficient, other telecommunication services benefit from the use of multiple bearers, each tailored to suit the characteristics of the user plane information to be transported. This is particularly true for multi-media applications involving audio, video and data. The call provides a context in which the various bearers can exist and a means of binding the bearers together.

5.2 Point-to-point and multi-party CC

In a point-to-point configuration the protocol specified in this Standard operates between any two adjacent CC entities between the calling party and the called party.

In a multi-party configuration the protocol specified in this Standard operates between any two adjacent CC entities on the point-to-point leg between a co-ordination point in the network and a called party. In this case a network node in the calling party's network takes responsibility for co-ordinating the responses from multiple point-to-point signalling associations to the called parties into one signalling association to the calling party. Enhancements to be made to the CC protocol in order to make it suitable also for this signalling association to the calling party in case of a multi-party call are outside the scope of this Standard.

Figure 1 shows an example configuration for a multi-party call with two called parties involved.

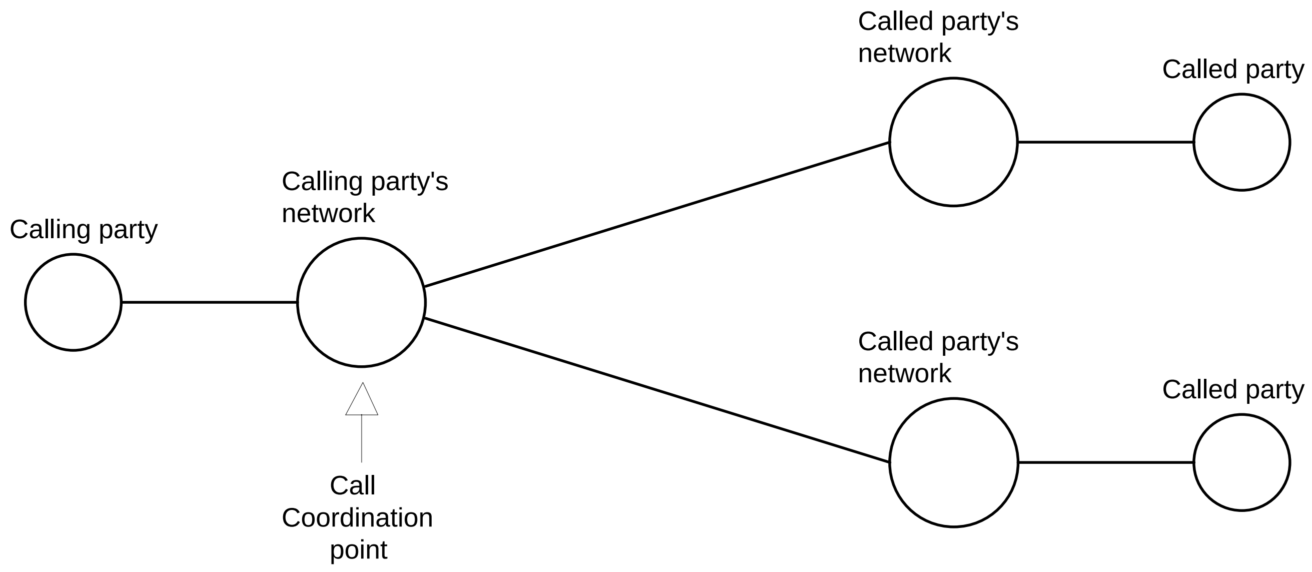

Figure 1 - Multi-party call with two called parties

5.3 CC architecture

CC provides the means of establishing, maintaining and clearing down a call, including the operation of any supplementary services that relate to the call rather than to individual bearers. Control of a call is effected by means of a number of call control entities (CC entities) located in the users' terminals and in various network nodes. Whereas BC involves a bearer control entity (BC entity) at each network node through which a bearer passes, CC requires a CC entity only at those network nodes that provide call-related functionality, e.g., the nodes serving the terminals concerned or nodes that, in the context of the call, provide interworking between networks. In particular, CC entities are not required at nodes that would only provide transit functionality. The precise criteria for determining whether a network node needs to provide CC functionality for a given call are outside the scope of this Standard.

The various CC entities involved in a given call are linked in series by signalling associations. These CC entities and signalling associations are created during call establishment and cleared down when the call is cleared down. The protocol specified in this Standard provides such an association between adjacent CC entities and conveys call-related signalling information between those CC entities. That part of a call between two CC entities that communicate directly via a single signalling association is known as a call segment. This is illustrated in figure 2 for a call that involves four CC entities (e.g., one at each terminal and one at each node serving those terminals) and consequently three call segments.

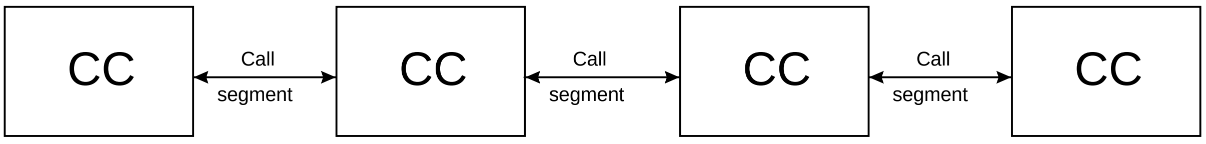

Figure 2 - CC involving four CC entities (three call segments)

5.4 Relationship to BC architecture

BC requires functionality, and hence a BC entity, at each terminal and at every network node through which the bearer is routed. This is in contrast to CC, which involves a CC entity only at the terminals and selected network nodes. Each bearer can be routed independently of other bearers and independently of the routeing of CC signalling associations. However, each bearer is required to be routed through each network node at which there is a CC entity, and hence have a BC entity at each of these nodes, so that the CC entity can manage the bearer if required. This is illustrated in figure 3 for the same call as in figure 2 and a single bearer that has a BC entity collocated with each CC entity and an additional BC entity (e.g. at a transit node) located between the second and third CC entities.

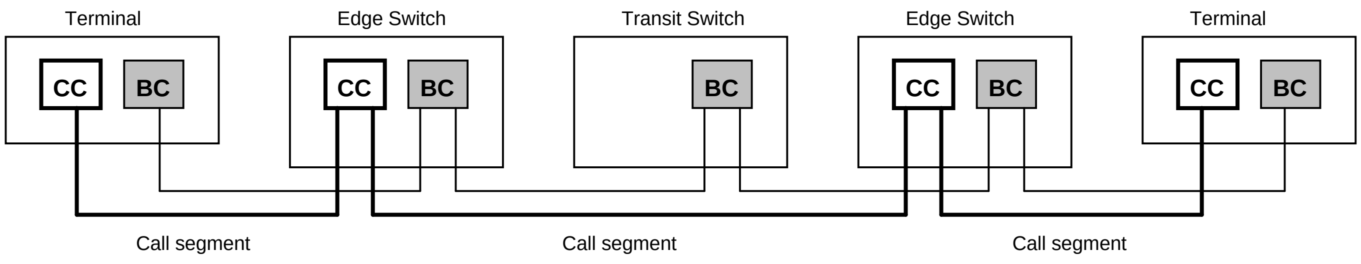

Figure 3 - Relationship of CC and BC architecture

NOTE

Signalling between BC entities is outside the scope of this Standard.

5.5 Screening function

The model for call and BC functional entities (figure 3) shows all CC entities existing at the same location as BC entities. While BC can exist independently of CC, the opposite does not apply. Each CC entity includes bearer co-ordination capabilities.

Although at network boundaries, CC entities will normally be present, network boundaries can also be crossed without provision of a CC entity and the connections associated with those calls can be routed differently from each other and from the call, thereby crossing network boundaries at different locations or even being routed through different networks.

Where a call crosses a network boundary, a number of functions will need to be performed within CC that do not require the presence of a connection. These include:

a) service control. Control of the provision of basic and supplementary services, and subscription arrangements. Identification of correct service profile;

b) translation of numbering plans where the two networks use different numbering plans (e.g. public to private). Even where the same numbering plan is used, the addition of the country code may be necessary;

c) provision of some supplementary services that provide security control on a network basis, e.g. closed user group;

d) support of supplementary services related to numbering (e.g., DDI, MSN) and restriction of numbers (CLIR, COLR).

The functions listed above are outside the scope of this Standard.

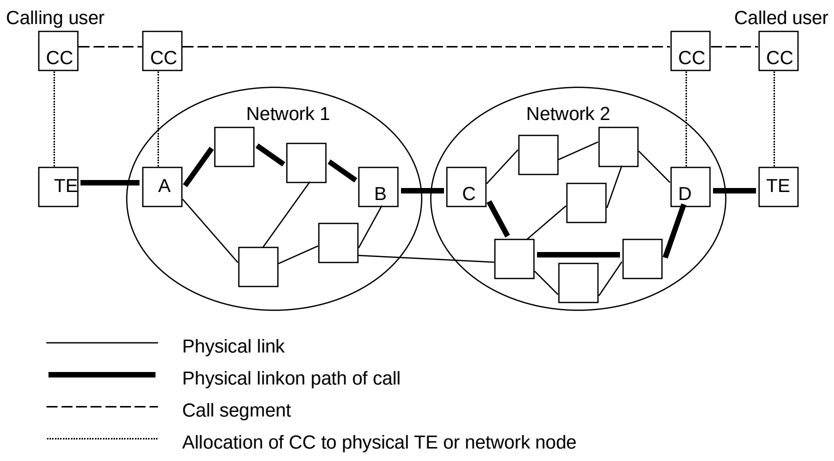

Figure 4 - Call originating and terminating in different networks

Figure 4 shows an example of a call originating and terminating in different networks. If calling line identification restriction (CLIR) applies to the call, this information will be known at the CC at node A, which can add a presentation restricted indicator to the calling party number forwarded on across the next call segment. Normally, on exit from one network to another, a calling party number with an associated presentation restricted indicator is not forwarded, but instead just the presentation restricted indicator is forwarded. The absence of a CC at node B (point of egress to network 2) means that there will be no opportunity to filter out the calling party number until the CC at node D is reached. This is clearly insecure.

Possible alternatives are:

1) Ensure that there is a CC at node B (and similarly at node C to handle this type of situation in the reverse direction). However, CCs are points of bearer coordination, and the presence of CCs at nodes B and C would force all bearers to go via nodes B and C. This would deny the possibility of using other routes between the two networks. For some bearers, the route via nodes B and C might not be the cheapest, or may be congested, or may not provide the desired quality of service. It is desirable to minimize the number of CCs in order to provide maximum flexibility for routeing bearers.

2) Ensure that the CC at node A performs the filtering. However, this requires node A to have knowledge that the call segment leads to another network. This knowledge may not always be available.

To solve the problem without introducing the disadvantages of alternatives 1 and 2, filtering (screening) functionality may be provided at node B (and node C). A functional entity (screening function) may optionally appear between CC entities, and is located as necessary at incoming and outgoing gateways between networks. Figure 5 shows an example.

The screening functional entity has no impact on the information flows, except that this functional entity may impose itself as a transit point on an existing flow.

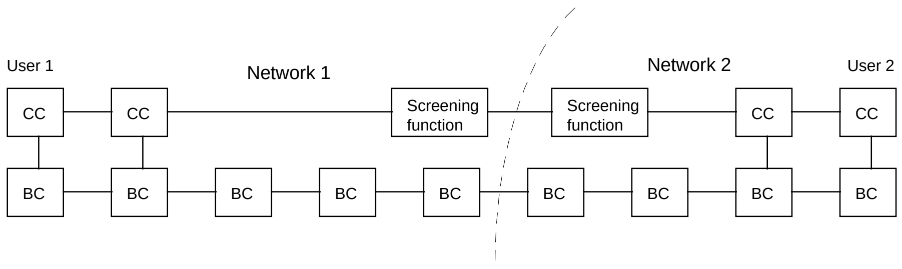

Figure 5 - Screening functional entities at network boundaries

6 Operational requirements

6.1 Provision and withdrawal

The provision of this capability within a network is a network provider option.

The provision of this capability between networks or between a network and a user is by bilateral agreement.

6.2 Transport mechanism

The choice of the underlying transport mechanism within a network is a network provider option.

The choice of the transport mechanism between networks or between a network and a user is by bilateral agreement.

7 Primitive definitions and state definitions

7.1 Service primitives

7.1.1 Service primitive architecture

The following services for call establishment and release are defined:

ESTABLISH-CALL	confirmed
RELEASE-CALL	confirmed

COMPLETE-CALL	unconfirmed
STATUS-CALL	unconfirmed
PROCEED-CALL	unconfirmed
ERROR	indication

Figure 6 shows the architecture which is assumed for two concatenated call segments.

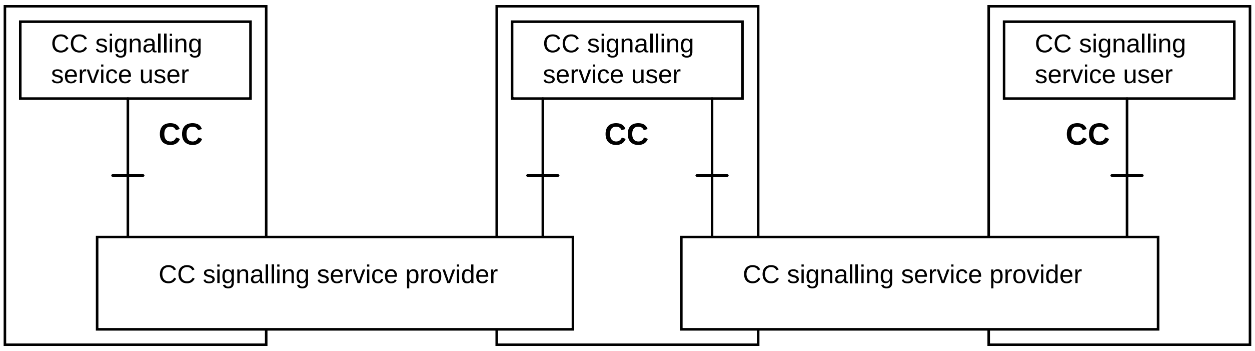
CC signalling service user
CC signalling service user
CC signalling service user
CC
CC
CC
CC signalling service provider
CC signalling service provider

Figure 6 - Architecture

7.1.2 ESTABLISH-CALL

This service is used by the CC signalling service user to establish a call and its information model. It is a confirmed service. Table 1 shows the parameters of the ESTABLISH-CALL primitive.

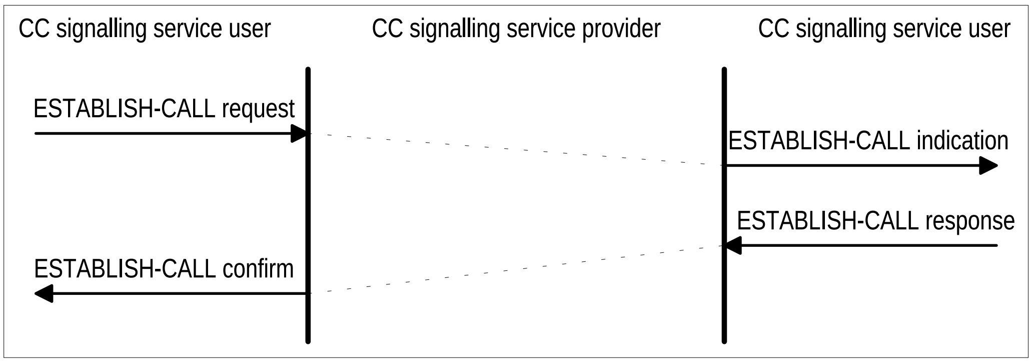
CC signalling service user
CC signalling service provider
CC signalling service user
ESTABLISH-CALL request
ESTABLISH-CALL indication
ESTABLISH-CALL response
ESTABLISH-CALL confirm

Figure 7 - ESTABLISH-CALL service

Table 1 - ESTABLISH-CALL parameters

Parameter name	**request**	**indication**	**response**	**confirmation**
Call segment ID	M	M	M	M
Bearer establishment address	M	M	O	O
Await complete indicator	M	M	-	-
Call description	M	M	O (see note)	O (see note)
Result	-	-	M	M
Diagnostics	-	-	M	M
NOTE: Mandatory if Result is positive, else optional.				

7.1.3 COMPLETE-CALL

This service is used by the CC signalling service user to complete establishment of a call and its information model. It is an unconfirmed service. Table 2 shows the parameters of the COMPLETE-CALL primitive.

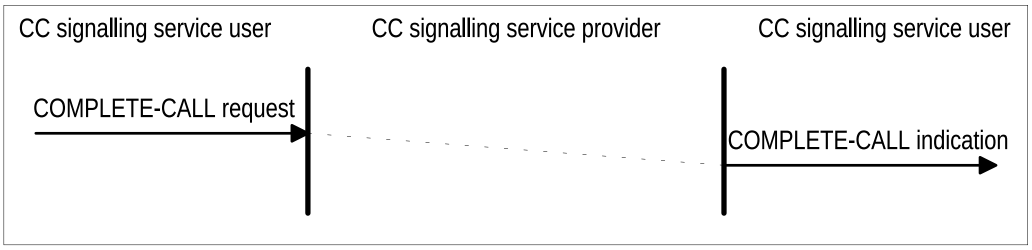
CC signalling service user
CC signalling service provider
CC signalling service user
COMPLETE-CALL request
COMPLETE-CALL indication

Figure 8 - COMPLETE-CALL service

Table 2 - COMPLETE-CALL parameters

Parameter name	**request**	**indication**
Call segment ID	M	M

7.1.4 STATUS-CALL

This service is used by the CC signalling service user to report a change to the information model. It is an unconfirmed service. Table 3 shows the parameters of the STATUS-CALL primitive.

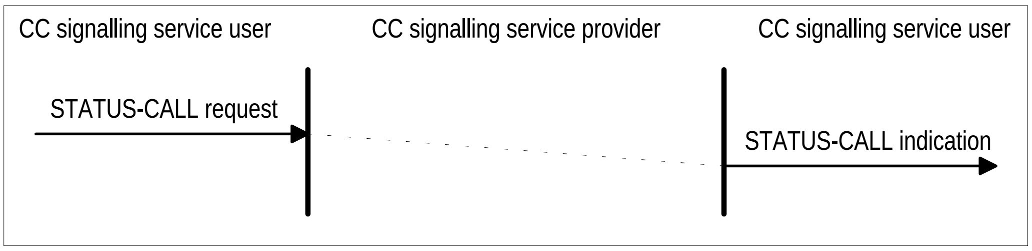
CC signalling service user
CC signalling service provider
CC signalling service user
STATUS-CALL request
STATUS-CALL indication

Figure 9 - STATUS-CALL service

Table 3 - STATUS-CALL parameters

Parameter name	**request**	**indication**
Call segment ID	M	M
Call changed parameter	M	M

7.1.5 RELEASE-CALL

This service is used by the CC signalling service user to release a call and its information model. It is a confirmed service. Table 4 shows the parameters of the RELEASE-CALL primitive.

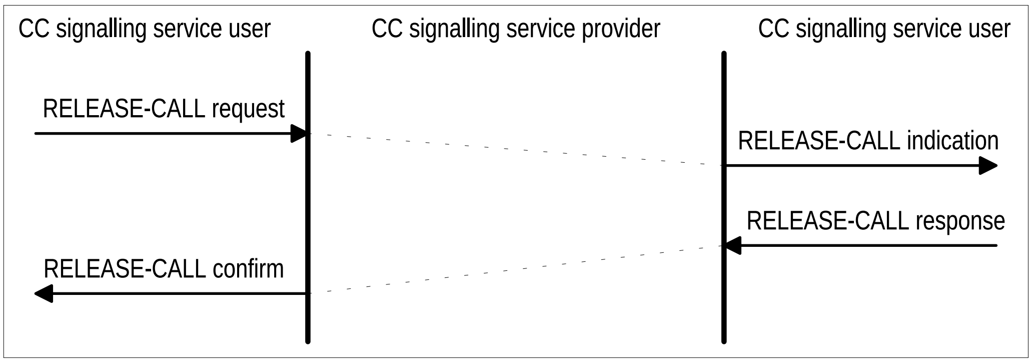

Figure 10 - RELEASE-CALL service

Table 4 - RELEASE-CALL parameters

Parameter name	**request**	**indication**	**response**	**confirmation**
Call segment ID	M	M	M	M
Release cause	M	M	-	-

7.1.6 PROCEED-CALL

This service is used by the CC signalling service user to indicate receipt of a call at the succeeding side of a call segment. It is an unconfirmed service. Table 5 shows the parameters of the PROCEED-CALL primitive.

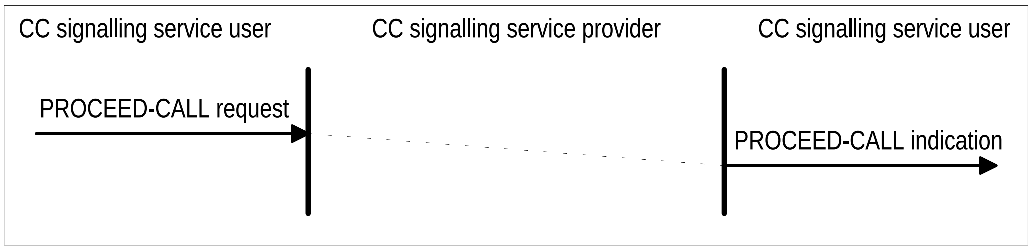
CC signalling service user
CC signalling service provider
CC signalling service user
PROCEED-CALL request
PROCEED-CALL indication

Figure 11 - PROCEED-CALL service

Table 5 - PROCEED-CALL parameters

Parameter name	**request**	**indication**
Call segment ID	M	M
Bearer establishment address	M	M

7.1.7 ERROR

This indication primitive is used by the CC signalling service provider to indicate the occurrence of an exceptional condition to the CC signalling service user. Table 6 shows the parameters of the ERROR indication primitive.

Table 6 - ERROR parameters

Parameter name	indication
Call segment ID	M
Diagnostics	M

7.2 Parameters

7.2.1 Call segment ID

A pair of values which together uniquely identify the call at the two adjacent CC entities bounding a call segment.

7.2.2 Call description

The type CallDescription, as used in the operation callEstablish, contains an information model which describes the properties of a call. The information model comprises sequences of network relevant and end-to-end relevant object descriptions. An object description consists of:

- an object reference, used to refer to particular instances of objects, which should therefore be unique within a particular call description;
- an object status, which is used for example to indicate whether the object is Optional (O) or Mandatory (M);
- an identifier that identifies the class of object to which the instance belongs; and
- an argument of type specific to the class of object to which the instance belongs.

A CC object class is a class of object that inherits the properties of the super class CALLCONTROLOBJECT. Each object class in this super class comprises an identifier and an O argument. The type of the argument depends on the particular object class in the super class, as identified by the identifier. The argument defines the type of attributes of an object of this class.

The OpenCall parameter indicates to the non-call owners in a communications configuration their rights to modify the configuration. The OpenCall parameter is an attribute of the call that is set by the user (CC signalling service user); it is stored in the information model and is transferred in the call description. The CC protocol provides the procedures for exchanging information in an orderly manner. The call status change report provides the mechanism for informing all CC signalling service users involved in the call of one or more changes in the call permissions.

7.2.3 Call changed parameter

The type CallChangedParameter, as used in the operation callStatus, contains a list of those objects of the information model which have been modified and the modification of which has to be reported to the other CC signalling service users involved in the call.

7.2.4 Await complete indicator

Has the value TRUE if call and information model establishment uses a three message sequence or FALSE if call and information model establishment uses an two message sequence.

7.2.5 Bearer establishment address

The address of a network node / terminal to which bearer (connection) establishment shall be routed.

7.2.6 Release cause

Reason for a call release request.

7.2.7 Result

Positive / negative result for an ESTABLISH-CALL request or indication.

7.2.8 Diagnostics

Further explanation of the result (e.g. error values) in an ESTABLISH-CALL response / confirm or the explanation of the exceptional condition that caused an ERROR indication.

7.3 CC states

This chapter describes the states which exist for a CC signalling service provider within a CC entity.

7.3.1 Call Idle

No call exists.

7.3.2 Call Initiated

This state exists at a preceding CC entity when a request for call establishment has been sent to the succeeding CC entity but no response has been received.

7.3.3 Outgoing Call Proceeding

This state exists at a preceding CC entity when acknowledgement that the call is accepted for this call segment has been received from the succeeding CC entity.

7.3.4 Call Ready

This state exists at a preceding CC entity when an indication has been received from the succeeding CC entity that it is ready to complete the establishment of the call and its information model.

7.3.5 Call Present

This state exists at a succeeding CC entity that has not yet responded to the request for call establishment.

7.3.6 Incoming Call Proceeding

This state exists at a succeeding CC entity that has sent to the preceding CC entity acknowledgement that the call is accepted for this call segment.

7.3.7 Await Call Completion

This state exists at a succeeding CC entity that is awaiting an indication from the preceding CC entity that establishment of the call and its information model is to be completed.

7.3.8 Call Active

This state exists at a preceding CC entity that has received an indication that the called user has answered from the succeeding CC entity. This state exists at a succeeding CC entity that has sent an indication that the called user has answered to the preceding CC entity.

7.3.9 Call Release Request

This state exists when a CC entity has sent out a request for call release but a response has not yet been received.

7.3.10 Call Release Indication

This state exists when a CC entity has received a call release indication but the user has not yet answered.

8 Coding requirements

8.1 Abstract definition of the CC operations

Table 7 shows the definition of the operations, errors and types required for the CC protocol using Abstract Syntax Notation One (ASN.1) as defined in ITU-T Recommendations X.680, X.681, X.682, X.683 and using the OPERATION and ERROR object classes as defined in ITU-T Rec. X.880.

Application Protocol Data Units (APDUs) based on these operations shall be of types invoke, returnResult, returnError and reject as defined in table B.1 of ETS 300 796-1. The Basic Encoding Rules (BER) as defined in ITU-T Rec. X.690 shall be applied to the encoding of APDUs based on these operations and errors.

Table 7 - Definition of operations for the CC protocol

```
CC-Operations   { itu-t recommendation q 2981  cc-operations (1) }
DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN
EXPORTS CcOperations, CallSegmentId;
IMPORTS OPERATION, ERROR FROM Remote-Operations-Information-Objects
                { joint-iso-itu-t (2) remote-operations (4) informationObjects (5) version1 (0) }
        PartyNumber FROM Addressing-Data-Elements
                { itu-t recommendation q 932 addressing-data-elements (7) }
        -- The definition of PartyNumber is reproduced in Annex F
        CALLCONTROLOBJECTCLASS FROM Call-Control-Object-Super-Class
                { itu-t recommendation q 2981  call-control-object-super-class (4) }
        -- The definition of CALLCONTROLOBJECTCLASS is given in subclause 8.2
        call, localPartyEP, remotePartyEP, directCallAssociation, remoteCallAssociation, serviceComponent
        FROM Call-Object-Class-Definitions
                { itu-t recommendation q 2981  call-object-class-definitions (5) } ;
        -- The definition of call, localPartyEP, remotePartyEP, directCallAssociation,
remoteCallAssociation,
        -- serviceComponent is given in subclause 8.3

CcOperations    OPERATION  ::=   { callEstablish | callProceeding | callComplete | callRelease |
callStatus }

ccOperationsDefinitions  OBJECT IDENTIFIER ::= { itu-t recommendation q 2981  cc-operations-definitions
(2) }

-- The callEstablish operation is used to establish a call and its information model. It is a confirmed
operation.
callEstablish   OPERATION  ::=   {
        ARGUMENT    SEQUENCE
                          { callSegmentId       CallSegmentId,
                    callDescription     CallDescription,
                    bearerEstablAddress BearerEstablishmentAddress,
                    awaitCompleteIndicator  BOOLEAN,
                    parameterActionIndicator    ParameterActionIndicator,
                    ...  }
        RESULT      SEQUENCE
                          { callSegmentId       CallSegmentId,
                    callDescription     CallDescription,
                    parameterActionIndicator    ParameterActionIndicator,
                    bearerEstablAddress BearerEstablishmentAddress  OPTIONAL,
                    ...  }
        ERRORS  { callDescriptionNotAccepted | unallocatedNumber | noUserResponding |
                  noAnswerFromUser | callRejected | destinationOutOfOrder |
                  addressIncomplete | networkOutOfOrder | temporaryFailure |
                  userBusy | userNotReachable | unspecified }
        CODE        global :    { ccOperationsDefinitions 1 }   }

-- The callProceeding operation is used by the succeeding call control entity to inform the preceding call
control entity
-- that the call is in progress and connection establishment may start for this segment. It is an
unconfirmed operation.
callProceeding  OPERATION  ::=   {
        ARGUMENT    SEQUENCE
                          { callSegmentId       CallSegmentId,
                    bearerEstablAddress BearerEstablishmentAddress,
                    parameterActionIndicator    ParameterActionIndicator,
                    ...  }
        RETURN RESULT   FALSE
        ALWAYS RESPONDS FALSE
        CODE        global :    { ccOperationsDefinitions 2 }   }

-- The callRelease operation is used to release an existing call and its information model. It is a
confirmed operation.
callRelease OPERATION  ::=   {
        ARGUMENT    SEQUENCE
                          { callSegmentId   CallSegmentId,
                    releaseCause    ReleaseCause,
                    parameterActionIndicator    ParameterActionIndicator,
                    ...  }
        RESULT      SEQUENCE
                          { callSegmentId   CallSegmentId ,
                    parameterActionIndicator    ParameterActionIndicator,
                    ...  }
        CODE        global :    { ccOperationsDefinitions 3 }   }
```

```
-- The callComplete operation is used to indicate completion of establishment of a call and its
information model.
-- It is an unconfirmed operation.
callComplete    OPERATION  ::=  {
        ARGUMENT    SEQUENCE
                          { callSegmentId   CallSegmentId,
                    parameterActionIndicator    ParameterActionIndicator,
                    ...  }
        RETURN RESULT   FALSE
        ALWAYS RESPONDS FALSE
        CODE        global :    { ccOperationsDefinitions 4 }   }

-- The callStatus operation is used to report a change to the information model. It is an unconfirmed
operation.
callStatus  OPERATION  ::=  {
        ARGUMENT    SEQUENCE
                          { callSegmentId   CallSegmentId,
                    callChangedParameter    SEQUENCE OF CallChangedParameter,
                    parameterActionIndicator    ParameterActionIndicator,
                    ...  }
        RETURN RESULT   FALSE
        ALWAYS RESPONDS FALSE
        CODE        global :    { ccOperationsDefinitions 5 }   }

ParameterActionIndicator    ::= ENUMERATED  { clearCallAndItsInformationModel (0),
                          discardApduAndReject (1),
                          discardApduNoReject (2),
                          discardParameterAndPassApduToApplication (3),
                          ignoreParameterAndPassApduToApplication (4) }
    -- Used to indicate action to be taken if a parameter in an operation is not recognised


BearerEstablishmentAddress ::= PartyNumber

NetworkRelevantObjectClassSet CALLCONTROLOBJECTCLASS  ::=  { call | localPartyEP | remotePartyEP |
                                directCallAssociation | remoteCallAssociation , ... }
EndToEndRelevantObjectClassSet CALLCONTROLOBJECTCLASS  ::=  { serviceComponent , ... }

CallDescription  ::=  SEQUENCE {
        networkRelevantPart     SEQUENCE OF
NetworkRelevantObjectDescription{{NetworkRelevantObjectClassSet}},
        endToEndRelevantPart    SEQUENCE OF
EndToEndRelevantObjectDescription{{EndToEndRelevantObjectClassSet}} OPTIONAL }

NetworkRelevantObjectDescription {CALLCONTROLOBJECTCLASS: NetworkRelevantObjectClassSet}  ::=
SEQUENCE {
    objectReference INTEGER,
    objectActionInd ObjectActionIndicator,
    objectStatus    ObjectStatus,
    objectClassId   CALLCONTROLOBJECTCLASS.&objectClassIdentifier ({NetworkRelevantObjectClassSet}),
    objectArgument  CALLCONTROLOBJECTCLASS.&ArgumentType ({NetworkRelevantObjectClassSet}
                                        {@objectClassId}) OPTIONAL,
    ... }

EndToEndRelevantObjectDescription {CALLCONTROLOBJECTCLASS: EndToEndRelevantObjectClassSet} ::=
SEQUENCE {
    objectReference INTEGER,
    objectActionInd ObjectActionIndicator,
    objectStatus    ObjectStatus,
    objectClassId   CALLCONTROLOBJECTCLASS.&objectClassIdentifier ({EndToEndRelevantObjectClassSet}),
    objectArgument  CALLCONTROLOBJECTCLASS.&ArgumentType ({EndToEndRelevantObjectClassSet}
                                        {@objectClassId}) OPTIONAL,
    ... }

CallChangedParameter  ::= SEQUENCE {
    modifiedNetworkRelevantPart SEQUENCE OF
ModifiedNetworkRelevantObjectDescription{{NetworkRelevantObjectClassSet}},
    modifiedEndToEndRelevantPart    SEQUENCE OF
ModifiedEndToEndRelevantObjectDescription{{EndToEndRelevantObjectClassSet}} OPTIONAL }

ModifiedNetworkRelevantObjectDescription {CALLCONTROLOBJECTCLASS: NetworkRelevantObjectClassSet}  ::=
SEQUENCE {
    operation       ENUMERATED {deleteObject (0), modifyAttributes (1), ... },
    objectReference     INTEGER,
    objectActionInd     ObjectActionIndicator,
    modifiedArgument    CALLCONTROLOBJECTCLASS.&ArgumentType({NetworkRelevantObjectClassSet})   OPTIONAL }
```

```
ModifiedEndToEndRelevantObjectDescription {CALLCONTROLOBJECTCLASS: EndToEndRelevantObjectClassSet}  ::=
SEQUENCE {
    operation       ENUMERATED {deleteObject (0), modifyAttributes (1), ... },
    objectReference     INTEGER,
    objectActionInd     ObjectActionIndicator,
    modifiedArgument    CALLCONTROLOBJECTCLASS.&ArgumentType({EndToEndRelevantObjectClassSet})  OPTIONAL }

ObjectActionIndicator   ::= ENUMERATED  { clearCall (0),
                          discardNotify (1),
                          discardUnknown (2),
                          progressTransit (3), ... }
    -- Used to indicate action to be taken if an object or object attribute is not recognised

ObjectStatus ::= ENUMERATED {
    mandatory(0),
    optional(1),
    conditional(2),
    ... }

CallSegmentId ::= SEQUENCE  { precedingSideCallSegId        CallSegmentIdComponent,
                  succeedingSideCallSegId   CallSegmentIdComponent }

CallSegmentIdComponent ::= INTEGER (-2147483648 .. 2147483647)  -- 4 octets
    -- The value 0 is to be used as a null value for the succeeding side call segment identifier
    -- in the callEstablish invoke APDU.

ReleaseCause ::= SEQUENCE   { causeValue    CauseValue,
                  location  Location,
                  ... }

CauseValue ::= ENUMERATED   { callDescriptionNotAccepted (0),
                  normalCallClearing (3),
                  temporaryFailure (11),
                  recoveryOnTimerExpiry (12),
                  unspecified (4),
                  ... }

Location ::= ENUMERATED { unspecified (0),
                  user (1),
                  networkLocalCallSegment (2),
                  networkNonLocalCallSegment (3),
                  ... }

ccOperationsErrors  OBJECT IDENTIFIER ::= { itu-t recommendation q 2981  cc-operations-errors (3) }

callDescriptionNotAccepted  ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location
Location,
                                    callDescription CallDescription  OPTIONAL,
                                    ...  }
                            CODE  global : { ccOperationsErrors 1 }  }
userBusy        ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location,
                                 callDescription CallDescription  OPTIONAL,
                                 ...  }
                            CODE  global : { ccOperationsErrors 2 }  }
unallocatedNumber   ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location,
... }
                            CODE  global : { ccOperationsErrors 3 }  }
noUserResponding    ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location ,
... }
                            CODE  global : { ccOperationsErrors 4 }  }
noAnswerFromUser    ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location ,
... }
                            CODE  global : { ccOperationsErrors 5 }  }
callRejected        ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location ,
... }
                            CODE  global : { ccOperationsErrors 6 }  }
destinationOutOfOrder   ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location
Location , ... }
                            CODE  global : { ccOperationsErrors 7 }  }
addressIncomplete   ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location ,
... }
                            CODE  global : { ccOperationsErrors 8 }  }
networkOutOfOrder   ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location ,
... }
                            CODE  global : { ccOperationsErrors 9 }  }
temporaryFailure        ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location
Location , ... }
                            CODE  global : { ccOperationsErrors 10 }  }
```

```
userNotReachable    ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location ,
... }
                            CODE  global : { ccOperationsErrors 11 }  }
unspecified     ERROR  ::=  { PARAMETER  SEQUENCE  { callSegmentId CallSegmentId, location Location , ...
}
                            CODE  global : { ccOperationsErrors 12 }  }

END
```

8.2 Definition of CC Object Super Class

Table 8 shows the definition of the CC Object Super Class using ASN.1 as defined in ITU-T Recommendations X.680, X.681, X.682, X.683.

Table 8 - Definition of CC Object Super Class

```
Call-Control-Object-Super-Class
    { itu-t recommendation q 2981  call-control-object-super-class (4) }

DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN
EXPORTS CALLCONTROLOBJECTCLASS;

CALLCONTROLOBJECTCLASS ::= CLASS
{   &ArgumentType           OPTIONAL,
    &argumentTypeOptional       BOOLEAN OPTIONAL,
    &objectClassIdentifier      OBJECT IDENTIFIER UNIQUE    }

WITH SYNTAX
{   [ARGUMENT   &ArgumentType   [OPTIONAL  &argumentTypeOptional] ]
    IDENTIFIER  &objectClassIdentifier  }

END -- Call-Control-Object-Super-Class
```

8.3 Definitions of CC Object Classes in the Information Model

CC object classes are inherited from the super class CALLCONTROLOBJECTCLASS, used for the purpose of object descriptions within the call description. For each object class, the ARGUMENT type and unique IDENTIFIER are defined.

Table 9 shows the definition of the CC Object Classes in the Information Model described in DEG/SPS-05134 using ASN.1 as defined in ITU-T Recommendations X.680, X.681, X.682, X.683.

Table 9 - Definition of CC Object Classes in the Information Model

```
Call-Object-Class-Definitions
    { itu-t recommendation q 2981  call-object-class-definitions (5) }

DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN

EXPORTS
call, localPartyEP, remotePartyEP, directCallAssociation, remoteCallAssociation,
serviceComponent,BearerId;

IMPORTS
CALLCONTROLOBJECTCLASS FROM Call-Control-Object-Super-Class
    { itu-t recommendation q 2981  call-control-object-super-class (4) }
PresentedAddressScreened, PartyNumber, PartySubaddress FROM Addressing-Data-Elements
    { itu-t recommendation q 932 addressing-data-elements (7) }  ;

ccObjectClasses  OBJECT IDENTIFIER ::= { itu-t recommendation q 2981  cc-object-classes (6) }
```

```
call CALLCONTROLOBJECTCLASS ::= {
    ARGUMENT    SEQUENCE {
                localPEPId      ObjectReferenceId,
                remotePEPId     ObjectReferenceId,
                serviceReference    ObjectReferenceId  OPTIONAL,
                directCallAssociationIds    ObjectReferenceIdList,
                remoteCallAssociationIds    ObjectReferenceIdList  OPTIONAL,
                bearerIdList        BearerIdList  OPTIONAL,
                telecomsServiceType TelecomsServiceType,
                callPermissions     OpenCall
                     }
    IDENTIFIER  { ccObjectClasses 1 }
              }

ObjectReferenceId ::= INTEGER (-2147483648 .. 2147483647)    -- 4 octets
    -- refers to an object reference, unique to each object within a call

ObjectReferenceIdList ::= SEQUENCE OF ObjectReferenceId

BearerIdList ::= SEQUENCE OF BearerId
BearerId ::= OCTET STRING (SIZE (1..3))

OpenCall ::= BIT STRING {
    reserved(7), externalPartyAddAllowed(6), existingPartyAddAllowed(5), notifyAllPartiesFlag(4),
    notifyOwnerFlag(3), permissionRequiredFlag(2), addConnectionAllowed(1),
    addServiceComponentAllowed(0)   }

TelecomsServiceType ::= ENUMERATED {
        realtimeMultiMedia(0),
        nonRealtimeMultiMedia(1),
        undefined(2),
        ... }

localPartyEP CALLCONTROLOBJECTCLASS ::= {
    ARGUMENT    PartyObjectArgument
    IDENTIFIER  { ccObjectClasses 2 }
                  }

remotePartyEP CALLCONTROLOBJECTCLASS ::= {
    ARGUMENT    PartyObjectArgument
    IDENTIFIER  { ccObjectClasses 3 }
                  }

PartyObjectArgument ::= SEQUENCE {
    partyAddress    SEQUENCE {
                presentedAddressScreened    PresentedAddressScreened,
                defaultAddress              DefaultAddress  OPTIONAL,
                networkInternalAddress      NetworkInternalAddress  OPTIONAL
                    },
    partyOwnerPEPId     ObjectReferenceId,
    associatedResourcePEPIds    ObjectReferenceIdList OPTIONAL,
    associatedPEPIds        ObjectReferenceIdList OPTIONAL,
    partyType       ENUMERATED {initiator(0), receiver(1), callOwner(2), …},
    partyStatus     ENUMERATED {confirmed (0), virtual(1), alerting(2), ...}
                }
DefaultAddress ::= OCTET STRING (SIZE (1..21))
NetworkInternalAddress ::= OCTET STRING (SIZE (1..21))

directCallAssociation CALLCONTROLOBJECTCLASS ::= {
    ARGUMENT    SEQUENCE { remotePEPId  ObjectReferenceId }
    IDENTIFIER  { ccObjectClasses 4 }
                  }

remoteCallAssociation CALLCONTROLOBJECTCLASS ::= {
    ARGUMENT    SEQUENCE {
                localPEPId  ObjectReferenceId,
                remotePEPId ObjectReferenceId
                }
    IDENTIFIER  { ccObjectClasses 5 }
                  }
```

```
serviceComponent CALLCONTROLOBJECTCLASS ::= {
    ARGUMENT
            SEQUENCE {
        callPEPId                               ObjectReferenceId,
        serviceComponentCharacteristics         ServiceComponentCharacteristics  OPTIONAL,
        communicationConfiguration              CommunicationConfiguration  OPTIONAL,
        serviceTrafficDescriptorRequirements    ServiceTrafficDescriptorRequirements  OPTIONAL,
        serviceComponentQoSRequirements         ServiceQoSRequirements  OPTIONAL,
        associatedServiceModuleId               ObjectReferenceId  OPTIONAL,
        associatedResourceComponentId           ObjectReferenceId  OPTIONAL
            }
    IDENTIFIER  { ccObjectClasses 6 }
                  }

ServiceComponentCharacteristics ::= OCTET STRING
CommunicationConfiguration ::= ENUMERATED {source(0), sink(1), biDirectional(2), …}
ServiceTrafficDescriptorRequirements ::= OCTET STRING
ServiceQoSRequirements ::= OCTET STRING

END
```

9 Procedures

9.1 Call establishment request

9.1.1 Preceding CC entity

On receipt of an ESTABLISH-CALL request primitive from the CC signalling service user, the preceding CC entity shall initiate call establishment by sending a callEstablish invoke APDU towards the succeeding CC entity and start timer T703. Following the transmission of the APDU, the preceding CC entity shall enter the Call Initiated state.

The preceding CC entity shall include element awaitCompleteIndicator in the callEstablish invoke APDU reflecting the value (TRUE or FALSE) of the Await Complete Indicator parameter within the ESTABLISH-CALL request primitive and store this value locally as "Await Complete Indicator".

The callEstablish invoke APDU shall contain the CallSegmentId with the precedingSideCallSegId set to the value provided by the CC signalling service user and the succeedingSideCallSegId set to the null value. Within the argument of the callEstablish invoke APDU, element bearerEstablishmentAddress shall contain an address which, when used as a destination address for bearer establishment in the backward direction from the subsequent CC entity, causes the bearer to be routed to the preceding CC entity's terminal or network node, i.e., to the terminal or network node at the start of the call segment.

The preceding CC entity shall include element callDescription in the callEstablish invoke APDU, as provided by the CC signalling service user.

Prior to sending the callEstablish invoke APDU towards the succeeding CC entity, the preceding CC entity shall initiate establishment of a transport mechanism connection to the succeeding CC entity or use an existing transport mechanism (e.g. an already existing connection or a permanently available transport mechanism).

NOTE

The CC protocol is independent of the underlying transport mechanism. It is therefore out of scope of this Standard which transport mechanism is used.

9.1.2 Succeeding CC entity

On receipt of a callEstablish invoke APDU, the succeeding CC entity shall enter the Call Present state.

The succeeding CC entity shall store the contents of element awaitCompleteIndicator as "Await Complete Indicator".

The receipt of the callEstablish invoke APDU is indicated to the CC signalling service user with an ESTABLISH-CALL indication primitive.

9.2 Call Proceeding

9.2.1 Preceding CC entity

On receipt of a callProceeding invoke APDU while in state Call Initiated, the preceding CC entity shall stop timer T703, start timer T710, and enter the Outgoing Call Proceeding state.

The receipt of the callProceeding invoke APDU is indicated to the CC signalling service user with a PROCEED-CALL indication primitive.

NOTE

This is the earliest time at which the CC signalling service user can establish bearer connections across this segment.

9.2.2 Succeeding CC entity

On request of the CC signalling service user (PROCEED-CALL request primitive) and while in state Call Present, the succeeding CC entity shall send a callProceeding invoke APDU to the preceding CC entity and enter the Incoming Call Proceeding state.

The callProceeding invoke APDU shall contain the element callSegmentId with the precedingSideCallSegId set to the value as received in the callEstablish invoke APDU and the succeedingSideCallSegId set to the value provided by the CC signalling service user.

Within the argument of the callProceeding invoke APDU, element bearerEstablishmentAddress shall contain an address which, when used as a destination address for bearer establishment in the forward direction from the preceding CC entity, causes the bearer to be routed to the succeeding CC entity's terminal or network node, i.e., to the terminal or network node at the end of the call segment.

NOTE 1

This is the earliest time at which the CC signalling service user can establish bearer connections across this segment.

NOTE 2

A bearer establishment can be initiated from either the call originating user or from the call destination user.

9.3 Call accepted

9.3.1 Preceding CC entity

On receipt of the callEstablish return result APDU while in state Call Initiated or Outgoing Call Proceeding, the preceding CC entity shall stop timer T703 or T710 and:

- if the value of stored item "Await Complete Indicator" is TRUE, then enter state Call Ready;
- if the value of stored item "Await Complete Indicator" is FALSE, then enter state Call Active.

The receipt of the callEstablish return result APDU is indicated to the CC signalling service user with an ESTABLISH-CALL confirm primitive.

NOTE

If the callEstablish return result APDU was the first response received to a callEstablish invoke, this is the earliest time at which the CC signalling service user can establish bearer connections across this segment.

9.3.2 Succeeding CC entity

On request of the CC signalling service user (ESTABLISH-CALL response primitive), the succeeding CC entity shall send a callEstablish return result APDU to the preceding CC entity and:

- if the value of stored item "Await Complete Indicator" is TRUE, then enter state Await Call Completion and start timer T701;
- if the value of stored item "Await Complete Indicator" is FALSE, then enter state Call Active.

The callEstablish return result APDU shall contain the element callSegmentId. If the succeeding CC entity has not previously sent a callProceeding invoke APDU, the argument of the callEstablish return result APDU shall contain element CallSegmentId with the precedingSideCallSegId set to the value as received in the callEstablish invoke APDU and the succeedingSideCallSegId set to the value provided by the CC signalling service user. If it has previously sent a callProceeding invoke APDU, the argument of the callEstablish return result APDU shall contain element CallSegmentId as indicated by the CC signalling service user.

The callEstablish return result APDU shall contain the element CallDescription, as provided by the CC signalling service user in the ESTABLISH-CALL response primitive.

If the succeeding CC entity has not previously sent a callProceeding invoke APDU, the argument of the callEstablish return result APDU shall contain element bearerEstablishmentAddress. This element shall contain an address which, when used as a destination address for bearer establishment in the forward direction from the preceding CC entity, causes the bearer to be routed to the succeeding CC entity's terminal or network node, i.e., to the terminal or network node at the end of the call segment.

NOTE

This is the earliest time at which the CC signalling service user can establish bearer connections across this segment.

9.4 Completion of call establishment

9.4.1 Preceding CC entity

On receipt of a Complete-Call request primitive while in state Call Ready, the preceding CC entity shall send a callComplete invoke APDU to the succeeding CC entity and enter state Call Active.

The preceding CC entity shall include element callSegmentId in the callComplete invoke APDU.

9.4.2 Succeeding CC entity

On receipt of a callComplete invoke APDU in state Await Call Completion, the succeeding CC entity shall enter state Call Active, stop timer T701 and give a Complete-Call indication primitive to the CC signalling service user.

9.5 Call status change report

9.5.1 General

A change of the information model of a call is reflected in the Call Description belonging to that call. In order to report such a change to the peer CC entity, the callStatus operation, which contains the element callChangedParameter, shall be used. This element indicates changes to the original call description:

- addition and deletion of a service component object;
- changes to the attributes of a party object (i.e. modification of the status of a party);
- changes to the OpenCall attribute of the call object (i.e. modification of the call permissions).

A call status change report indicates one or more changes, but any deleted object shall not be referenced in a modified attribute.

9.5.2 Initiating CC entity

On receipt of a Status-Call request primitive while in state Await Call Completion or Call Active, the initiating CC entity shall send a callStatus invoke APDU and remain in the same state.

The initiating CC entity shall include elements callSegmentId and callChangedParameter in the callStatus invoke APDU as indicated by the CC signalling service user.

The element callChangedParameter contains a list of the changed objects such that:

- where an object specified in the original Call Description is deleted, the value "deleteObject" shall be used, the objectReference shall be included, and modifiedArgument shall not be included;
- where an object specified in the original Call Description is modified, the value "modifyAttributes" shall be used.

9.5.3 Receiving CC entity

On receipt of a callStatus invoke APDU in state Call Ready or Call Active, the receiving CC entity shall give a Status-Call indication primitive to the CC signalling service user and remain in the same state.

9.6 Call establishment failure

9.6.1 Preceding CC entity

On receipt of a callEstablish return error APDU while in state Call Initiated or Outgoing Call Proceeding, the preceding CC entity shall stop all timers (if timers are running) and enter the Call Idle state.

Call establishment failure is indicated to the CC signalling service user by an ESTABLISH-CALL confirm primitive with negative Result parameter.

NOTE

The CC signalling service user gives an indication to BC that the bearer connections for this call shall be cleared.

Call establishment failure can also be indicated from the preceding CC entity while in state Call Ready to the succeeding CC entity by the use of the procedures for call clearing as defined in 9.7.

9.6.2 Succeeding CC entity

On CC signalling service user request (ESTABLISH-CALL response primitive with negative Result parameter) and while in state Call Present or Incoming Call Proceeding, the succeeding CC entity shall send a callEstablish return error APDU with a suitable error value and enter the Call Idle state.

Suitable error values are:

- callDescriptionNotAccepted, if the received call description was not accepted by the CC signalling service user. In this case an alternative call description may be returned together with the error value;
- userBusy, if the called user is busy. In this case, if the received call description was not acceptable by the CC signalling service user, an alternative call description may be returned as additional information together with the error value;
- unallocatedNumber, if the received call description contained an unallocated number;
- noUserResponding, if the called user did not respond to the callEstablish invoke APDU;
- noAnswerFromUser, if no answer from the called user to the callEstablish invoke APDU was received;
- callRejected, if the called user rejected the call;
- destinationOutOfOrder, if the called user's equipment is out of order;
- addressIncomplete, if an address contained in the received call description was incomplete;
- networkOutOfOrder, if equipment in the network is out of order;
- temporaryFailure, if a temporary failure has occurred;
- userNotReachable, if the called user is not reachable;
- unspecified in any other case.

When sending the callEstablish return error APDU, the Location parameter shall indicate the location at which the failure occurred. When an Error is first generated in a user's terminal, the location value 'user' shall be used. When an Error is first generated in a network node, the location value 'networkLocalCallSegment' shall be used. If that Error is passed on by a CC entity to another call segment the value shall be changed to 'networkNonLocalCallSegment'.

The callEstablish return error APDU may contain a Call Description element if provided by the CC signalling service user.

NOTE

The CC signalling service user gives an indication to BC that the bearer connections for this call shall be cleared.

Call establishment failure can also be indicated from the preceding CC entity to the succeeding CC entity while in state Await Call Completion by the use of the procedures for call clearing as defined in 9.7.

9.7 Call clearing

9.7.1 Procedures at the CC entity that initiates call clearing

On receipt of a RELEASE-CALL request primitive from the CC signalling service user, the CC entity initiating call clearing shall send a callRelease invoke APDU, stop all timers, start timer T708, and enter the Call Release Request state.

The argument of the callRelease invoke APDU shall contain a causeValue and the appropriate location parameter in element releaseCause. Valid cause values are 'normalCallClearing', 'callDescriptionNotAccepted', or 'temporaryFailure', depending on the reason why the CC signalling service user initiates call clearing.

Cause values shall be used as follows:

- normalCallClearing, if call release was initiated by one of the involved users;
- callDescriptionNotAccepted, if the received call description was not accepted by the CC signalling service user;
- temporaryFailure, if a temporary failure has occurred.

The location parameter shall indicate the location at which the failure occurred. When a releaseCause is first generated in a user's terminal, the location value 'user' shall be used. When a releaseCause is first generated in a network node, the location value 'networkLocalCallSegment' shall be used. If that releaseCause is passed on by a CC entity to another call segment the value shall be changed to 'networkNonLocalCallSegment'.

The callRelease invoke APDU shall only be sent with a complete callSegmentId. Therefore, a CC entity which has initiated a call establishment, i.e. has sent a callEstablish invoke APDU, shall not initiate call clearing before either a callEstablish return result or callProceeding invoke APDU has been received.

On receipt of the callRelease return result APDU while in state Call Release Request, the CC entity initiating clearing shall stop timer T708 and enter the Call Idle state.

The receipt of the callRelease return result APDU is indicated to the CC signalling service user with a RELEASE-CALL confirm primitive.

NOTE

When sending the RELEASE-CALL request primitive to the CC signalling service provider, the CC signalling service user also requests BC that the bearer connections for this call shall be cleared.

9.7.2 Procedures at the CC entity that responds to clearing

In any state except Call Idle, Call Release Request, and Call Release Indication the CC entity receiving a callRelease invoke APDU shall stop any timers that are running and enter the Call Release Indication state. The receipt of the callRelease invoke APDU is indicated to the CC signalling service user with a RELEASE-CALL indication primitive.

On CC signalling service user request (RELEASE-CALL response primitive) the receiving CC entity shall send a callRelease return result APDU and return to the Call Idle state.

NOTE

When receiving the RELEASE-CALL indication primitive, the CC signalling service user requests BC that the bearer connections for this call shall be cleared.

9.7.3 Call clearing collision

In case of call clearing collision, i.e. on receipt of a callRelease invoke APDU while in state Call Release Request, the CC entity shall stop timer T708 and enter the Call Idle state.

9.8 Exceptional Procedures

9.8.1 Timer Expiry

9.8.1.1 Procedures at the preceding CC entity

If timer T703 expires, i.e. if no response to the callEstablish invoke APDU is received, the preceding CC entity shall clear the call internally, release the CallSegmentId, and enter the Call Idle state.

Call clearing is indicated to the CC signalling service user by an ESTABLISH-CALL confirm primitive with negative Result parameter.

If timer T710 expires, i.e. if a callProceeding invoke APDU has been received as a response to the callEstablish invoke APDU, but no callEstablish return result or return error APDU is received, the preceding CC entity shall clear the call by sending a callRelease invoke APDU, stop all timers, start timer T708, and enter the Call Release Request state. The callRelease invoke APDU sent to initiate call clearing shall contain causeValue 'recovery on timer expiry' and an appropriate location value in element releaseCause.

Call clearing is indicated to the CC signalling service user by an ESTABLISH-CALL confirm primitive with negative Result parameter.

9.8.1.2 Procedures at the succeeding CC entity

If timer T701 expires, i.e. if no callComplete invoke APDU is received in the Await Call Completion state, the succeeding CC entity shall clear the call internally, release the CallSegmentId, and enter the Call Idle state.

This exceptional condition is indicated to the CC signalling service user by an ERROR indication primitive with an appropriate Diagnostics parameter.

9.8.1.3 Procedures at the CC entity that initiates call clearing

If timer T708 expires, i.e. if no response to the callRelease invoke APDU is received, the CC entity shall release the call segment ID and enter the Call Idle state.

A RELEASE-CALL confirm primitive is given to the CC signalling service user if call clearing has been initiated on user request.

9.8.2 Receipt of APDUs with unknown Call Segment Id

If an APDU other than a callEstablish invoke is received containing an unknown CallSegmentId, this APDU shall be ignored.

9.8.3 Receipt of APDUs with duplicated Call Segment Id

If a callEstablish invoke APDU is received containing a CallSegmentId which is already in use, this APDU shall be ignored.

9.8.4 Receipt of APDUs out of sequence

If an APDU out of sequence is received, i.e. a callProceeding invoke after a callEstablish return result, with the same CallSegmentId, this APDU shall be ignored.

9.8.5 Receipt of Reject APDUs

9.8.5.1 Receipt of a reject APDU that is correlated to a callEstablish invoke

On receipt of a reject APDU that is correlated to a callEstablish invoke APDU while in state Call Initiated, the preceding CC entity shall stop timer T703, clear the call internally, release the CallSegmentId, and enter the Call Idle state.

Call establishment failure is indicated to the CC signalling service user by an ESTABLISH-CALL confirm primitive with negative Result parameter.

9.8.5.2 Receipt of a reject APDU that is correlated to a callProceeding invoke

On receipt of a reject APDU that is correlated to a callProceeding invoke APDU while in state Incoming Call Proceeding, the succeeding CC entity shall remain in this state.

This exceptional condition is indicated to the CC signalling service user by an ERROR indication primitive with an appropriate Diagnostics parameter.

9.8.5.3 Receipt of a reject APDU that is correlated to a callComplete invoke

On receipt of a reject APDU that is correlated to a callComplete invoke APDU while in state Call Active, the preceding CC entity shall clear the call by sending a callRelease invoke APDU, start timer T708, and enter the Call Release Request state. The callRelease invoke APDU sent to initiate call clearing shall contain causeValue 'temporary failure' and an appropriate location value in element releaseCause.

This exceptional condition is indicated to the CC signalling service user by an ERROR indication primitive with an appropriate Diagnostics parameter.

9.8.5.4 Receipt of a reject APDU that is correlated to a callStatus invoke

On receipt of a reject APDU that is correlated to a callStatus invoke APDU while in state Await Call Completion or Call Active, the CC entity shall remain in the same state.

This exceptional condition is indicated to the CC signalling service user by an ERROR indication primitive with an appropriate Diagnostics parameter.

9.8.5.5 Receipt of a reject APDU that is correlated to a callRelease invoke

On receipt of a reject APDU that is correlated to a callRelease invoke APDU while in state Call Release Request, the CC entity shall stop timer T708, release the call segment ID, and enter the Call Idle state.

A RELEASE-CALL confirm primitive is given to the CC signalling service user if call clearing has been initiated on user request.

9.8.5.6 Receipt of a reject APDU that is correlated to a callEstablish return result

On receipt of a reject APDU that is correlated to a callEstablish return result APDU while in state Call Active, or Await Call Completion, the succeeding CC entity shall clear the call internally, release the CallSegmentId, and enter the Call Idle state.

This exceptional condition is indicated to the CC signalling service user by an ERROR indication primitive with an appropriate Diagnostics parameter.

9.8.5.7 Receipt of a reject APDU that is correlated to a callEstablish return error

On receipt of a reject APDU that is correlated to a callEstablish return error APDU, no action shall be taken.

9.8.5.8 Receipt of a reject APDU that is correlated to a callRelease return result

On receipt of a reject APDU that is correlated to a callRelease return result APDU, no action shall be taken.

9.8.6 Handling of unrecognized parameters within CC-Operations

On receipt of an APDU containing a CC-Operation with one or more unrecognized parameters, the receiving CC entity shall examine the parameterActionIndicator contained in this operation and follow the procedures described below as appropriate.

- if the parameterActionIndicator is set to 'clearCallAndItsInformationModel', the receiving CC entity shall clear the call and its information model in accordance with the procedures specified in subclause 9.7;
- if the parameterActionIndicator is set to 'discardApduAndReject', the receiving CC entity shall discard the entire APDU and initiate sending of a reject APDU back to the peer CC entity;
- if the parameterActionIndicator is set to 'discardApduNoReject', the CC entity shall discard the entire APDU and not initiate sending of a reject APDU;
- if the parameterActionIndicator is set to 'discardParameterAndPassApduToApplication', the receiving CC entity shall discard the unrecognized parameter and pass the APDU without the unrecognized parameter on to the CC signalling service user;
- if the parameterActionIndicator is set to 'ignoreParameterAndPassApduToApplication', the receiving CC entity shall ignore the unrecognized parameter and pass the APDU including the unrecognized parameter on to the CC signalling service user.

10 Parameter values (Timers)

Timer number	Timer value	Call state	Cause for start	Normally terminated	Action to be taken when timer expires	Succeeding CC	Preceding CC
T703	3s-15s note	Call Initiated	On sending of a callEstablish invoke APDU	On receipt of a callEstablish return result/return error, or callProceeding invoke APDU	Clear call internally, release CallSegmentId, enter Call Idle state	-	M
T708	30 s	Call Release Request	On sending of a callRelease invoke APDU	On receiving a callRelease return result APDU	Release call segment ID and enter Call Idle state	M	M
T710	30 s	Outgoing Call Proceeding	callProceeding invoke APDU received	On receiving a callEstablish return result/return error APDU	Clear call	-	M
T701	180 s	Await Call Completion	On sending of a callEstablish return result APDU	On receipt of a callComplete invoke APDU	Clear call internally, release CallSegmentId, enter Call Idle state	M	-
NOTE: The value of T703 shall be chosen based on the underlying transport mechanism to be used.							

For timer value T703 the tolerance shall be -300ms/+3s.

For the other timer values specified in this clause the tolerance shall be +/- 10%.

11 Transport mechanism

The design of this protocol does not demand for a special transport mechanism. However, a reliable transport mechanism is required.

Transport mechanisms which can be used are listed in annex D. Subclauses in this annex D for the various transport mechanisms specify how a particular mechanism shall be used if it has been chosen.

NOTE

Annex D is normative but not exclusive, i.e. other reliable transport mechanisms which are not mentioned there can be used as well.

12 SDL Diagrams

The diagrams in this Standard use the Specification and Description Language defined in CCITT Rec. Z.100 (1993).

The diagrams represent the behaviour of the CC signalling service provider, which is split into outgoing CC-ASE and incoming CC-ASE (Application Service Element).

Input signals from the left and output signals to the left represent primitives to and from the CC signalling service user. Also protocol timer expiry is indicated by an input signal from the left.

Input signals from the right and output signals to the right represent APDUs sent to and received from the peer CC entity.

The following abbreviations are used:

inv.	invoke APDU	EST	ESTABLISH	req.	request primitive
res.	return result APDU	REL	RELEASE	ind.	indication primitive
err.	return error APDU	PROC	PROCEED	resp.	response primitive
rej.	reject APDU		conf.	confirm primitive	
	(+)	positive result			
	(-)	negative result			

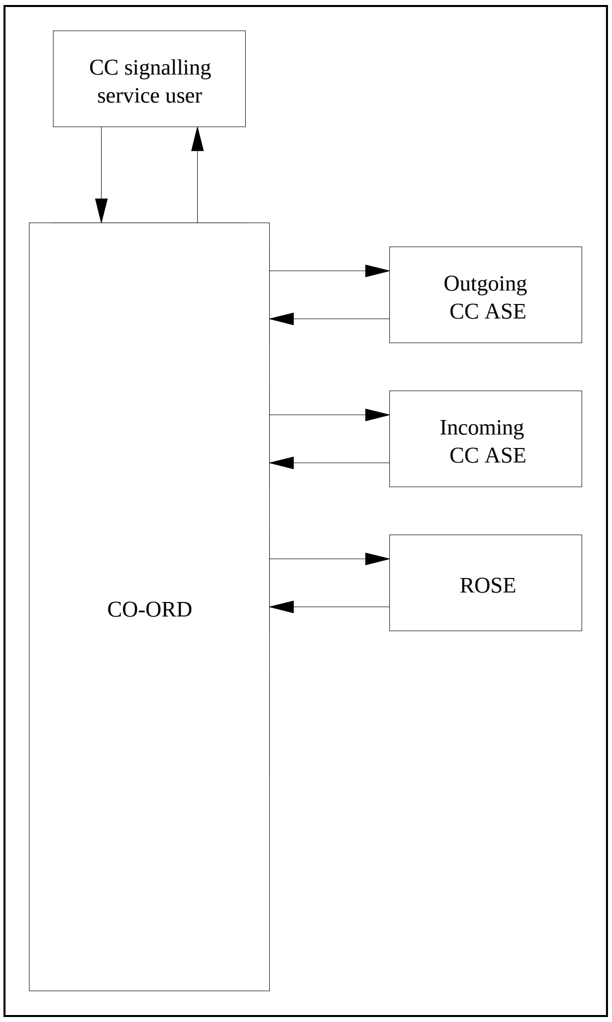

Figure 12 - Block diagram

Table 10 - Signal Routes

CO-ORD_to_Outgoing-CC-ASE	**Incoming-CC-ASE_to_CO-ORD**
Primitives: ESTABLISH_CALL_request COMPLETE_CALL_request STATUS_CALL_request RELEASE_CALL_request RELEASE_CALL_response	Primitives: ESTABLISH_CALL_indication COMPLETE_CALL_indication STATUS_CALL_indication RELEASE_CALL_indication RELEASE_CALL_confirm ERROR_indication
APDUs: callProceeding_invoke callEstablish_return_result callEstablish_return_error callEstablish_reject callStatus_invoke callRelease_invoke callRelease_return_result callRelease_reject	APDUs: callProceeding_invoke callEstablish_return_result callEstablish_return_error callStatus_invoke callRelease_invoke callRelease_return_result
Outgoing-CC-ASE_to_CO-ORD	**CO-ORD_to_Incoming-CC-ASE**
Primitives: ESTABLISH_CALL_confirm PROCEED_CALL_indication STATUS_CALL_indication RELEASE_CALL_indication RELEASE_CALL_confirm ERROR_indication	Primitives: ESTABLISH_CALL_response PROCEED_CALL_request STATUS_CALL_request RELEASE_CALL_request RELEASE_CALL_response
APDUs: callEstablish_invoke callComplete_invoke callStatus_invoke callRelease_invoke callRelease_result	APDUs: callEstablish_invoke callComplete_invoke callStatus_invoke callRelease_invoke callRelease_return_result callRelease_reject callEstablish_reject

12.1 Outgoing CC-ASE

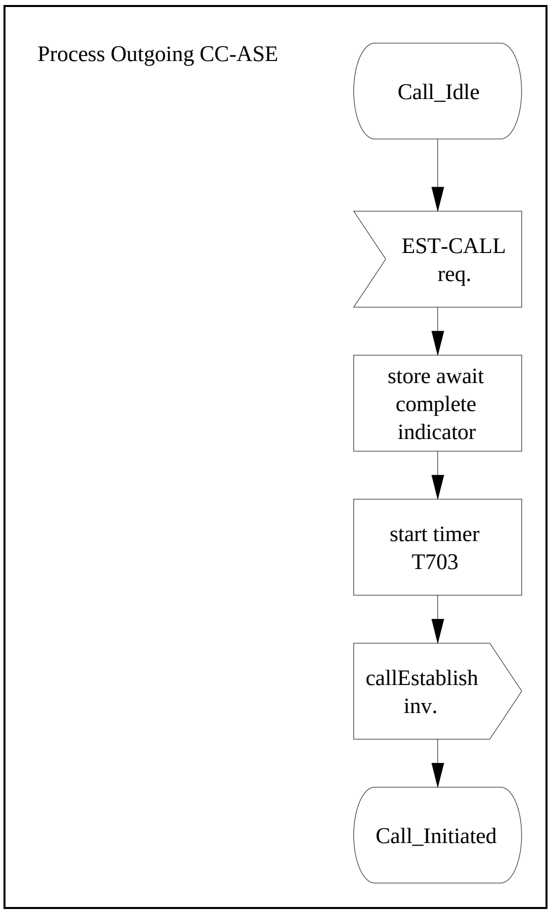

Figure 13 - SDL for CC, outgoing call

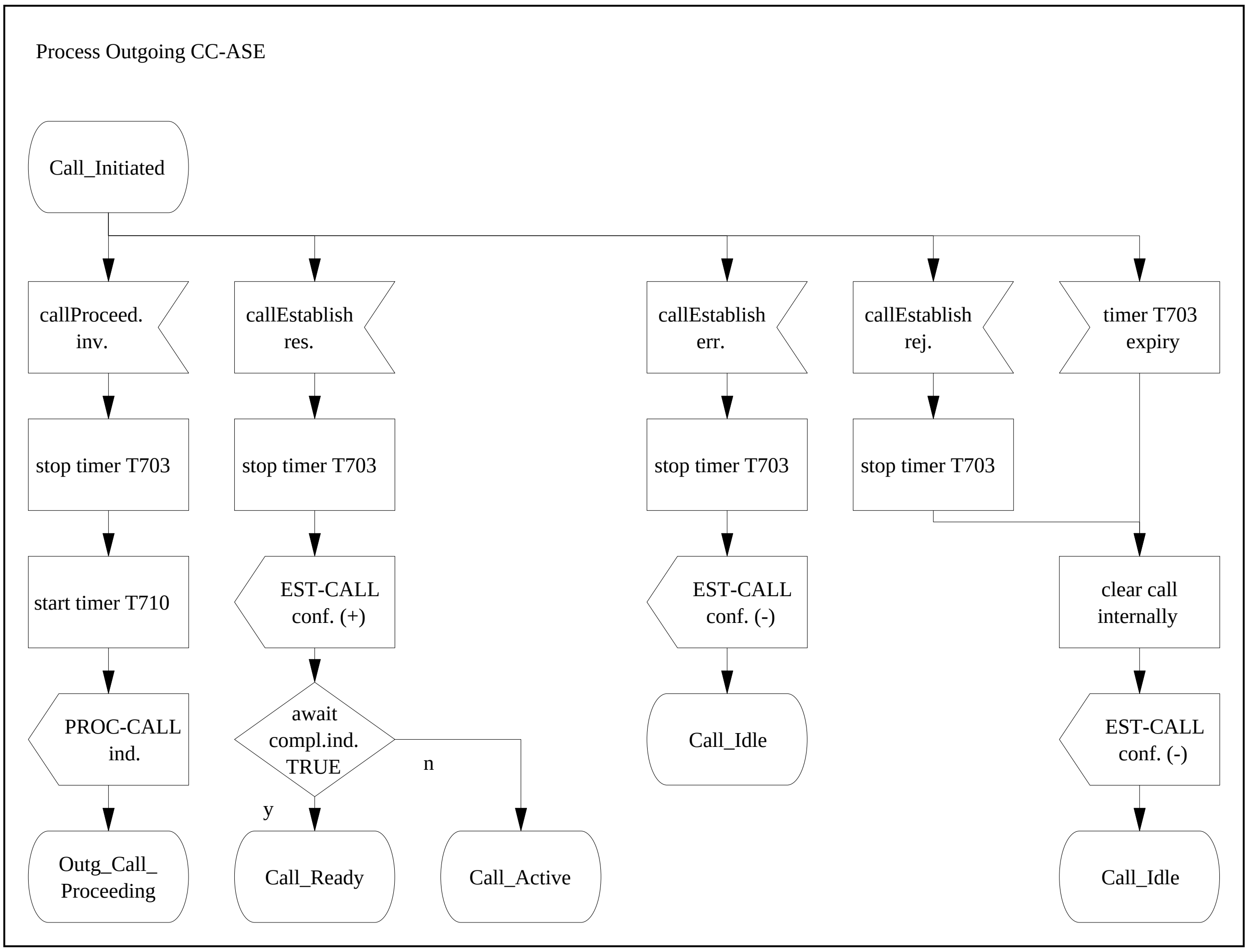

Figure 14 - SDL for CC, outgoing call

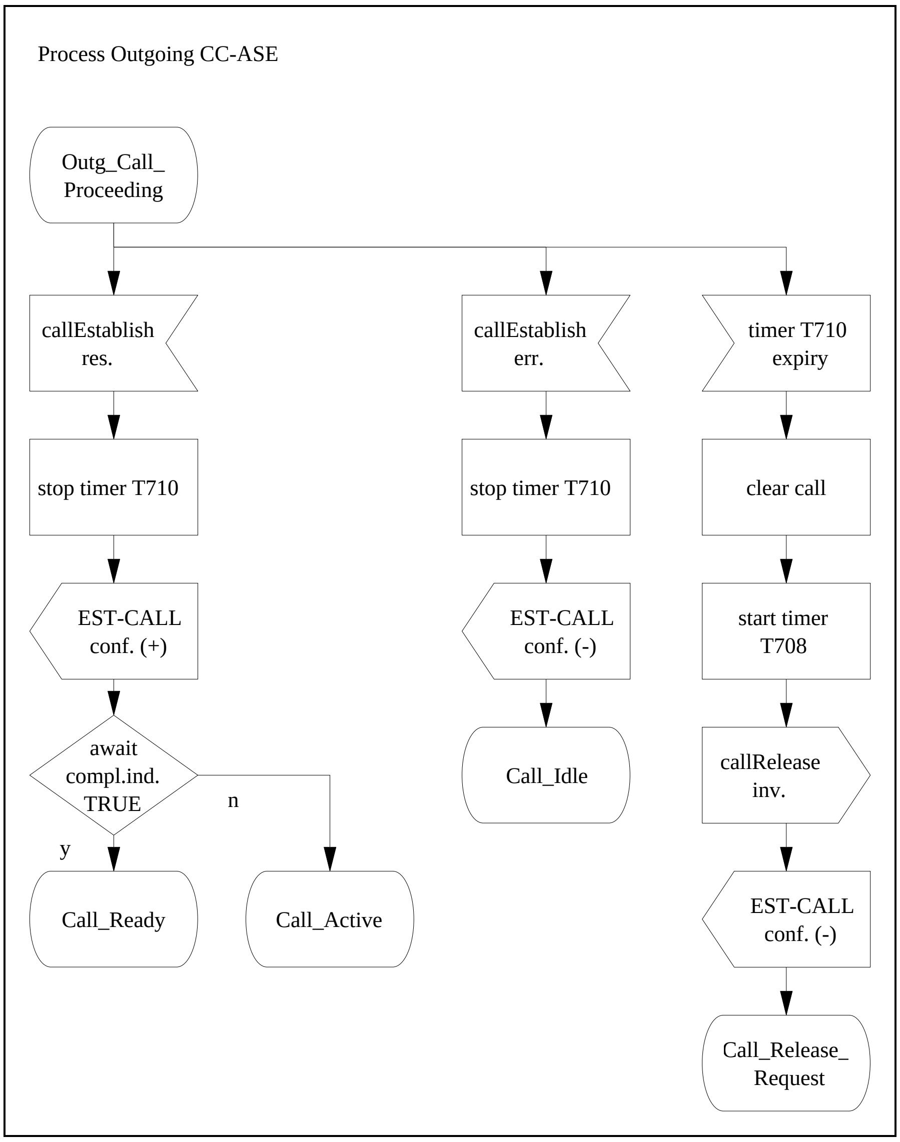

Figure 15 - SDL for CC, outgoing call

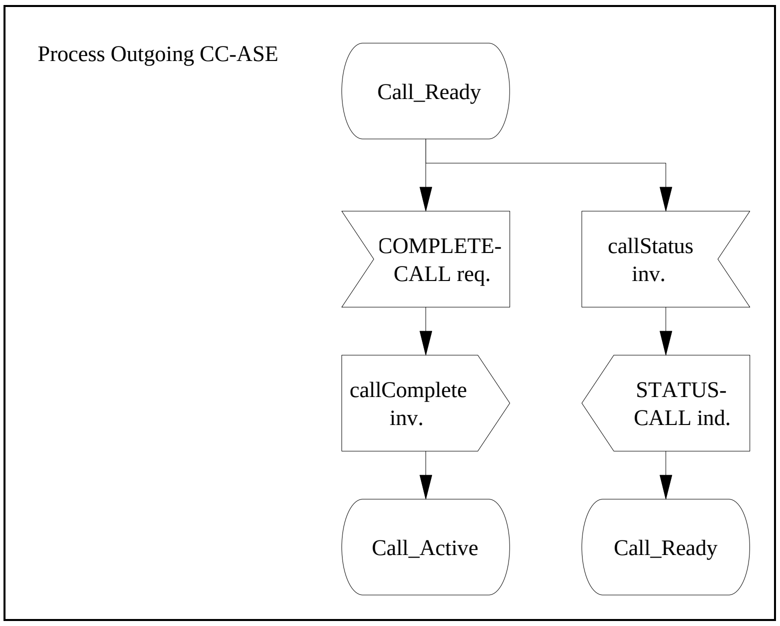

Figure 16 - SDL for CC, outgoing call

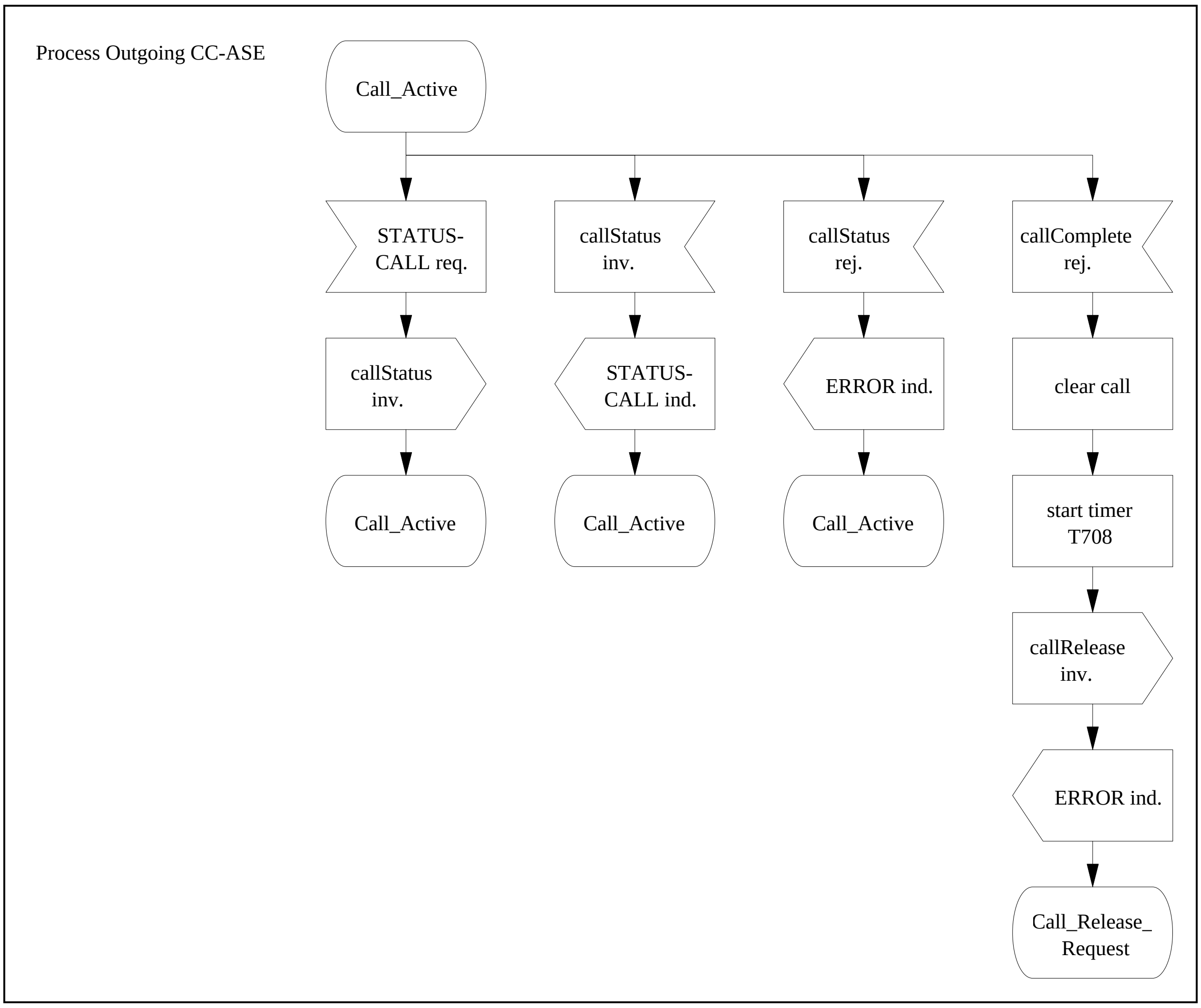

Figure 17 - SDL for CC, outgoing call

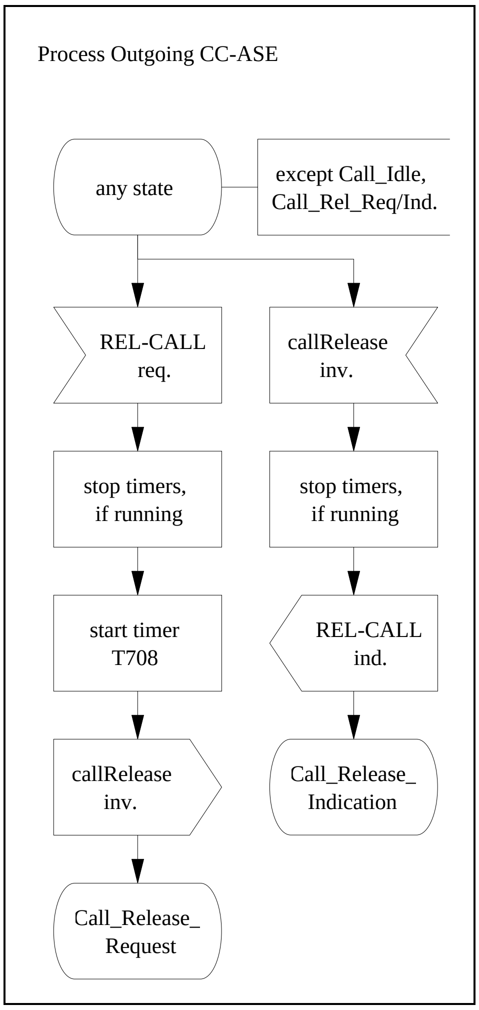

Figure 18 - SDL for CC, outgoing call, call release

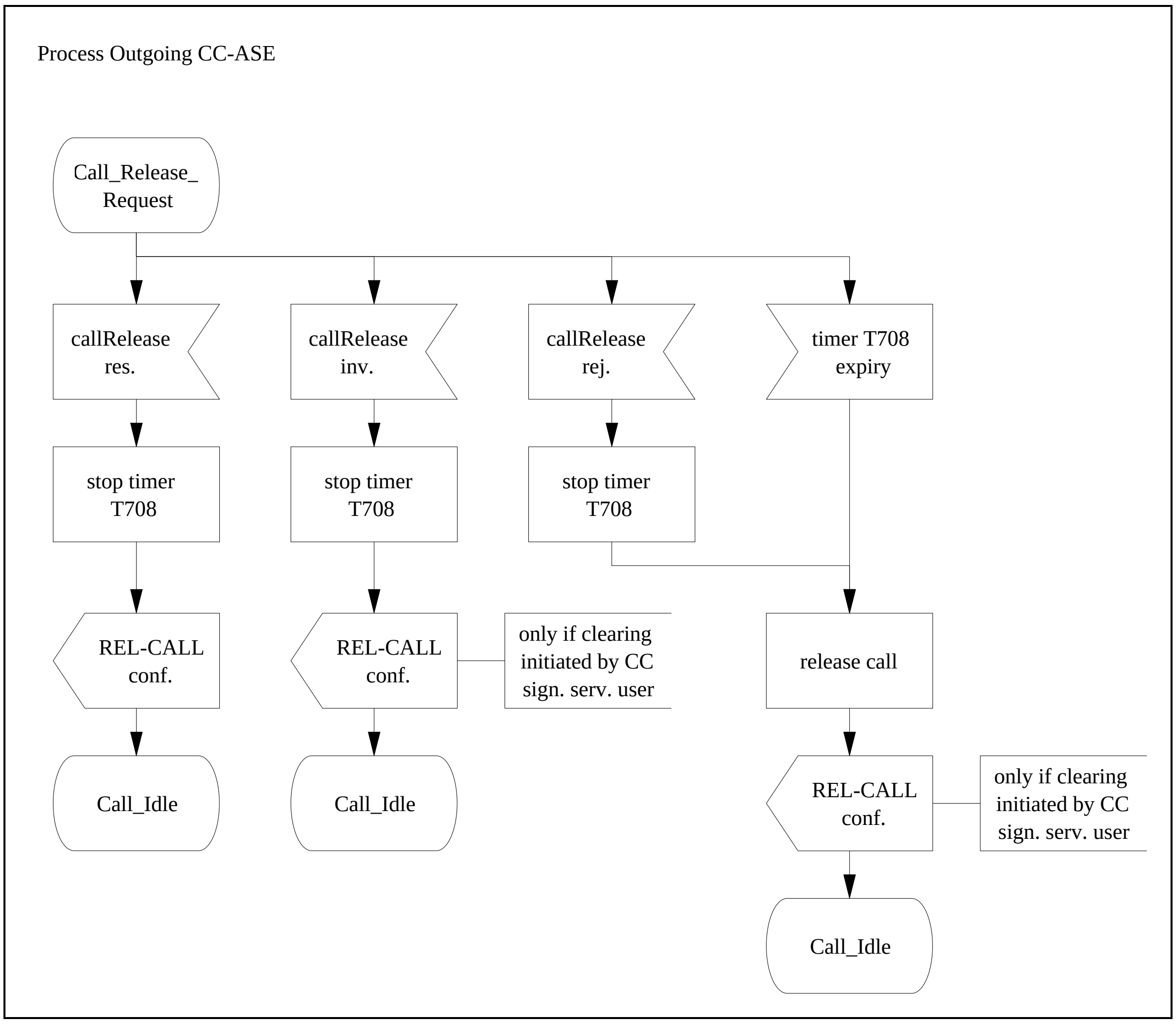

Figure 19 - SDL for CC, outgoing call, call release

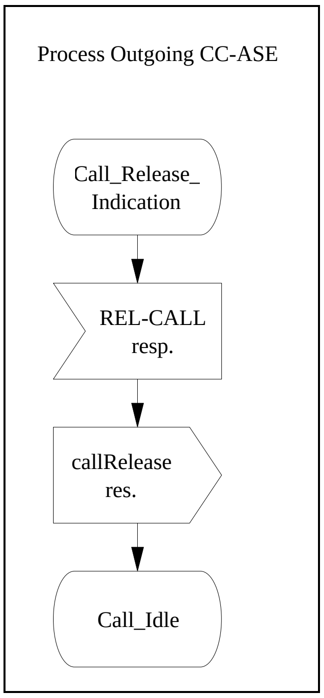

Figure 20 - SDL for CC, outgoing call, call release

12.2 Incoming CC-ASE

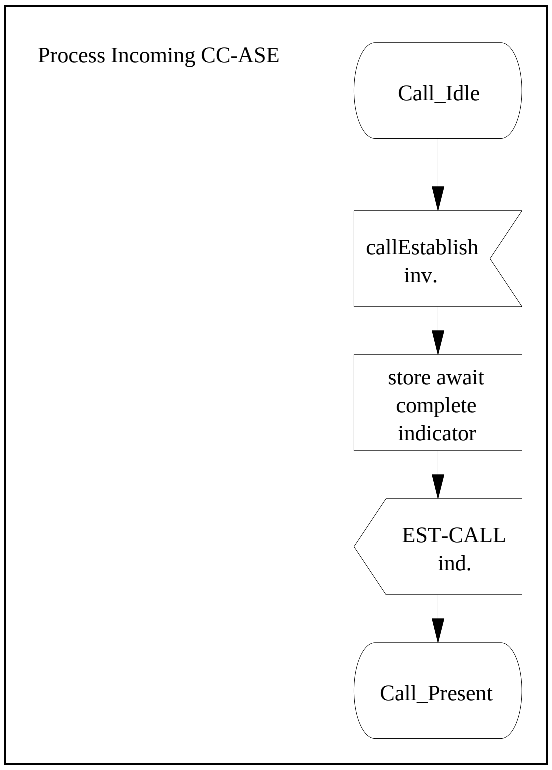

Figure 21 - SDL for CC, incoming call

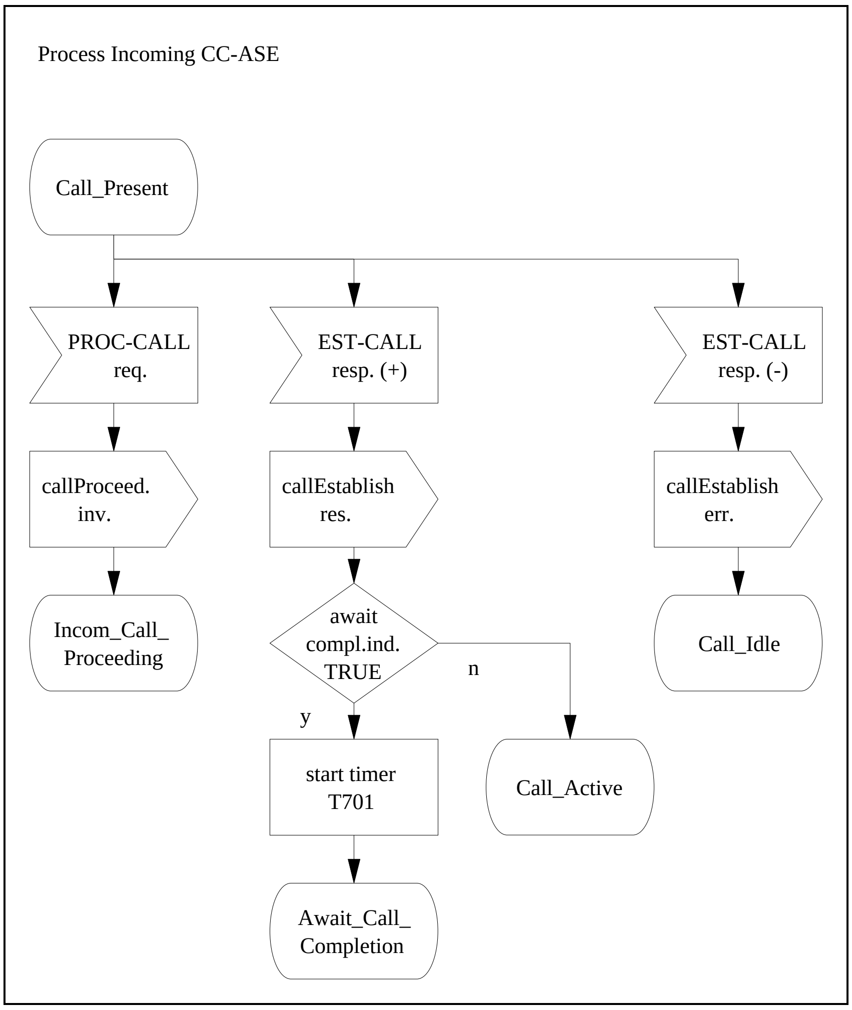

Figure 22 - SDL for CC, incoming call

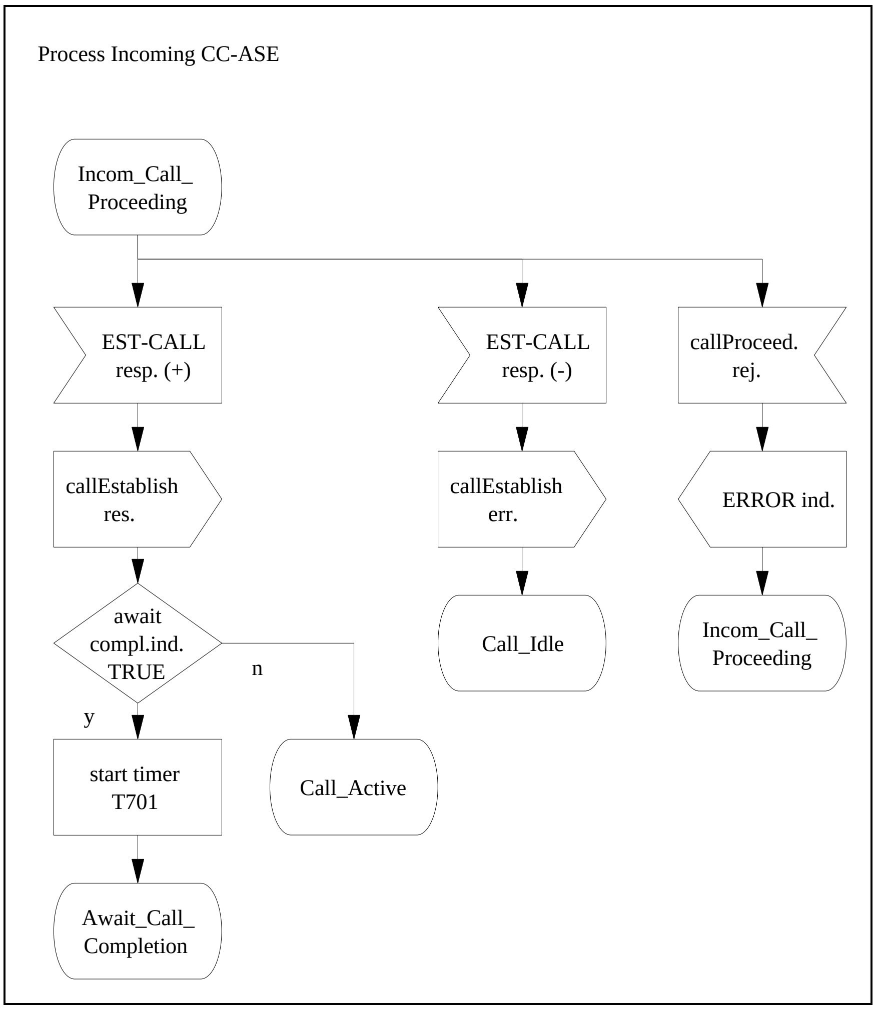

Figure 23 - SDL for CC, incoming call

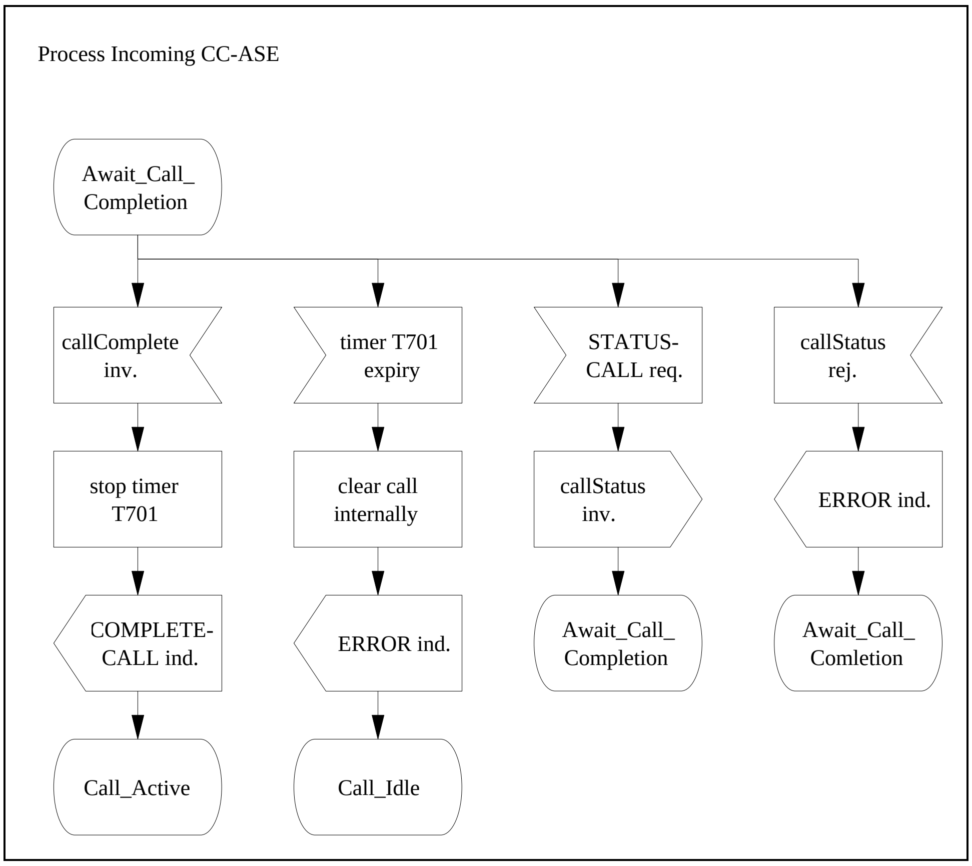

Figure 24 - SDL for CC, incoming call

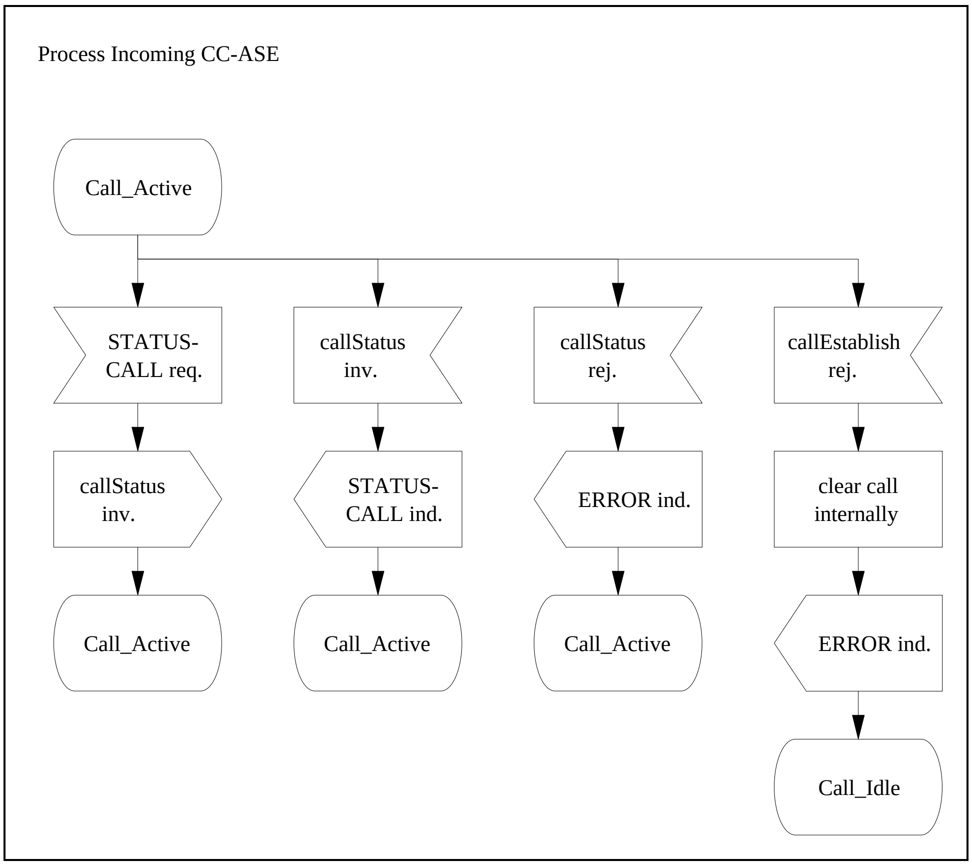

Figure 25 - SDL for CC, incoming call

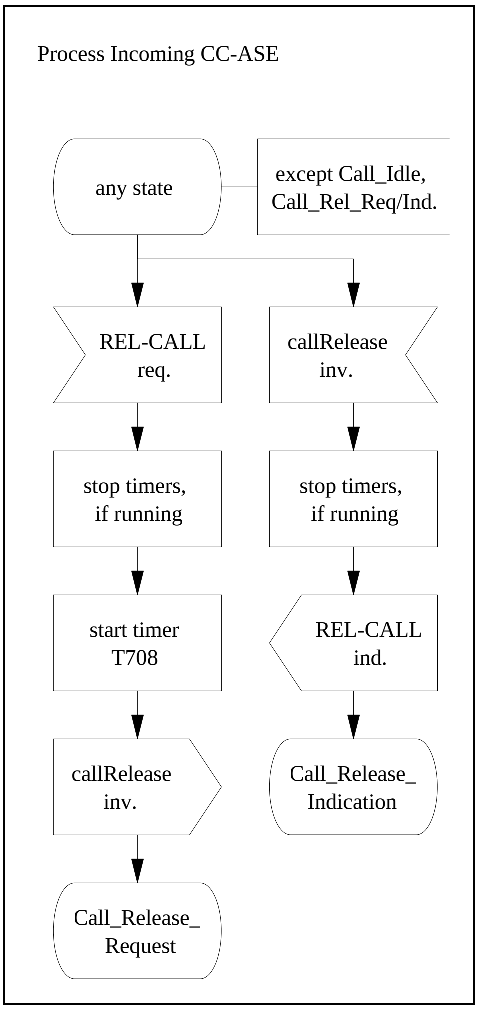

Figure 26 - SDL for CC, incoming call, call release

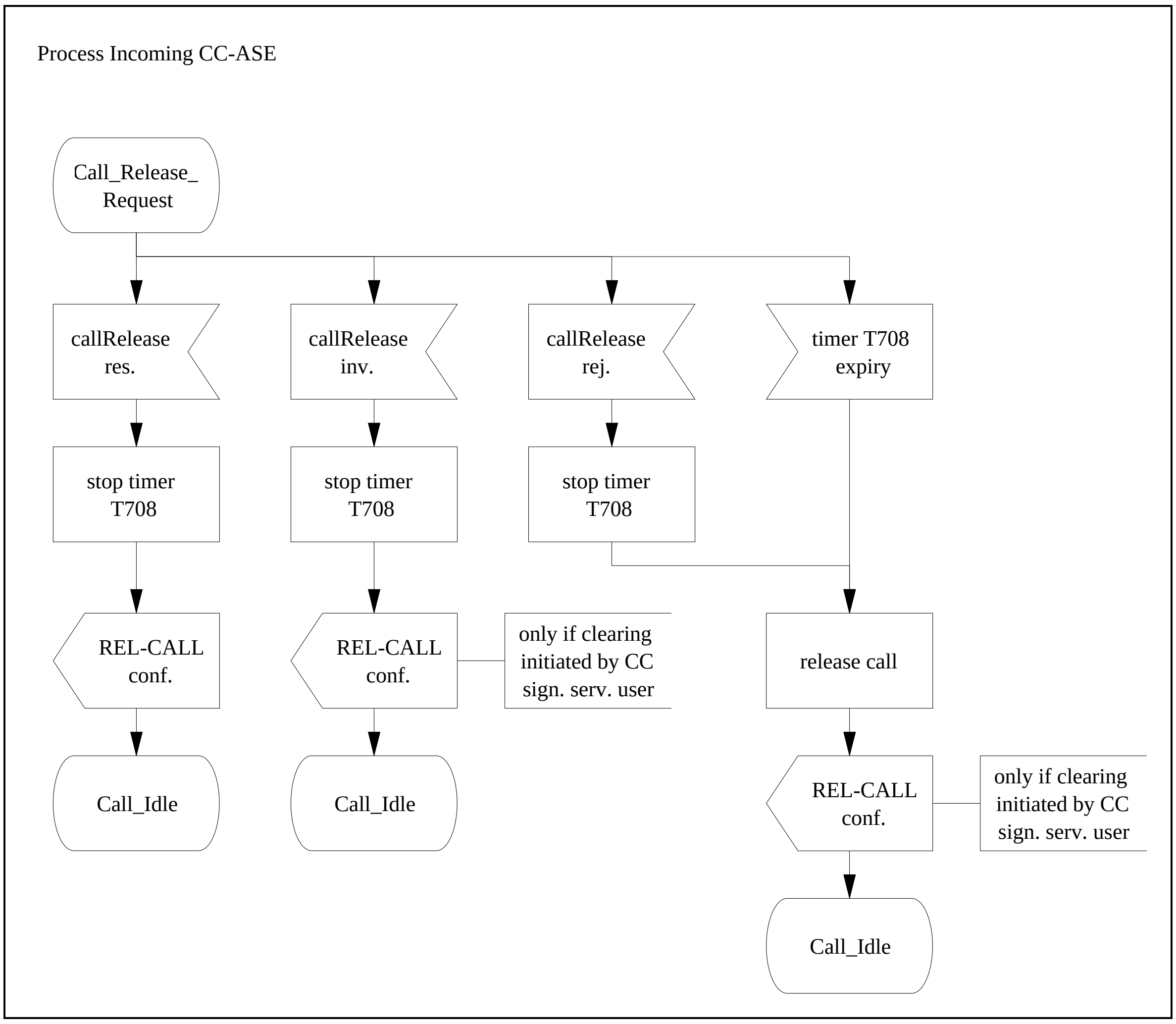

Figure 27 - SDL for CC, incoming call, call release

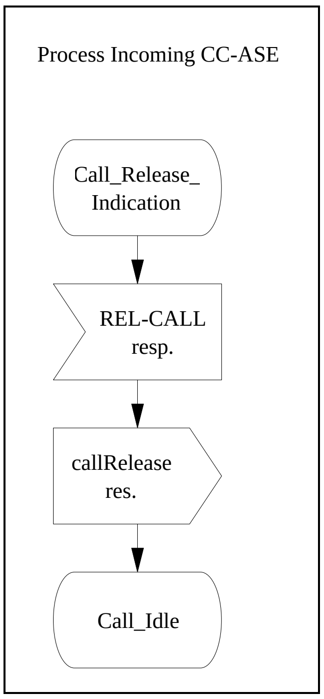

Figure 28 - SDL for CC, incoming call, call release

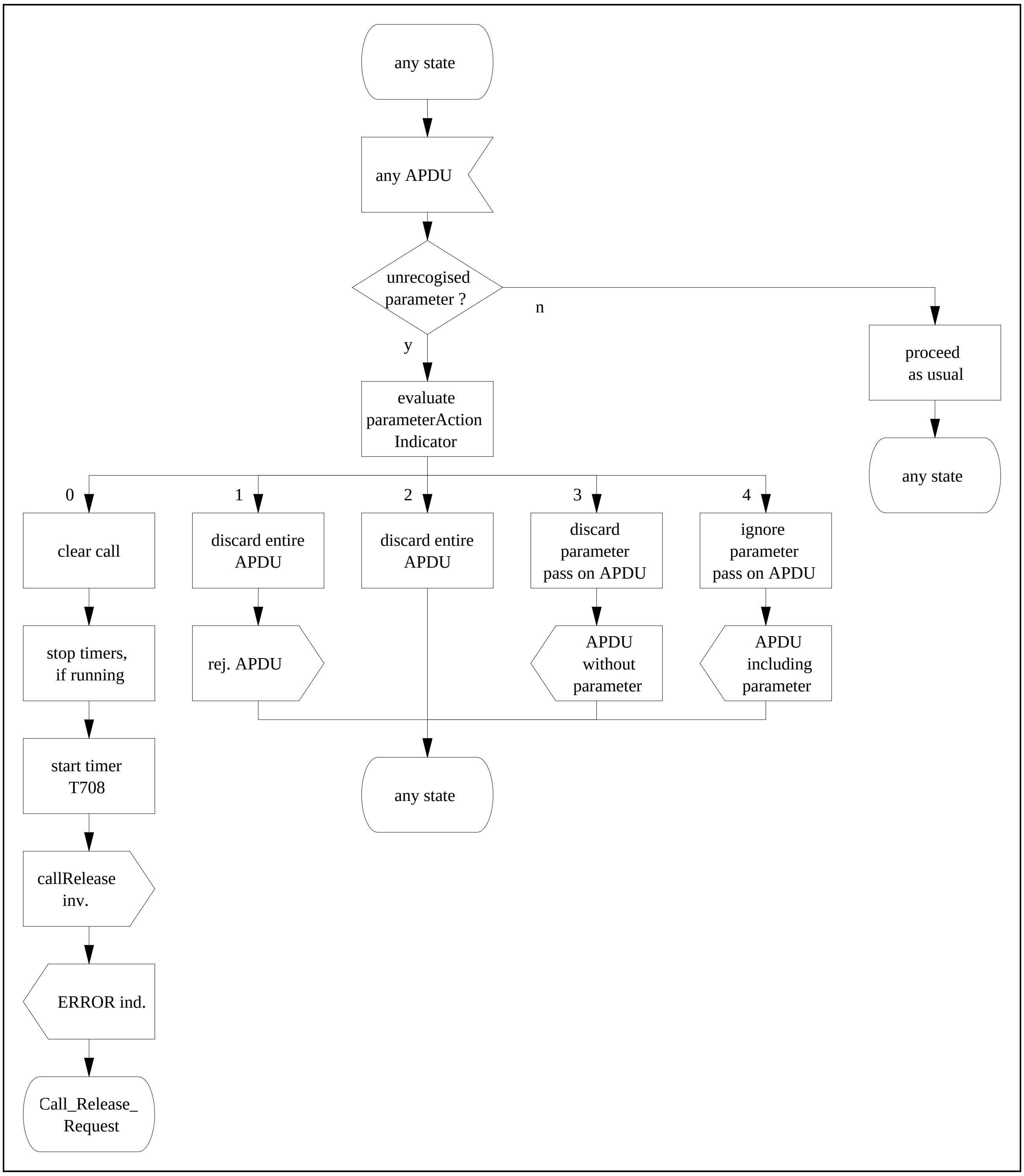

Figure 29 - SDL for the handling of unrecognised parameters within CC operations

Annex A

(normative)

Bearer co-ordination requirements for CC signalling service users

Signalling for BC is outside the scope of this Standard. The way in which a CC entity co-ordinates the call's bearers is also outside the scope of this Standard, except that the following requirements shall be met.

A.1 Requirements at a CC entity that establishes a bearer towards an adjacent CC entity

The requirements in this subclause apply to an end CC entity that initiates bearer establishment towards the adjacent CC entity on request of the application. An end CC entity can be an originating CC entity that initiates bearer establishment in the forward direction (with respect to the direction of call establishment) or a terminating CC entity that initiates bearer establishment in the backward direction (with respect to the direction of call establishment).

The requirements in this subclause also apply to a transit CC entity that continues bearer establishment towards the subsequent CC entity following receipt of an incoming bearer from the preceding CC entity (such a bearer being in the forward direction with respect to the direction of call establishment), or that continues bearer establishment towards the preceding CC entity following receipt of an incoming bearer from the subsequent CC entity (such a bearer being in the backward direction with respect to the direction of call establishment).

Bearer establishment in the forward direction shall not commence before one of the following APDUs has been received from the subsequent CC entity:

- callProceeding invoke APDU; or
- callEstablish return result APDU.

The contents of elements bearerEstablishmentAddress and callSegmentId in the first of the above APDUs to be received shall be used as the destination address and the call segment identifier respectively for bearer establishment.

NOTE

Omission of either of these elements from the first of these APDUs constitutes a protocol error.

Bearer establishment in the backward direction shall not commence before one of the following APDUs has been sent to the preceding CC entity:

- callProceeding invoke APDU; or
- callEstablish return result APDU.

The contents of element callSegmentId in the first of the above APDUs to be transmitted and element bearerEstablishmentAddress in the received callEstablish invoke APDU shall be used as the call segment identifier and the destination address respectively for bearer establishment.

NOTE 1

For bearer establishment in either direction, the value used for the bearer destination address will cause the bearer to be routed to the terminal or network node where the CC entity at the opposite end of the call segment is located.

NOTE 2

For bearer establishment in either direction, the call segment identifier is conveyed transparently by BC signalling to the terminal or network node where the CC entity at the opposite end of the call segment is located, thereby allowing that terminal or network node to link the bearer to the call.

NOTE 3

CC signalling service users should ensure that bearer connection elements established by associated BC prior to call acceptance are consistent with all variants of selectable options within the call description included in the call *establishment request. After call establishment CC entities should ensure that bearer connection elements are consistent with the final call description. Bearer connection elements and bearer connection establishment requests failing these criteria should be rejected or released as appropriate.*

For bearer establishment in either direction, an address representing the local CC entity (i.e., the CC entity at the start of the segment) shall be used as the calling address of the bearer.

For bearer establishment in either direction, a bearer identifier value by which the bearer is to be known by the two CC entities at either end of the call segment may be provided as part of the bearer establishment request. The bearer identifier value shall have significance across the call segment concerned.

NOTE 4

For bearer establishment in either direction, the bearer identifier value, if provided in the bearer establishment request, is conveyed transparently by BC signalling to the terminal or network node where the CC entity at the opposite end of the call segment is located. Its purpose is to allow that CC entity to associate the bearer establishment with a bearer reference conveyed within CC signalling (e.g., as an attribute of an Attachment object). If CC signalling has already made reference to this bearer, prior to its establishment, the bearer identifier value used in the bearer establishment request should be the same as the bearer reference already used by CC signalling. Any further CC signalling that needs to refer to this bearer should use the bearer identifier value used in the bearer establishment request.

Bearer establishment shall not commence after a callRelease invoke APDU has been sent or received across the segment concerned.

A.2 Requirements at a CC entity that receives a bearer establishment from an adjacent CC entity

The requirements in this subclause apply to a CC entity that receives an incoming bearer establishment from an adjacent CC entity. This can be from the preceding CC entity (in the case of a bearer established in the forward direction with respect to the direction of call establishment) or from the subsequent CC entity (in the case of a bearer established in the backward direction with respect to the direction of call establishment).

A terminal or network node that receives an incoming bearer establishment signal containing a destination address that indicates that that terminal or network node is the destination of the bearer shall attempt to match the received call segment identifier with a call segment identifier assigned to a call segment that is associated with a CC entity on that terminal or network node. If a match is found, the CC entity concerned shall proceed with establishment of the bearer.

NOTE 1

In the case of a transit CC, the time at which bearer establishment is continued across the next call segment is outside the scope of this Standard, apart from being subject to the restrictions in A.1.

NOTE 2

In the case of an end CC, the bearer should be presented to the application.

NOTE 3

CC signalling service users should ensure that bearer connection elements established by associated BC prior to call acceptance are consistent with all variants of selectable options within the call description included in the call establishment request. After call establishment CC entities should ensure that bearer connection elements are consistent with the final call description. Bearer connection elements and bearer connection establishment requests failing these criteria should be rejected or released as appropriate.

If the terminal or network node is the destination of the bearer but is unable to match the received call segment identifier with a call segment identifier assigned to a call segment that is associated with a CC entity on that terminal or network node, the terminal or network node shall reject the bearer establishment request.

Even though a succeeding CC entity is required to send a callProceeding invoke APDU or callEstablish return result APDU before commencing bearer establishment in the backward direction, it is possible for a bearer establishment request to arrive at a preceding CC entity before any of the above APDUs arrives. In this case, the preceding CC shall await the arrival of one of the above APDUs before continuing to process the bearer establishment request. If timer T703 expires, the bearer establishment request shall be released.

NOTE 4

If the received bearer establishment request contains a bearer identifier value, this value should be used by the CC entity in any future CC signalling relating to this bearer. If CC signalling has already made reference to this bearer, prior to its establishment, using this value, the incoming bearer establishment request should be associated with that bearer reference.

A.3 Additional requirements at a transit CC

A transit CC shall relay an alerting or answer indication from the outgoing bearer on to the incoming bearer.

A transit CC shall relay a release indication from either bearer on to the other bearer, except that a release indication from an outgoing bearer that is in the bearer establishment phase may instead, depending on the cause of release, result in re-routeing of the outgoing bearer.

A transit CC shall pass on any subaddress information from one bearer unchanged to the other bearer.

A transit CC shall pass on any low layer or high layer compatibility information (e.g., Q.2931 B-HLI, B-LLI, N-HLC, N-LLC or N-BC information elements) from one bearer unchanged to the other bearer.

A transit CC shall pass on any bearer-related supplementary service information from one bearer, unless acted upon at the network node concerned, unchanged to the other bearer.

A.4 Requirement on call clearing

On sending or receiving a callRelease invoke APDU across a call segment, the CC entity shall immediately initiate the release of any bearers across that call segment that are not already in the process of being released.

Annex B

(normative)

Call Description handling requirements for CC signalling service users

B.1 Call description handling at a CC signalling service user within an originating CC entity

When requesting the CC signalling service provider to send a callEstablish invoke APDU, a CC signalling service user within an originating CC shall include a call description containing the following objects:

Network relevant objects:

- one Call object, with status M;
- two Call Party End Point objects, one indicated in the Call object as local and identifying the local user, the other indicated in the Call object as remote, and both with status M;
- one Call Party End Point Association object referencing the two Call Party End Point objects, with status conditional;
- optionally one or more Service Component objects, each with status either M or O.

End-to-end relevant objects:

- for each Service Component object, two Participation objects (one per Call Party End Point object), each with status conditional.

B.2 Call description handling at a CC signalling service user within a transit CC entity

A CC signalling service user within a transit CC may modify the network relevant objects contained in a call description received in a callEstablish invoke APDU prior to requesting the CC signalling service provider to pass it on in a callEstablish invoke APDU towards the next CC. Modification shall be limited to the following:

- the removal of one or more objects with status O; and
- the removal of any objects with status conditional that depend on other objects removed.

NOTE 1

This means, for example, that objects cannot be added, objects with status M cannot be deleted, and object attributes cannot be modified.

If a CC signalling service user within a transit CC is unable to accept the network relevant objects contained in the call description as received and is unable to achieve an acceptable call description by means of modification in accordance with the rules above, it shall reject the call with an appropriate error value, e.g., callDescriptionNotAccepted.

NOTE 2

A return error APDU with error value callDescriptionNotAccepted can contain an alternative call description that would be acceptable to the CC signalling service user.

A CC signalling service user within a transit CC shall not modify a call description passed on in any other APDU (i.e., callEstablish return result or callEstablish return error), nor the callChangedParameter in a callStatus invoke APDU.

A CC signalling service user within a transit CC shall pass on transparently the end-to-end relevant objects contained in a call description received in a callEstablish invoke APDU.

B.3 Call description handling at a CC signalling service user within a terminating CC entity

A CC signalling service user within a terminating CC may modify a the end-to-end and network relevant objects contained in call description received in a callEstablish invoke APDU prior to requesting the CC signalling service provider to send back a callEstablish return result APDU. Modification shall be limited to the following:

- the removal of one or more objects with status O; and
- the removal of any objects with status conditional that depend on other objects removed.

NOTE 1

This means, for example, that objects cannot be added, objects with status M cannot be deleted, and object attributes cannot be modified.

If a CC signalling service user within a terminating CC is unable to accept the call description as received and is unable to achieve an acceptable call description by means of modification in accordance with the rules above, it shall reject the call with an appropriate error value, e.g., callDescriptionNotAccepted, userBusy, callRejected.

NOTE 2

A return error APDU with error value callDescriptionNotAccepted or userBusy can contain an alternative call description that would be acceptable to the CC signalling service user.

NOTE 3

One reason for a received call description being unacceptable is that it has been sent by equipment that supports enhanced capabilities beyond the scope of this Standard and therefore contains additional objects (e.g., more than two Party End Point objects).

B.4 Call description errors

When a call description parameter is received which has one or more unexpected object identifiers or object identifiers with unrecogn*ize*d attributes (arguments), the receiving entity shall examine the object action indicator, and follow the procedures described in a), b), c), d) or e) below as appropriate.

If more than one object identifiers and/or object attributes are received in error, only one response shall be given. The response shall be according to the handling of the object action indicator attribute according to the following order of priority: 'clearCall' (highest priority), 'discardNotify', 'discardUnknown', 'progressTransit'.

a) Object action indicator attribute = clearCall.
If the object action indicator attribute is equal to "clear call object model", the call shall be cleared according to the procedures defined in 9.7 except that the Cause information element shall contain the cause "call DescriptionNotAccepted".

b) If the object action indicator attribute = discardNotify.
The call description shall be ignored and a returnError APDU with error value "callDescriptionNotAccepted" shall be returned.

c) Object action indicator attribute = discardUnknown.
If the object action indicator attribute is equal to "discard unknown item and proceed", the unknown item (either the entire object or only the unknown attribute) shall be ignored and the call description shall be processed as if the unknown information was not received. No returnError APDU shall be sent.

d) Object action indicator attribute = progressTransit.
The unknown object or unknown attribute shall be progressed as an octet string parameter to succeeding CC (if the operation requires so), but shall not be retained by the CC service user once the operation is complete.

e) Object action indicator attribute = unknown value.
If an unknown object or unknown attribute has and associated object action indicator that contains an unknown value, then the receiver shall handle the call description as if the object action indicator attribute had been set to "progressTransit".

B.5 End-to-end relevant object handling at a CC signalling service user within a transit CC entity

End-to-end relevant objects (service components) shall always be handled by a CC signalling service user within a transit CC entity as unknown objects with an object action indicator attribute set to "progressTransit", regardless of the value that the object action indicator attribute is actually set to. The operation shall always be progressed when this type of object is the only unknown error.

B.6 Changes to the Information Model

The CC protocol provides a single mechanism to indicate changes to the information model, the Status Call procedure. This procedure provides for a single unconfirmed flow and therefore represents only non-negotiable changes that reflect events that have already happened. The changes to the information model that the CC signalling service user shall indicate using this procedure are:

- addition and deletion of a service component object;
- changes to the attributes of a party object (i.e. modification of the status of a party);
- changes to the OpenCall attribute of the call object (i.e. modification of the call permissions).

B.6.1 Deletion of a Service Component Object

The deletion of a service component reflects the ability of any user application to discontinue using some portion of an implementation's functionality. Any CC signalling service user at an originating or terminating CC entity can delete a service component in the information model. An attempt to delete a service component that does not exist shall not cause an error, but shall be ignored by the CC signalling service user at the peer originating or terminating CC entity.

B.6.2 Addition of a Service Component Object

The addition of a service component using the Status Call procedure reflects the ability of an originating or terminating CC entity to begin using some portion of an implementation's functionality. The Status Call procedure does not support either confirmation or negotiation of this functionality and so it is assumed that this does not place any additional requirements on the bearer. The CC signalling service user at the peer originating or terminating CC entity, receiving an indication of an added service component may choose to ignore the indication.

B.6.3 Changes to the attributes of the Party Object

The changes to the attributes of a party object that the CC signalling service user shall indicate are:

- changes to the status of a party (e.g. alerting);
- changes to the type of a party.

The procedures associated with the status call require that the complete party object shall be provided.

B.6.4 Changes to the Open Call attribute of the Call Object

The OpenCall attribute of the call object defines the rights of parties to modify the call by the addition of parties, connections or service components. The permissions provide the requirements for notification and requesting call owner permission.

The OpenCall parameter contains the following Boolean indications which shall take the values shown:

- externalPartyAddAllowed

  Value: False.

  Usage: indicates that another party, not currently part of the call, shall not join the call configuration.

- existingPartyAddAllowed

  Value: False

  Usage: indicates that a non-call owner party shall not introduce another party, not currently part of the configuration, into the call.

- notifyAllPartiesFlag

  Value: True

  Usage: indicates whether a party successfully joining the call is required to be notified to all the existing members of the call. This indication is provided for future capability and is not currently used. It shall be set to true as this is more restrictive

- notifyOwnerFlag

  Value: True.

  Usage: indicates whether a party successfully joining the call is required to be notified to the call owner. This indication is provided for future capability and is not currently used. It shall be set to true as this is more restrictive

- permissionRequiredFlag

  Value: True

  Usage: indicates whether a party may join the call only after the call owners permission has been sought. This indication is provided for future capability and is not currently used. It is shall be set to true as this is more restrictive

- addConnectionAllowed

  Value: True or False

  Usage: indicates whether the non-call owner is permitted to add a connection to an existing call. If the value is set to true then any party may add a connection and also the CC signalling service provider shall refuse any request to change this value from true to false. A change from false to true is permitted.

- addServiceComponentAllowed

  Value: True or False

  Usage: indicates whether the non-call owner is permitted to add a service component to an existing call. If the value is set to true then any party may add a service component to a connection. The CC signalling service provider shall refuse any request to change this value from true to false. A change from false to true is permitted.

The call permissions are indicated to the non-call owner as part of the call description during call establishment. The call permissions are not a subject for negotiation and shall not be changed by any CC signalling service user except the call owner CC signalling service user.

Changes to the call permissions that reduce or restrict the modification rights of non call owners can result in conflicts. To prevent protocol clashes, any CC protocol service provider shall refuse to accept a request by a CC signalling service user to alter the OpenCall parameter if such an alteration removes a capability already granted. A refusal shall result in the changes to the OpenCall attribute of the call object being ignored and not being passed on to any succeeding CC signalling service user.

A CC signalling service user that receives a STATUS-CALL indication primitive that alters the call modification permission shall store the new permissions and pass on unchanged the OpenCall parameter towards the next CC using the STATUS-CALL request.

The CC signalling service user that initiated the call may allow another party to add either connections or service components at call establishment by initiating a call with either the addConnectionAllowed flag set to TRUE or the addServiceComponentAllowed flag to TRUE or both. The CC signalling service user that initiated the call may change the permissions to allow another party to add either connections or service components after call establishment using the call status procedure to change either of the above flags from false to true. The call status procedure shall not be used to change either of the flags from TRUE to FALSE. The call status procedure shall not be used to change any of the other flags in the OpenCall parameter.

Annex C

(normative)

Interworking

Interworking occurs with other networks, which do not support the separation of CC and BC or which send simultaneous call and bearer establishment requests.

Interworking with such networks requires both CC and BC functions in the gateway node at the boundary to these networks.

The interworking is performed by the CC signalling service user of the CC entity in the gateway node.

C.1 Interworking with networks not supporting separation of CC and BC

C.1.1 Outgoing call establishment

In case of an outgoing call establishment to a network which does not support the separation of CC and BC, the CC entity in the gateway node shall act as the terminating CC entity for that call. When the call establishment has been accepted by the terminating CC entity in the gateway node, subsequent bearer establishment requests which match with that call, shall be forwarded to the other network.

C.1.2 Incoming bearer establishment

In case of an incoming bearer establishment from a network which does not support the separation of CC and BC, the establishment can optionally be progressed in accordance with this Standard. In this case the CC entity in the gateway node shall act as the originating CC entity and initiate establishment of a call, that matches with the required bearer. When the call establishment has been accepted, i.e. after receipt of a callProceeding invoke APDU or callEstablish return result APDU, the bearer establishment shall be progressed.

C.1.3 Addition of bearers to an existing call

When a call exists between an originating CC entity and the CC entity in a gateway node to a network which does not support the separation of CC and BC, and a bearer has to be added to that call, the decision as to whether or not the additional bearer is supported is not taken in the gateway, but another bearer establishment shall be made across the other network to the same addressed terminal.

C.2 Interworking with simultaneous call and bearer establishment

In this case of interworking the other network supports the separation of CC and BC but only the capability of simultaneous establishment of a call with one bearer is supported.

C.2.1 Outgoing call establishment

In case of an outgoing call establishment to a network which only supports simultaneous call and bearer establishment, the CC entity in the gateway node shall act as the terminating CC entity for that call. When the call establishment has been accepted by the terminating CC entity in the gateway node and when a subsequent bearer establishment request belonging to that call is received, a simultaneous call and bearer establishment shall be initiated towards the other network.

C.2.2 Incoming call and bearer establishment

In case of an incoming call and bearer establishment, the CC entity in the gateway node shall act as the originating CC entity and initiate separate establishment of a corresponding call. When the call establishment has been accepted, i.e. after receipt of a callProceeding invoke APDU or callEstablish return result APDU, the bearer establishment shall be progressed.

C.2.3 Addition of bearers to an existing call

When a call exists between a CC entity and the CC entity in a gateway node to a network which only supports simultaneous call and bearer establishment, and a bearer in outgoing direction has to be added to that call, the decision as to whether or not the additional bearer is supported is not taken in the gateway, but another bearer establishment shall be made across the other network to the same addressed terminal. If a bearer in incoming direction has to be added to that call, i.e. if an incoming bearer establishment is received, the gateway shall forward it to the corresponding destination.

Annex D

(normative)

Transport mechanisms

This annex lists transport mechanisms which can be used with the CC protocol and specifies how to use them.

NOTE

This annex is normative but not exclusive, i.e. other reliable transport mechanisms which are not mentioned here can be used as well.

D.1 Connection oriented - Bearer independent

The connection oriented - bearer independent (CO-BI) transport mechanism is specified in ETS 300 796-1 for the S_B and coincident S_B/T_B reference point and in ECMA-254 for the Q_B reference point. If the CO-BI transport mechanism is chosen, the following shall apply:

- the operations defined in 8.1 shall be coded in the Facility information element in accordance with ETS 300 796-1 or ECMA-254 respectively;
- the instruction indicator in the Facility information element shall be coded in accordance with ETS 300 796-1 or ECMA-254 respectively;
- the Facility information element shall be conveyed in the messages for the CO-BI transport mechanism as specified in ETS 300 796-1 or ECMA-254 respectively;
- the instruction indicator in the messages for the CO-BI transport mechanism shall be coded in accordance with ETS 300 796-1 or ECMA-254 respectively.

Additionally for the Q_B reference point:

- when conveying the invoke APDU of operations defined in 8.1, the NFE (Network Facility Extension) shall either be omitted or be included in accordance with ECMA-254;
- when conveying the invoke APDU of operations defined in 8.1, the Interpretation APDU shall either be omitted or be included in accordance with ECMA-254.

D.2 Connectionless - Bearer independent

The connectionless - bearer independent (CL-BI) transport mechanism is specified in ETS 300 796-1 for the S_B and coincident S_B/T_B reference point and in ECMA-254 for the Q_B reference point. If the CL-BI transport mechanism is chosen, the following shall apply:

- the operations defined in 8.1 shall be coded in the Facility information element in accordance with ETS 300 796-1 or ECMA-254 respectively;
- the instruction indicator in the Facility information element shall be coded in accordance with ETS 300 796-1 or ECMA-254 respectively;
- the Facility information element shall be conveyed in the FACILITY message for the CL-BI transport mechanism as specified in ETS 300 796-1 or ECMA-254 respectively;
- the instruction indicator in the FACILITY message for the CL-BI transport mechanism shall be coded in accordance with ETS 300 796-1 or ECMA-254 respectively.

Additionally for the Q_B reference point:

- when conveying the invoke APDU of operations defined in 8.1, the NFE (Network Facility Extension) shall either be omitted or be included in accordance with ECMA-254;

- when conveying the invoke APDU of operations defined in 8.1, the Interpretation APDU shall either be omitted or be included in accordance with ECMA-254.

Annex E

(informative)

Information flow diagrams

This annex describes some typical information flows for CC. The following conventions are used in the figures of this annex:

- the figures show APDUs exchanged between CC entities involved in CC. Only APDUs relevant to CC are shown;
- the figures show protocol states related to the incoming and outgoing side of the CC signalling service provider within a CC entity;
- the figures show the primitives to and from the CC signalling service user within the user CC which correspond to the exchanged APDUs.

E.1 Call Establishment using a two message sequence

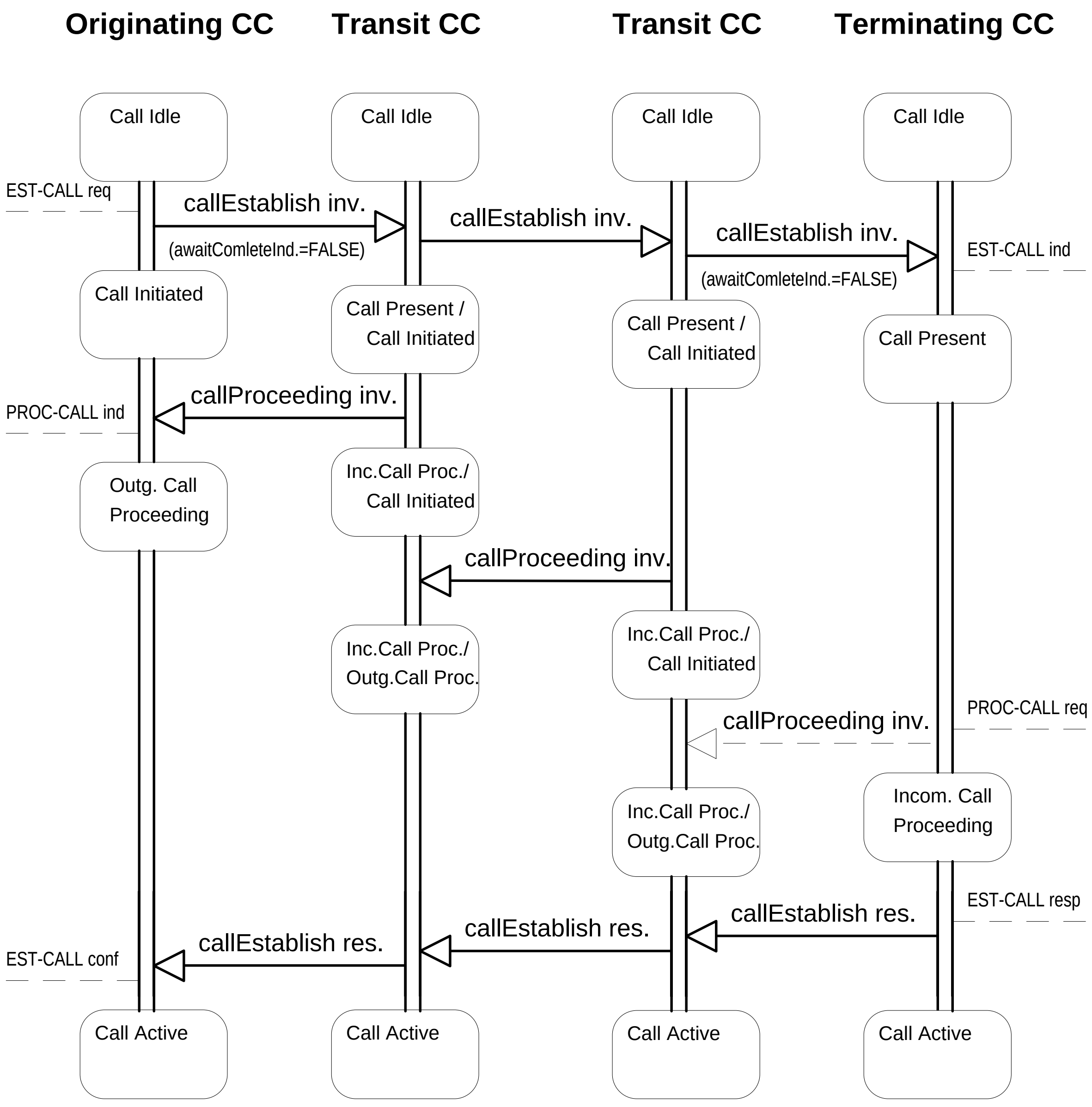

Figure E.1 - Example information flow for a successful call establishment using a two message sequence

E.2 Call Establishment using a three message sequence

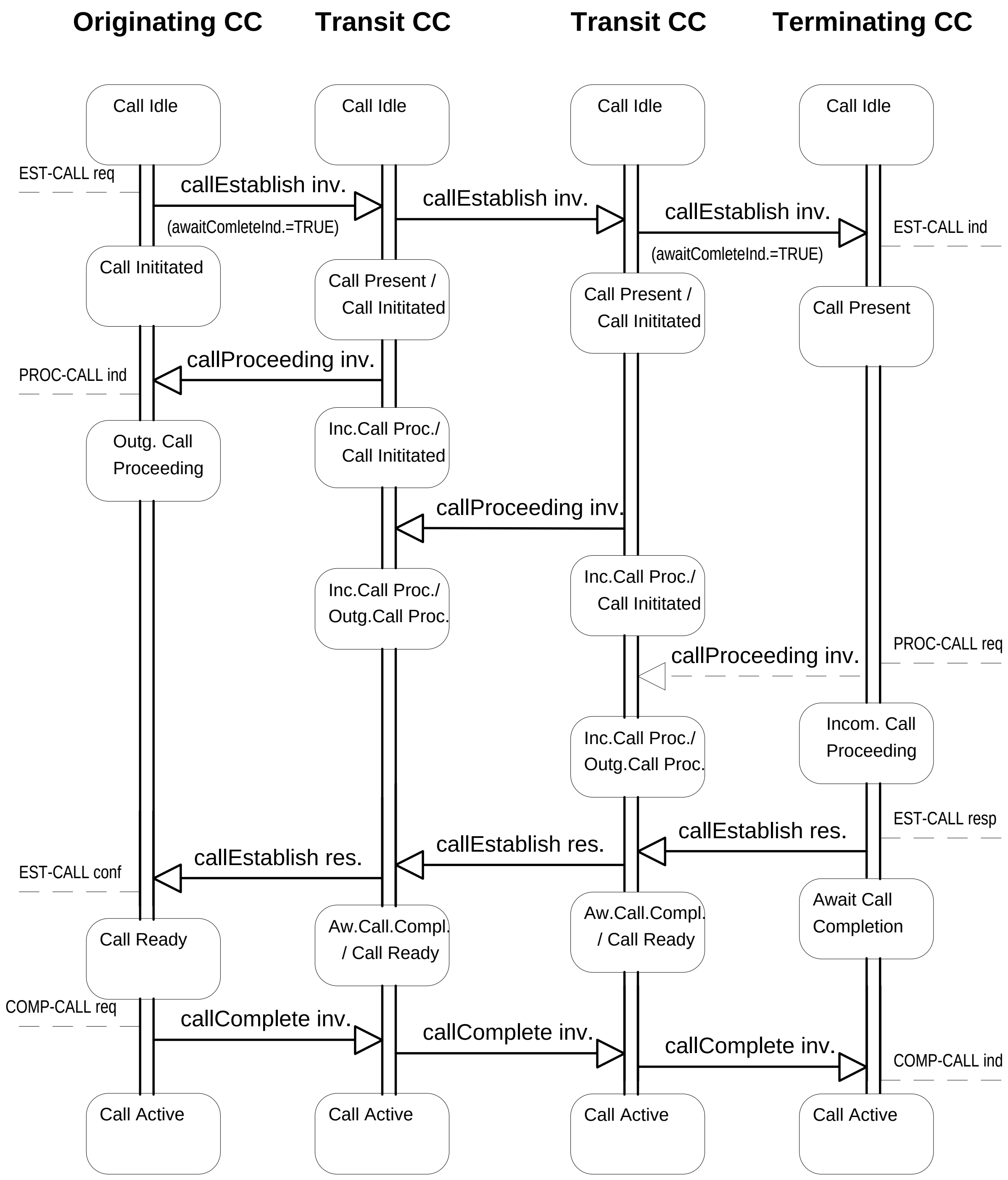

Figure E.2 - Example information flow for a successful call establishment using a three message sequence

E.3 Call Release

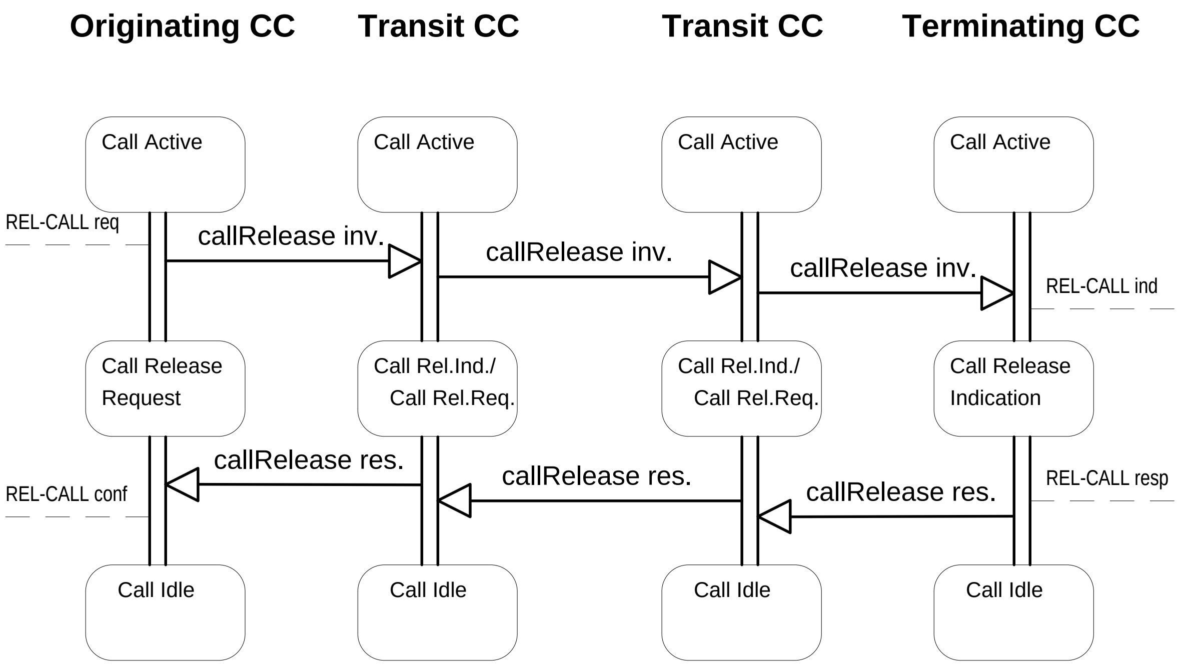

Figure E.3 - Example information flow for call release

Annex F

(informative)

Imported ASN.1 Definitions

Table F.1 is an extract from module Addressing-Data-Elements in ITU-T Rec. Q.932.

Table F.1 - Imported ASN.1 Definitions Addressing-Data-Elements

```
Addressing-Data-Elements
    {  itu-t recommendation q 932 addressing-data-elements (7)  }

DEFINITIONS ::=
BEGIN
EXPORTS PresentedAddressScreened,
        PartyNumber,
        PartySubaddress,
        ScreeningIndicator ;

    PresentedAddressScreened    ::= CHOICE  {
            presentationAllowedAddress      [0] IMPLICIT AddressScreened,
            presentationRestricted          [1] IMPLICIT NULL,
            numberNotAvailableDueToInterworking [2] IMPLICIT NULL,
            presentationRestrictedAddress       [3] IMPLICIT AddressScreened  }

    AddressScreened ::= SEQUENCE    {
                partyNumber     PartyNumber,
                screeningIndicator  ScreeningIndicator,
                partySubaddress     PartySubaddress     OPTIONAL    }

    PartyNumber     ::= CHOICE  {
                unknownPartyNumber  [0] IMPLICIT  NumberDigits,
                    -- the numbering plan is the default numbering plan of
                    -- the network.
                    -- it is recommended that this values is used.
                publicPartyNumber   [1] IMPLICIT PublicPartyNumber,
                    -- the numbering plan is according to Rec. E.163 and
                    -- E.164
                nsapEncodedNumber   [2] IMPLICIT NsapEncodedNumber,
                    -- ATM endsystem address encoded as an NSAP address
                dataPartyNumber [3] IMPLICIT    NumberDigits,
                    -- not used, value reserved
                telexPartyNumber    [4] IMPLICIT NumberDigits,
                    -- not used, value reserved
                privatePartyNumber  [5] IMPLICIT PrivatePartyNumber,
                nationalStandardPartyNumber [8]   IMPLICIT NumberDigits  }
                    -- not used, values reserved

        PublicPartyNumber   ::= SEQUENCE    {
                publicTypeOfNumber  PublicTypeOfNumber,
                publicNumberDigits  NumberDigits    }
        PrivatePartyNumber  ::= SEQUENCE    {
                privateTypeOfNumber PrivateTypeOfNumber,
                privateNumberDigits NumberDigits    }

        NumberDigits        ::= NumericString   (SIZE (1..20))

        PublicTypeOfNumber  ::= ENUMERATED  {
                unknown (0),
                    -- if used number digits carry prefix indicating type of
                    -- number according to national recommendations.
                internationalNumber (1),
                nationalNumber (2),
                networkSpecificNumber (3),
                    -- not used, value reserved
                subscriberNumber (4),
                abbreviatedNumber (6)   }
                    -- valid only for called party number at the outgoing access,
                    -- network substitutes appropriate number.
        PrivateTypeOfNumber ::= ENUMERATED  {
                    unknown (0),
                    level2RegionalNumber (1),
                    level1RegionalNumber (2),
                    pISNSpecificNumber (3),
                    localNumber (4),
                    abbreviatedNumber (6)       }

        NsapEncodedNumber   ::=     OCTET STRING (SIZE (1..20))

    PartySubaddress     ::= CHOICE  {
                    userSpecifiedSubaddress UserSpecifiedSubaddress,
                        -- not recommended
                    nSAPSubaddress  NSAPSubaddress  }
                        -- according to Rec. X.213.

        UserSpecifiedSubaddress ::= SEQUENCE    {
                    subaddressInformation   SubaddressInformation,
                    oddCountIndicator   BOOLEAN OPTIONAL    }
                        -- used when the coding of subaddress is BCD
        NSAPSubaddress  ::= OCTET STRING    (SIZE(1..20))
                        -- specified according to X.213. some networks may limit
                        -- the subaddress value to some other length, e.g. 4 octets.
            SubaddressInformation   ::= OCTET STRING    (SIZE(1..20))
                        -- coded according to user requirements. some networks
                        -- may limit the subaddress value to some other length,
                        -- e.g. 4 octets.


    ScreeningIndicator  ::= ENUMERATED  {
                    userProvidedNotScreened (0),
                        -- number was provided by a remote user terminal
                        -- equipment, and has been screened by a network that
                        -- is not the local public or the local private network.
                    userProvidedVerifiedAndPassed (1),
                        -- number was provided by a remote user terminal
                        -- equipment (or by a remote private network), and has
                        -- been screened by the local public or the local private
                        -- network.
                    userProvidedVerifiedAndFailed (2),
                        -- not used, value reserved.
                    networkProvided (3)     }
                        -- number was provided by local public or local private
                        -- network.

END     -- of Addressing-Data-Elements
```

Annex G

(informative)

Object identifiers defined in this Standard

This annex lists the object identifier values assigned in this Standard and data types, values and macros that are exported from any modules identified by those values. All the object identifiers in this Standard are defined using the ITU-T object identifier tree. This means that each object identifier value is assigned in the tree:

ccObjectIdTree ::= itu-t recommendation q 2981

Table G.1 lists the module number values and the data types, values and Macros which are exported from these modules.

Table G.1 - ASN.1 Module Object identifiers used in this Standard

Object Identifier	Reference	Notes
{ ccObjectIdTree cc-operations (1) }	Table 7	Exports: CcOperations, callSegmentId
{ ccObjectIdTree cc-operations-definitions (2) }	Table 7	
{ ccObjectIdTree cc-operations-errors (3) }	Table 7	
{ ccObjectIdTree call-control-object-super-class (4) }	Table 8	Exports: CALLCONTROLOBJECTCLASS
{ ccObjectIdTree call-object-class-definitions (5) }	Table 9	Exports: call, localPartyEP, remotePartyEP, directCallAssociation, remoteCallAssociation, serviceComponent, BearerId
{ ccObjectIdTree cc-object-classes (6) }	Table 9	

Annex H
(informative)

Bibliography

The following material, though not specifically referenced in the body of this Standard (or not publicly available), gives supporting information.

DEG/SPS-05134	Broadband Integrated Services Digital Network (B-ISDN); Digital Subscriber Signalling System No. two (DSS2) and Broadband QSIG (B-QSIG) protocols; Generic concepts for the support of multiconnection calls in a separated environment; Part 1: Protocol specification