Draft EN 301 140-1 V1.1.1 (1998-03)

European Standard (Telecommunications series)

Intelligent Network (IN);
Intelligent Network Application Protocol (INAP);
Capability Set 2 (CS2);
Part 1: Protocol specification

European Telecommunications Standards Institute

Reference
DEN/SPS-03038-1 (ak090ico.PDF)

Keywords
CS2, IN, INAP, protocol

ETSI Secretariat

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Internet
secretariat@etsi.fr
http://www.etsi.fr
http://www.etsi.org

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETR 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available **free of charge** from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http: //www.etsi.fr/ipr).

Pursuant to the ETSI Interim IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETR 314 (or the updates on http: //www.etsi.fr/ipr) which are, or may be, or may become, essential to the present document.

Foreword

This European Standard (Telecommunications series) has been produced by ETSI Technical Committee Signalling Protocols and Switching (SPS), and is now submitted for the Public Enquiry phase of the ETSI standards Two-step Approval Procedure.

This EN is part 1 of a multi-part standard covering Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 2 (CS2) 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)";

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

Part 5: "Distributed Functional Plane (DFP)".

The structure of the present document follows that of the proposed ITU-T Recommendation Q.1228 [28] rather than that usual for an ETSI deliverable. It is probable that, on publication, clauses will be removed from the present document and substituted with references to that Recommendation. However, at the time of release of the present document for Public Enquiry, ITU-T Recommendation Q.1228 [28] was not publicly available.

Proposed national transposition dates	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

1 Scope

The present document is based on draft ITU-T Recommendation Q.1228 [28]. It provides major modifications and further requirements to this base document.

This first part of the present document defines the Intelligent Network Application Protocol (INAP) required for support of Capability Set 2 (CS2). It supports interactions between the following five Functional Entities (FEs), as defined in the Intelligent Network (IN) functional model:

- Service Switching Function (SSF);
- Service Control Function (SCF);
- Service Data Function (SDF);
- Call Unrelated Service Function (CUSF);
- Specialized Resource Function (SRF).

The further development of the INAP for each of the Integrated Services Digital Network (ISDN) and Public Switched Telecommunications Network (PSTN) and Public Land Mobile Networks (PLMN) is covered by the present document, which is intended as a guide to implementers and network operators to ensure interworking between different manufacturers' equipment for the following IN CS2 defined interfaces:

- SCF-SSF;
- SCF-CUSF;
- SCF-SRF;
- SCF-SDF;
- SCF-SCF, and
- SDF-SDF).

As the present document is intended to facilitate the early introduction of IN in the existing ISDN/PSTN, only simple solutions are assumed for solving the service interaction problems between IN and ISDN/PSTN.

NOTE: More sophisticated solutions for the service interactions between IN and the ISDN/PSTN environment should be studied in the scope of future versions of INAP and the ISDN/PSTN signalling standards.

2 References

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case 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.

2.1 Normative references

[1] ETR 186-2: "Intelligent Network (IN); Interaction between IN Application Protocol (INAP) and Integrated Services Digital Network (ISDN) signalling protocols; Part 2: Switching signalling requirements for IN Capability Set 2 (CS2) service support in a Narrowband ISDN (N-ISDN) environment".

[2] ETS 300 008-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; Message Transfer Part (MTP) to support international interconnection; Part 1: Protocol specification [ITU-T Recommendations Q.701 (1993), Q.702 (1988), Q.703 to Q.706 (1993), modified]".

[3] ETS 300 009-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; Signalling Connection Control Part (SCCP) (connectionless and connection-oriented class 2) to support international interconnection; Part 1: Protocol specification [ITU-T Recommendations Q.711 to Q.714 and Q.716 (1993), modified]".

[4] ETS 300 121: "Integrated Services Digital Network (ISDN); Application of the ISDN User Part (ISUP) of CCITT Signalling System No.7 for international ISDN interconnections (ISUP version 1)".

[5] ETR 186-2: CS2 Signalling Interworking Requirements

[6] EN 300 196-1: "Integrated Services Digital Network (ISDN); Generic functional protocol for the support of supplementary services; Digital Subscriber Signalling System No. one (DSS1) protocol; Part 1: Protocol specification".

[7] ETS 300 287-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; Transaction Capabilities (TC) version 2; Part 1: Protocol specification [ITU-T Recommendations Q.771 to Q.775 (1993), modified]".

[8] ETS 300 348: "Intelligent Network (IN); Physical plane for intelligent network Capability Set 1 (CS1) [ITU-T Recommendation Q.1215 (1993)]".

[9] EN 300 356-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; ISDN User Part (ISUP) version 3 for the international interface; Part 1: Basic services [ITU-T Recommendations Q.761 to Q.764 (1997), modified]".

[10] ETS 300 374-1: "Intelligent Network (IN); Intelligent Network Capability Set 1 (CS1); Core Intelligent Network Application Protocol (INAP); Part 1: Protocol specification".

[11] EN 300 403-1: "Integrated Services Digital Network (ISDN); Digital Subscriber Signalling System No. one (DSS1) protocol; Signalling network layer for circuit-mode basic call control; Part 1: Protocol specification [ITU-T Recommendation Q.931 (1993), modified]".

[12] EN 301 070-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; ISDN User Part (ISUP) version 3 interactions with the Intelligent Network Application Part (INAP); Part 1: Protocol specification [ITU-T Recommendation Q.1600 (1997), modified]".

[13] GSM 04.08: "Digital cellular telecommunications system (Phase 2+); Mobile radio interface layer 3 specification (GSM 04.08)".

[14] GSM 09.02: "Digital cellular telecommunications system (Phase 2+); Mobile Application Part (MAP) specification (GSM 09.02)".

[15] ISO 639 (1988): "Code for the representation of names of languages".

[16] ISO 9545 (1989): "Information technology - Open Systems Interconnection –Application Layer structure".

[17] ITU-T Recommendation Q.71: "ISDN circuit mode switched bearer services".

[18] ITU-T Recommendation Q.77: "

[19] ITU-T Recommendation Q.700: "Introduction to CCITT Signalling System No.7".

[20] ITU-T Recommendation Q.710: "Signalling System No.7 - Simplified MPT version of small systems".

[21] ITU-T Recommendation Q.762: "General function of messages and signals of the ISDN user part of signalling system no.7".

[22] ITU-T Recommendation Q.763: "Formats and codes of the ISDN user part of Signalling System No.7".

[23] ITU-T Recommendation Q.767: "Application of the ISDN user part of CCITT Signalling System No.7 for international ISDN interconnections

[24] ITU-T Recommendation Q.850: "Usage of cause and location in the digital subscriber signalling system no.1 and the signalling system no.7 ISDN user part".

[25] ITU-T Recommendation Q.932: "Digital subscriber Signalling System No.1 (DSS 1) - Generic procedures for the control of ISDN supplementary services".

[26] ITU-T Recommendation Q.1224 (must be dated): "Distributed functional plane for intelligent network CS2".

[27] ITU-T Recommendation Q.1225: "Physical plane for intelligent network CS2".

[28] ITU-T Recommendation Q.1228: "Interface ITU-T Recommendation for intelligent network CS2".

[29] ITU-T Recommendation Q.1290: "Glossary of terms used in the definition of intelligent networks".

[30] ITU-T Recommendation Q.1400: "Architecture framework for the development of signalling and organization, administration and maintenance protocols using OSI principles".

[31] CCITT Recommendation X.208: "Specification of Abstract Syntax Notation One (ASN.1)".

[32] CCITT Recommendation X.209: "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)".

[33] CCITT Recommendation X.219: "Remote operations: Model, notation and service definition".

[34] CCITT Recommendation X.229: "Remote operations: Protocol specification".

[35] ITU-T Recommendation X.252: "

[36] ITU-T Recommendation X.500: "Information technology – Open systems Interconnection – The directory: Overview of concepts, models and services".

[37] ITU-T Recommendation X.501 | ISO/IEC 9594-2: "Information Technology – Open Systems Interconnection – The directory: Models".

[38] ITU-T Recommendation X.509: "Information technology - Open Systems Interconnection - The Directory: Authentication framework".

[39] ITU-T Recommendation X.511 (1993) third ed | ISO/IEC 9594-3: "Information technology – Open Systems Interconnection – The directory: Abstract service definition".

[40] ITU-T Recommendation X.518 (11/93): "Information technology - Open Systems Interconnection - The directory: Procedures for distributed operation".

[41] ITU-T Recommendation X.519 | ISO/IEC 9594-5: "Information technology – Open Systems Interconnection – The directory: Protocol specifications".

[42] ITU-T Recommendation X.525 | ISO/IEC 9594-9: "Information technology - Open System Interconnection - The Directory: Replication".

[43] ITU-T Recommendation X.680 ASN.1: "Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[44] ITU-T Recommendation X.690: "ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

[45] ITU-T Recommendation X.831: "Information technology – Open Systems Interconnection – Generic upper layers security: Security Exchange Service Element (SESE) service definition".

[46] ITU-T Recommendation X.832: "Information technology – Open Systems Interconnection – Generic upper layers security: Security Exchange Service Element (SESE) protocol specification".

[47] ITU-T Recommendation X.880 | ISO/IEC 9072-1: "Information technology – Remote Operations: Concepts, model and notation".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AC	Application Context
ACI	Access Control Information
AE	Application Entity
AEI	Application Entity Invocation
APDU	Application Protocol Data Unit
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
BCSM	Basic Call State Model
BCUSM	Basic Call Unrelated State Model
BPIM	Basic Primitive Interface Model
C-ALG	Data confidentiality algorithm
CBC	Connectionless Bearer Control
CCF	Call Control Function
CON	CONnect message
CPH	Call Party Handling
CS	
CS1	Capability Set 1
CS2	Capability Set 2
CSA	
CSCV	Call Segment Connection View
CSACV	Call segment association
CUSF	Call Unrelated Service Function
CUSME	CUSF Management
CUUI	Call Unrelated User Interaction
CVS	
DAP	Directory Access Protocol
DES	
DISP	Directory Information Shadowing Protocol
DIT	(See X.501)
DP	Detection Point
DSA	Directory System Agent
DSS1	Digital Subscriber Signalling System No. One
DUA	Directory User Agent
EDP	Event Detection Point
EDP-N	Event Detection Point - Notification
EDP-R	Event Detection Point - Request
FCI	Furnish Charging Information
FE	Functional Entity
FEAM	Functional Entity Access Manager
FSM	Finite State Model
GT	Global Title
GULS	Generic Upper Layers Security
I-ALG	data Integrity ALGorithm
IBI	Intra-BCSM Interface
ICA	
ID	IDentifier
IE	Information Element

IH	Interrupt Handler
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IP	Intelligent Peripheral
ISDN	Integrated Services Digital Network
ISPBX	Integrated Services Private Branch eXchange
ISUP	ISDN User Part
K-ALG	Key management ALGorithm
LE	Local Exchange
MAC	Multiple Association Control
MACF	Multiple Association Control Function
MSC	Message Sequence Chart
MTP	Message Transfer Part
O-ALG	One-way function ALGorithm
O-BCSM	Originating BCSM
P&C	Prompt and Collect
PA	Play Announcement
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PE	Physical Entity
PIA	Point In Association
PIC	Personal Identification Code
PLMN	Public Land Mobile Network
PSTN	Public Switched Telecommunication Network
QOP	Quality of Protection
RCO	Resource Control Object
ROS	Remote Operations Service
ROSE	ROS Element
RRB	Radio Regulations Board
SACF	Single Association Control Function
SAO	Single Association Object
SCCP	Signalling Connection Control Part
SCF	Service Control Function
SCME	SCF Management Entity
SCP	Service Control Point
SCSM	SCF Call State Model
SDF	Service Data Function
SDL	System Description Language
SDP	Service Data Point
SESE	Security Exchange Service Element
SL	Service Logic
SLP	Service Logic Program
SLPI	Service Logic Program Instance
SMF	Service Management Function
SPKM	Simple Public Key GSS-API Mechanism
SRF	Specialized Resource Function
SRME	SRF Management Entity
SRSM	SRF Call State Model
SS7	Signalling System no. 7
SSF	Service Switching Function
SSME	SSF Management Entity
SSN	Sub-System Number
SSP	Service Switching Point
STUI	Service To User Information
T-BCSM	Terminating BCSM
TC	Transaction Capabilities
TCAP	Transaction Capabilities Application Part
TDP	Trigger Detection Point
TDP-N	Trigger Detection Point - Notification
TDP-R	Trigger Detection Point - Request
USI	User Service Interface
UTSI	User To Service Information

4 Interface ITU-T Recommendation for telecommunication services

4.1 General

4.1.1 Definition methodology

The definition of the protocol can be split into three sections:

- the definition of the Single Association Control Function (SACF)/Multiple Association Control Function (MACF) rules for the protocol;
- the definition of the operations transferred between entities;
- the definition of the actions taken at each entity.

The SACF/MACF rules are defined in prose. The operation definitions are in Abstract Syntax Notation One (ASN.1), see CCITT Recommendations X.208 [31], ITU-T Recommendation X.680 [43]), and the actions are defined in terms of state transition diagrams. Further guidance on the actions to be performed on receipt of an operation can be gained from the description of the relevant information flow in ITU-T Recommendation Q.1224 [26].

The Intelligent Network Application Protocol (INAP) is a ROS Element (ROSE) user protocol (see CCITT Recommendation X.219 [33] and CCITT Recommendation X.229 [34]). The ROSE protocol is contained within the component sublayer of Transaction Capabilities Application Part (TCAP) (see ETS 300 287-1 [7] toETS 300 287-1 [7]) and Digital Subscriber Signalling System No One (DSS1) (ITU-T Recommendation Q.932 [25]). At present the ROSE Application Protocol Data Units (APDUs) are conveyed in transaction sublayer messages in Signalling System no. 7 (SS7) and in the EN 300 403-1 [11] REGISTER, FACILITY and call control messages in DSS1. Other supporting protocols may be added at a later date.

The INAP (as a ROSE user) and the ROSE protocol have been specified using ASN.1 (see ITU-T Recommendation X.680 [43]). The encoding of the resulting Protocol Data Units (PDUs) should use the Basic Encoding Rules (see ITU-T Recommendation X.690 [44]).

4.1.2 Example physical scenarios

The protocol will support any mapping of functional to Physical Entities (PEs). It is the responsibility of network operators and equipment manufacturers to decide how to collocate FE's to the best possible advantage as this may vary between manufacturers and between network operators. Therefore the protocol is defined assuming maximum distribution (i.e. one PE per FE).

The figures depicted in this subclause show how INAP would be supported in an SS7 network environment. This does not imply that only SS7 may be used as the network protocol to support INAP.

The interface between remotely located SCF and SDF will be INAP using TCAP which in turn uses the services of the connectionless Signalling Connection Control Part (SCCP) and Message Transfer Part (MTP) (see figure 1). The SDF is responsible for any interworking to other protocols to access other types of networks.

When TCAP appears in one of the following figures, it shall be understood as representing the TCAP functionalities associated with a single dialogue and transaction (as opposed to a TCAP entity).

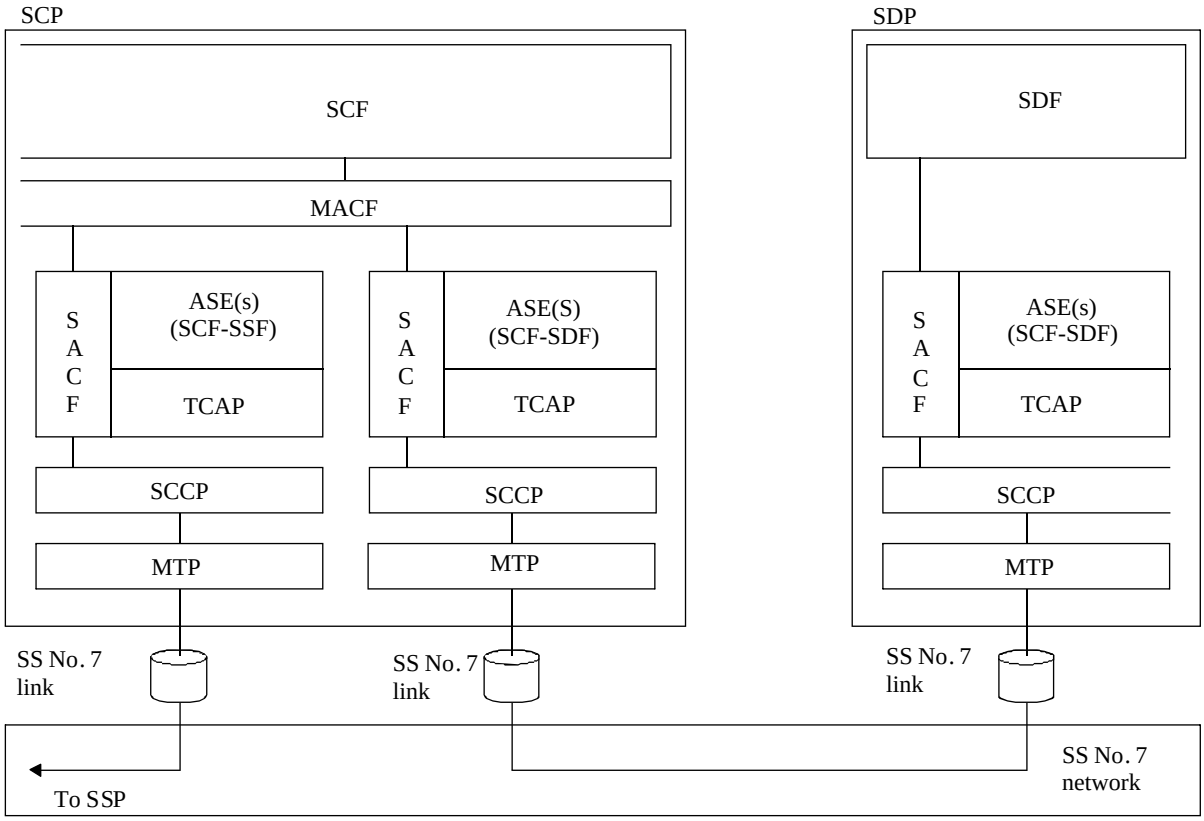


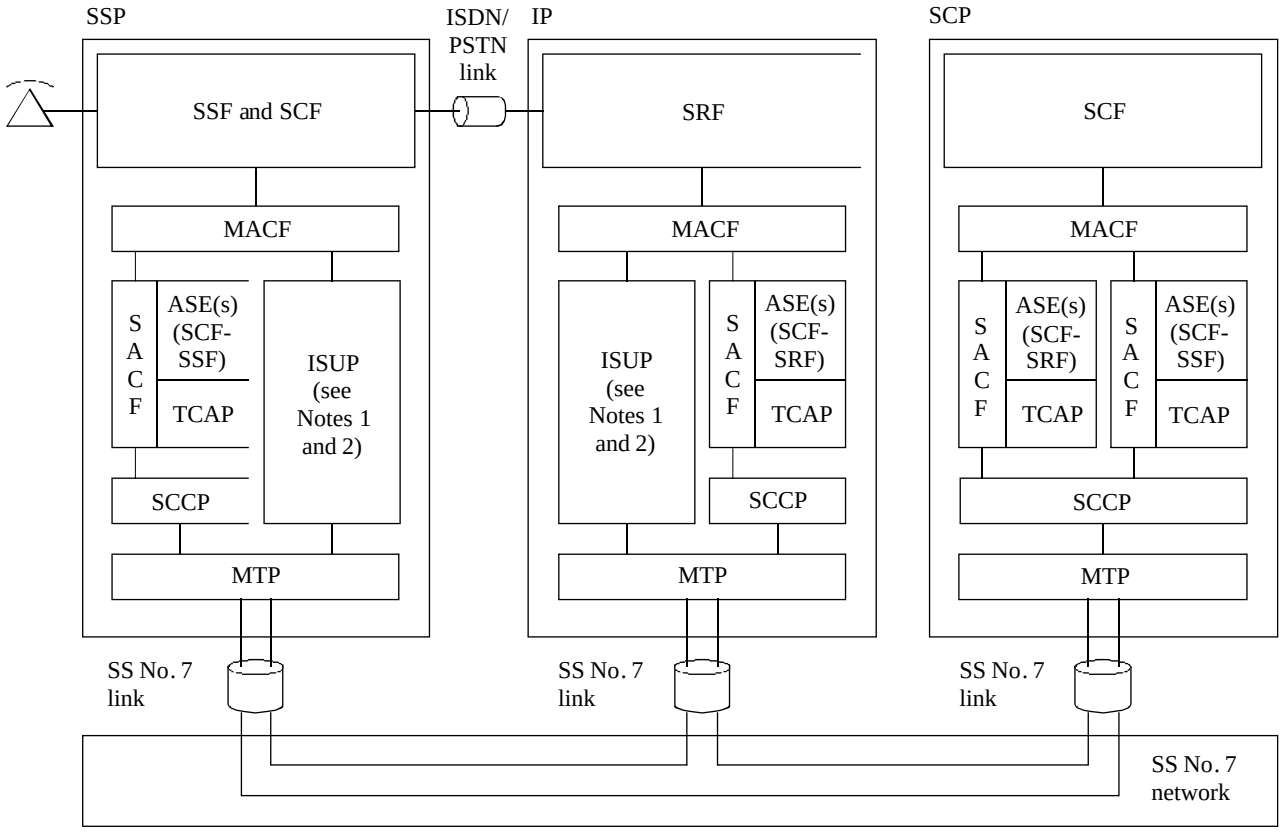
Figure 3-1: Physical interface between Service Control Point (SCP) and Service Data Point (SDP)

If segmentation and re-assembly of INAP messages is required on the SCF-to-SDF interface (and on other interfaces, if needed) due to the length of messages, the segmentation and re-assembly procedure for SCCP connectionless messages, as specified in ETS 300 009-1 [3], should be used.

A number of example scenarios have been identified for support of the SCF, Service Switching Function (SSF) and SRF functional entities as PE. These are illustrated as figures 2 to 6. Each example is characterized by:

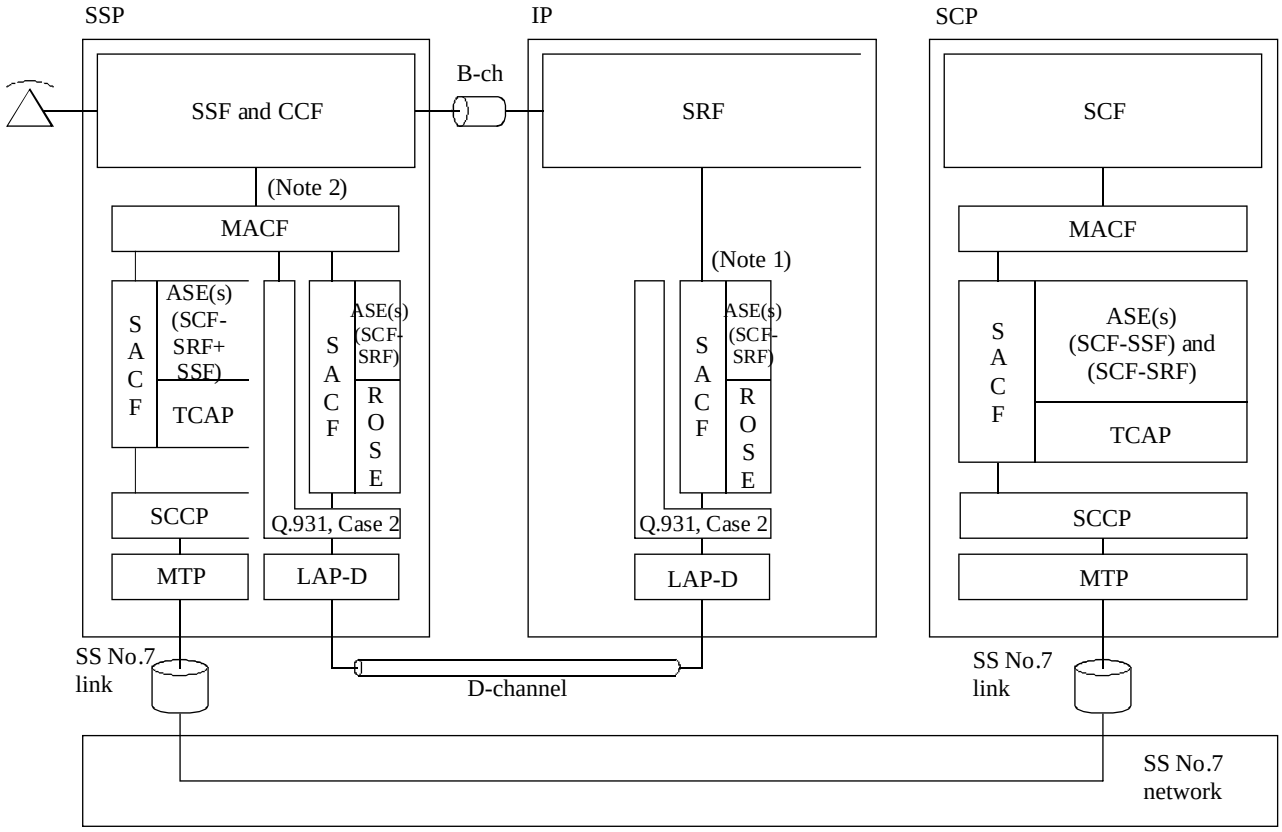
i) the method to support SCF-SRF relationship; and

ii) the type of signalling system between SSF and SRF.



NOTE1: Transfer of correlation information needs to be supported. This may be supported in ISUP without introducing new ISUP parameter.
NOTE 2: Other signalling systems may be used.
NOTE 3: The IP can be integrated into a local exchange, or indirectly attached via a local exchange to the Service Switching Point (SSP) that isinteracting with the SCP.

3-2: Example architecture for supporting SRF, Case 1 (SRF in IP connected to SSP and accessed by SCP through direct SS7 connection)



NOTE 1: Info flows between SCF and SRF are supported by this (ROSE) entity.
NOTE 2: Relay function is provided either by MACF or by application process at SSP.

Figure 3-3: Example architecture for supporting SRF, Case 2 (SRF in IP connected to SSP and accessed by SCP through D-channel via SSP)

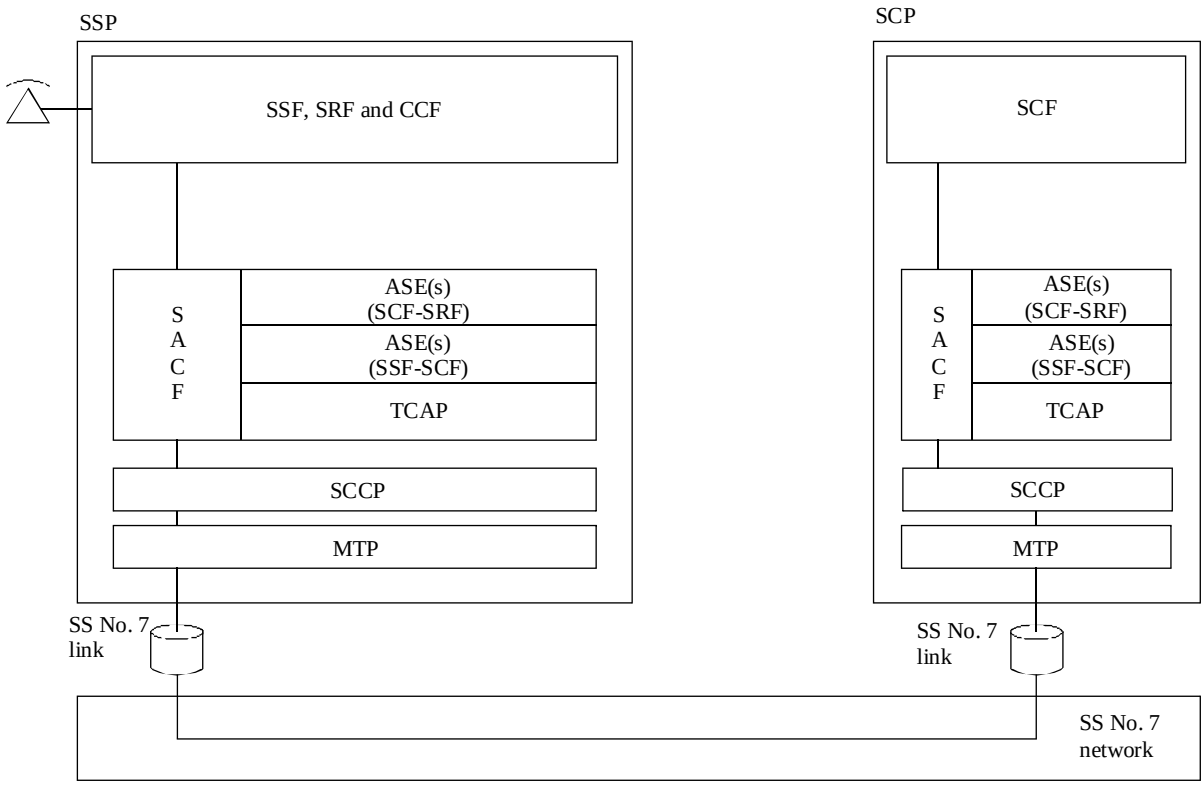
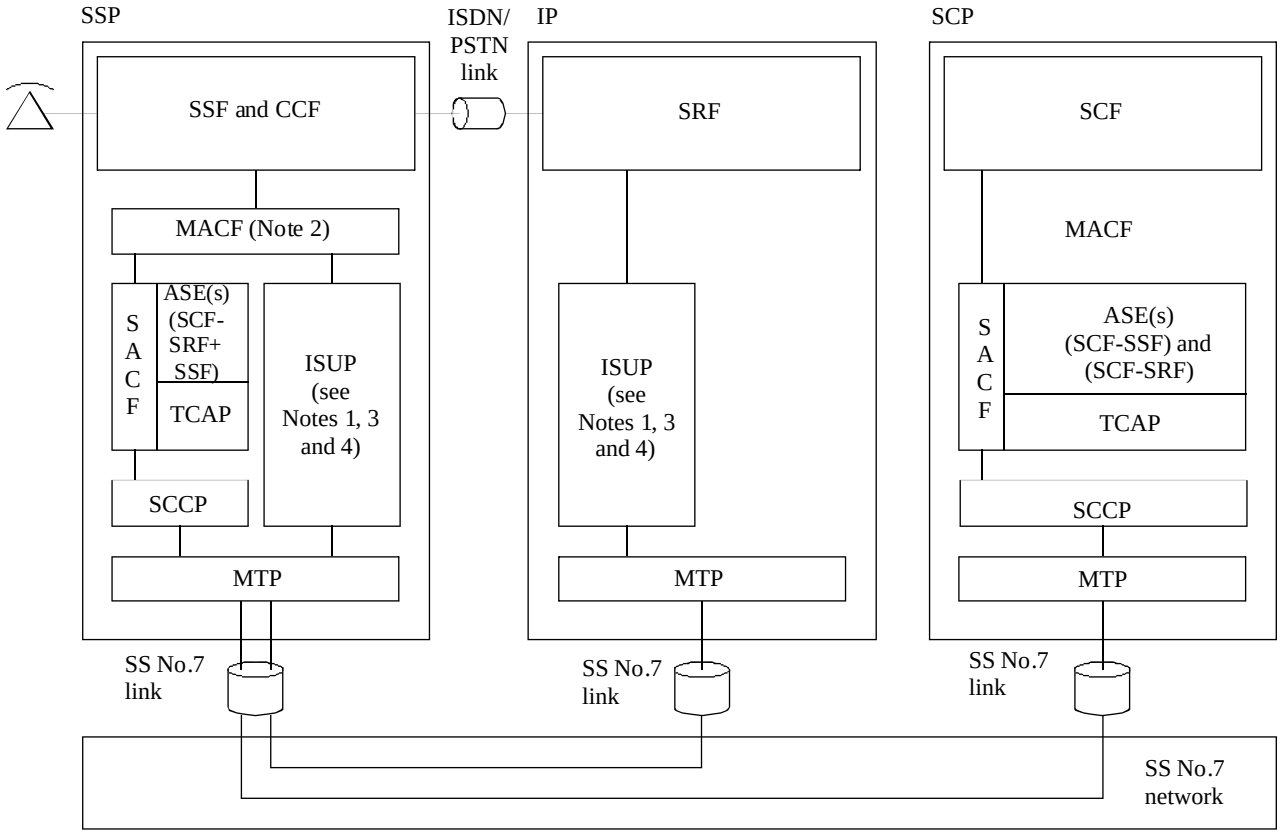
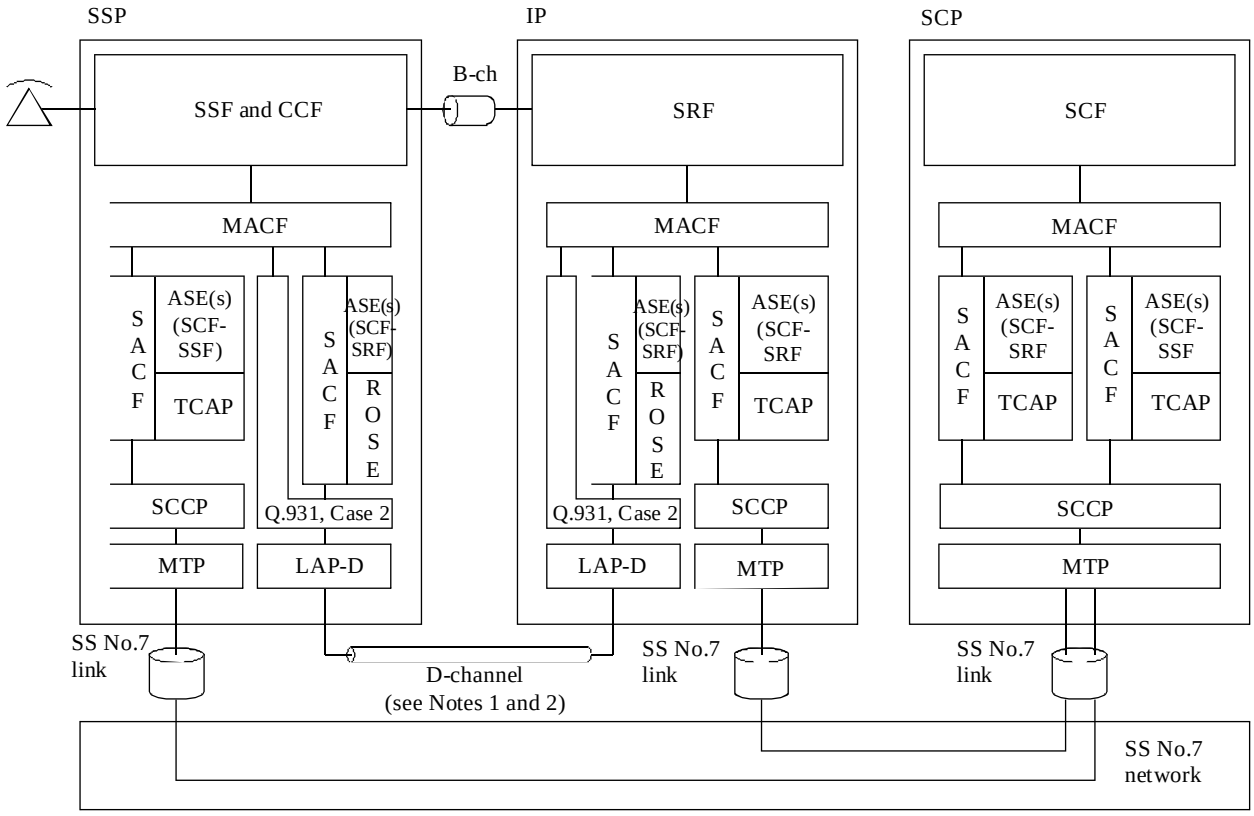


Figure 3-4: Example architecture for supporting SRF, Case 3 (SRF in SSP and accessed via AP of SSP)



NOTE 1: Info flows between SCF and SRF as well as connection control are directly supported by ISUP.
NOTE 2: Relay function is provided either by MACF or by application process at SSP.
NOTE 3: Assumes that ISUP porvides a means to transport ROSE information.
NOTE 4: Other signalling systems may be used.

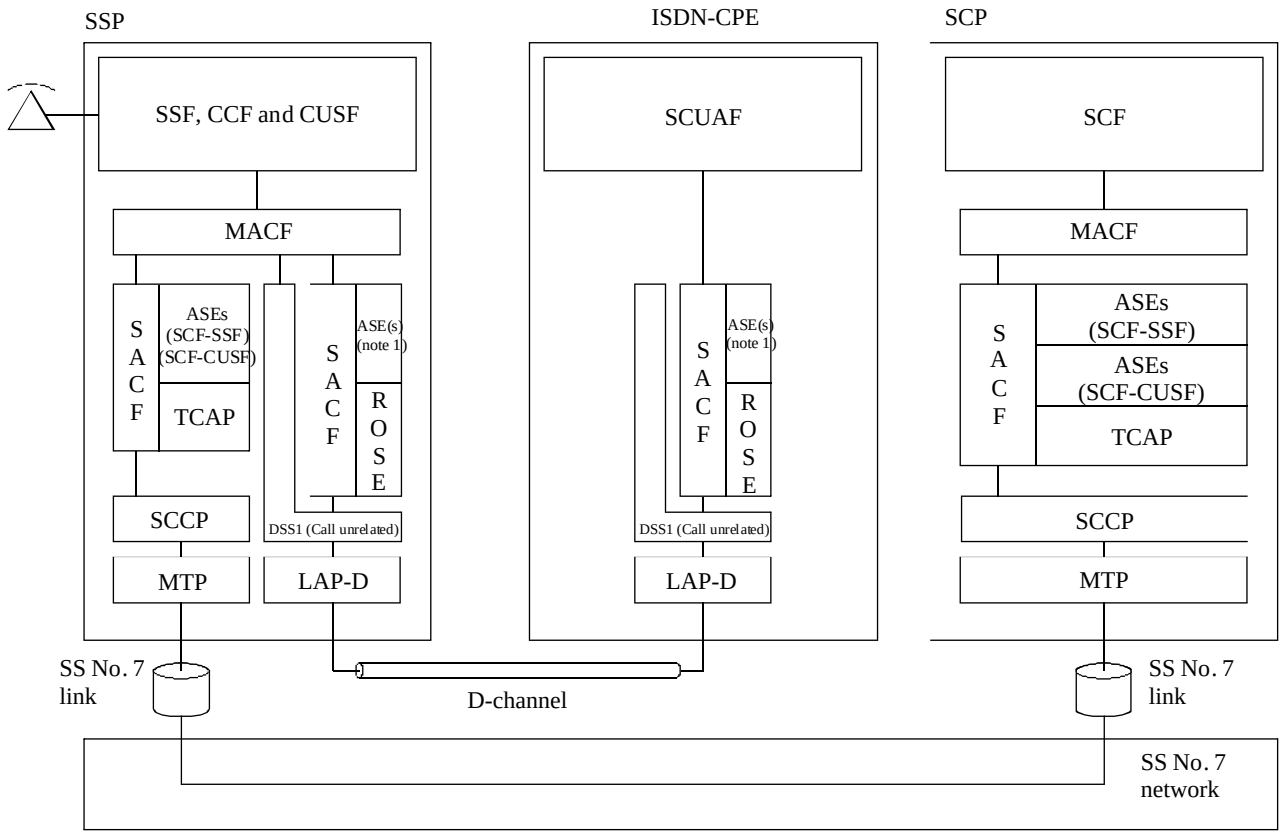
Figure 3-5: Example architecture for supporting SRF, Case 4 (SRF in IP connected to SSP and accessed by SCP through ISUP via SSP)



NOTE 1: Transfer of correlation information needs to be supported.
NOTE 2: Other signalling systems may be used.

Figure 3-6: Example architecture for supporting SRF, Case 5 (SRF in IP connected to SCP and SSP and accessed via both SS7 and D-channel respectively)

As there may be several configurations for the SRF mapping, this figure does not mention all possible architecture but a possible stack for the SSP with the CUSF, the CCF and the SSF.



NOTE 1: These Application Service Elements (ASEs) are defined on the UNI for DSS1 supplementary services.
NOTE 2: IP is omitted for simplicity, however SSP has ISUP/DSSI link to IP.

Figure 3-7: Example architecture focusing on the CUSF (CUSF being mapped to SSP with SSF and CCF)

Table 3-1 summarizes the selection of features for each figure.

Table 3-1

Type of signalling system between SSF and SRF	**Method to support SCF-SRF relationship**	
	Direct TCAP link	**Relay via SSP**
ISUP	Figure 3-2 (note 1)	Figure 3-5 (note 4)
DSS1	Figure 3-6 (note 5)	Figure 3-3 (note 2)
Implementation dependent	As figures 3-2 or 3-6 but with implementation dependent SCP-IP interface	Figure 3-4 (note 3)

Additional information related to each figure:

NOTE1: Figure 3-2: All associations are supported by SS7, either TCAP or ISUP. In this case the IP is one of the network nodes.

NOTE 2: Figure 3-3: IP can be accessed by DSS1 only. The IP can be a PE residing outside the network.

NOTE 3: Figure 3-4: SSP supports both CCF/SSF and SRF. The handling of SRF by SCF could be the same as the of figure 3-3.

NOTE 4: Figure 3-5: IP can be accessed by ISUP only. The handling of SRF by SCF could be the same as that of figure 3-3.

NOTE 5: Figure 3-6: The handling of SRF by SCF could be the same as that of figure 3-2. Other types of signalling systems could be used.

4.1.2.1 "SCF - External SRF" Communication in the Relay Case

In the Relay case, when the SCF uses the *ConnectToResource* operation to connect to an External SRF, the SCF and the SRF embed the "User Interaction" operations exchanged with each other using the operations:

SendSTUI, *ReportUTSI* and *RequestReportUTSI*.

In this case, it is necessary to affect a new value of the *serviceIndicator* parameter for the "External SRF connection":

SRF_Connection.

As in Capability Set 1 (CS1), the "External SRF connection" is not modelled at the SSF level. Once receiving the *SendSTUI* (resp. *RequestReportUTSI*) operation from the SCF with a *serviceIndicator* parameter value set to *SRF_connection*, the SSF checks only this parameter to decide that this operation is related to the "SCF-External SRF communication". The same processing applies for the *reportUTSI* operation in the "SRF to SCF" direction.

At a time, only one party (Called Party or Calling Party) can be connected to the SRF.

The Message Sequence Chart (MSC) in figure 3-8 illustrates the User Interaction in the Relay case. Further information the SRF procedures can be found in clause 13.

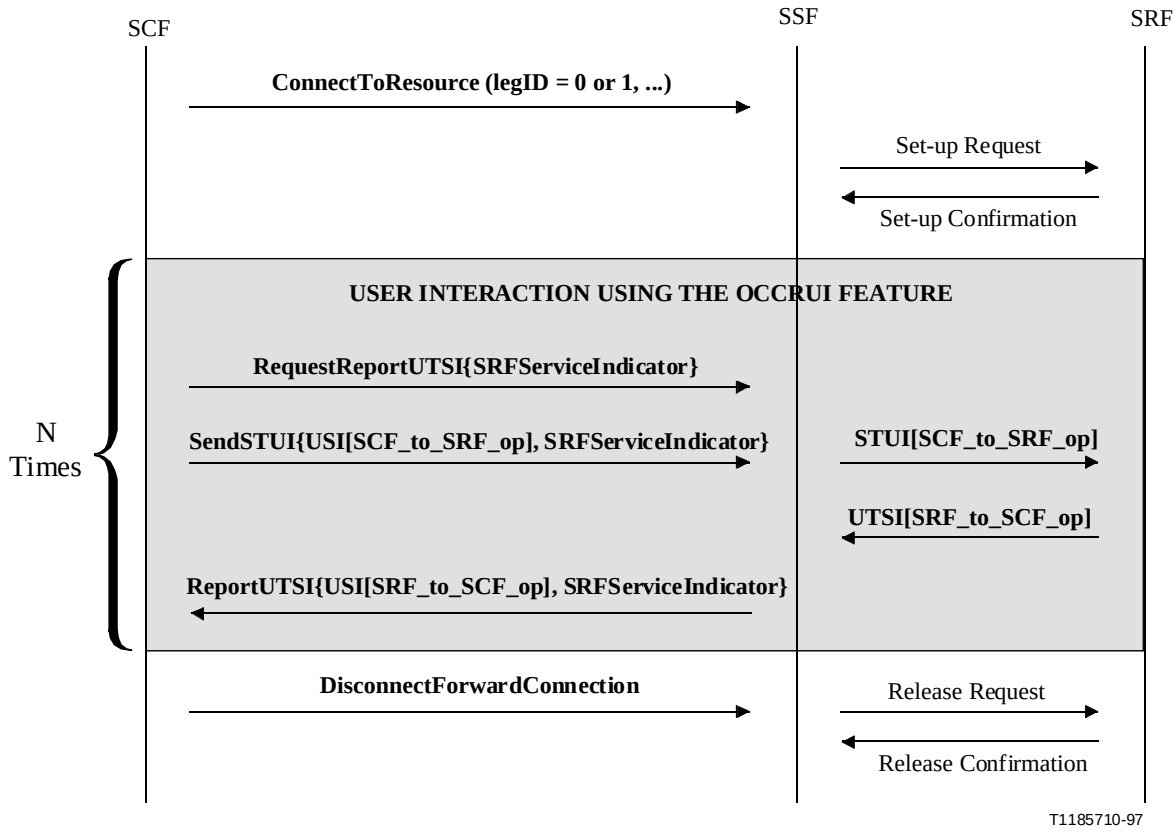


Figure 3-8: User Interaction in the Relay Case

4.1.3 INAP protocol architecture

Many of the terms used in this clause are based on the OSI application layer structure as defined in ISO 9545 [16].

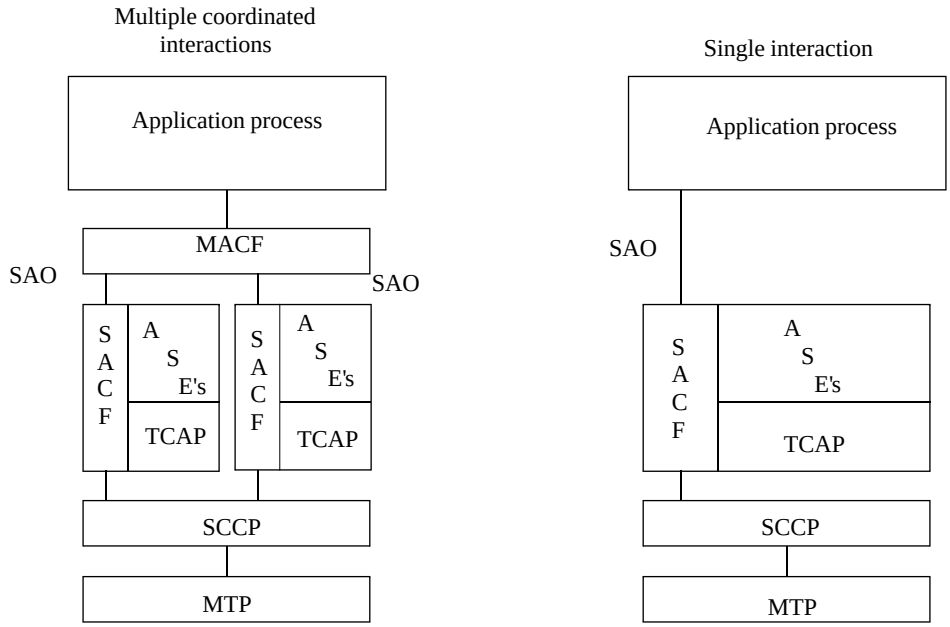
The INAP protocol architecture can be illustrated as shown in figure 3-9.

A PE has either single interactions (case a) or multiple co-ordinated interactions (case b) with other PE.

In case a, SACF provides a co-ordination function in using ASE's, which includes the ordering of operations supported by ASE(s), (based on the order of received primitives). The Single Association Object (SAO) represents the SACF plus a set of ASE's to be used over a single interaction between a pair of PE's.

In case b, MACF provides a co-ordinating function among several SAO's, each of which interacts with an SAO in a remote PE.

Each ASE supports one or more operations. Description of each operation is tied with the action of corresponding FE modelling (see ITU-T Recommendation Q.1224 [26] and clause 11-18 of the present document). Each operation is specified using the OPERATION macro described in figure 3-10.



SACF	Single association control function
MACF	Multiple association control function
SAO	Single association object
ASE	Application service element
INAP	Intelligent network application protocol

NOTE – INAP is the collection of specifications of all in ASE's.

Figure 3-9: INAP protocol architecture

The use of the Application Context (AC) negotiation mechanism as defined in the ITU-T Recommendation Q.77 X-Series (*Transaction capabilities application part*) [18] allows the two communicating entities to identify exactly what their capabilities are and also what the capabilities required on the interface should be. This should be used to allow evolution through Intelligent Network (IN) capability sets.

If the indication of a specific AC is not supported by a pair of communicating FEs, some mechanism to pre-arrange the context needs to be supported.

4.1.3.1 INAP signalling congestion control for SS7

The same type of procedure shall apply as defined for ISDN User Part (ISUP) signalling congestion control. The INAP procedures for signalling congestion control shall as far as possible be aligned with the ISUP signalling congestion control procedures as specified (ITU-T Recommendation Q.767 [23], clause D.2.11), i.e. on receipt of N-PCSTATE indication primitive with the information "signalling point congested" from SCCP, the INAP shall reduce the traffic load (e.g. InitialDP, CreateCallSegmentAssociation, or InitiateCallAttempts) in the affected direction in several steps.

The above procedure may only apply to traffic which uses MTP Point Code addressing in the affected direction.

4.1.4 INAP addressing

SCCP Global Title (GT) and MTP point code addressing (see ITU-T Recommendations Q.710-Series (*Signalling connection control part*) [20] and ITU-T Recommendation Q.700-Series (*Message transfer part*) [19]) ensure that PDUs reach their physical destination (i.e. the correct point code) regardless of which network it is in.

Within a node, it is the choice of the network operator/implementor as to which Sub-System Number (SSN) or SSNs are assigned to INAP.

Regardless of the above, any addressing scheme supported by the SCCP may be used.

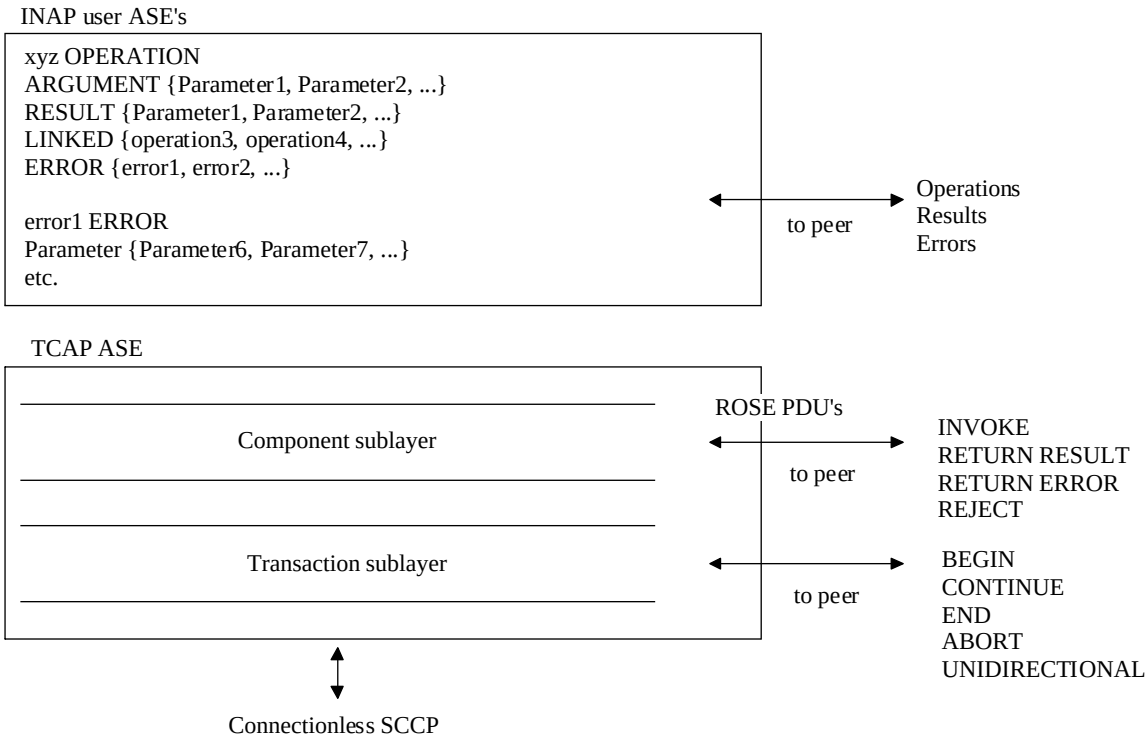


Figure 3-10: Operation description

4.1.5 Relationship between ITU-T Recommendation Q.1224 and the present document

The following is a complete list of information flows. These map one to one with operations except where indicated.

Note that for new operations introduced in the present document there are no equivalent information flows in ITU-T Recommendation Q.1224 [26] and they are therefore excluded from the following list.

Refer to clause 18 (Services Assumed from TCAP) to determine mapping of operations onto TCAP dialogue and component portions.

Table 3-2: Information/operation mapping

ITU-T Recommendation Q.1224 [26] subclause	Information Flow	Operation
SCF - SSF		
12.4.3.1	Activate Service Filtering	Same
12.4.3.2	Activate Trigger Data	ManageTriggerData
12.4.3.3	Activate Trigger Data Confirmation	Return Result from ManageTriggerData
12.4.3.4	Activity Test	Same
12.4.3.5	Activity Test Response	Return Result from ActivityTest
12.4.3.8	Apply Charging	Same
12.4.3.9	Apply Charging Report	Same
12.4.3.10	Assist Request Instructions	Same
12.4.3.12	Call Gap	Same
12.4.3.13	Call Information Report	Same
12.4.3.14	Call Information Request	Same
12.4.3.15	Cancel All Requests	Cancel (All Requests)
12.4.3.17	Collect Information	Same
12.4.3.19	Connect	Same
12.4.3.20	Connect to Resource	Same
12.4.3.21	Continue	Same, ContinuewithArgument
12.4.3.22	Create Call Segment Association	Same
12.4.3.23	Create Call Segment Association Result	Return Result from CreateCallSegmentAssociation
12.4.3.24	Deactivate Trigger Data	ManageTriggerData
12.4.3.25	Deactivate Trigger Data Confirmation	Return Result from ManageTriggerData
12.4.3.26	Disconnect Forward Connection	Same, DFCwithArgument
12.4.3.27	Disconnect Leg	Same
12.4.3.28	Entity Released	Same
12.4.3.29	Establish Temporary Connection	Same
12.4.3.30	Event Notification Charging	Same
	(continued)	

Table 3-2 (continued): Information/operation mapping

ITU-T Recommendation Q.1224 [26] subclause	Information Flow	Operation
12.4.3.31	Event Report Basic Call State Model (BCSM)	Same
12.4.3.34	Furnish Charging Information (FCI)	Same
12.4.3.36	Initial DP	Same
12.4.3.37	Initiate Call Attempt	Same
12.4.3.38	Merge Call Segments	Same
12.4.3.39	Move Call Segments	Same
12.4.3.40	Move Leg	Same
12.4.3.51	Release Call	Same
12.4.3.52	Report UTSI	Same
12.4.3.53	Request Notification Charging Event	Same
12.4.3.54	Request Report BCSM Event	Same
12.4.3.56	Request Report UTSI	Same
12.4.3.58	Reset Timer	Same
12.4.3.62	Send Charging Information	Same
12.4.3.64	Send Service To User Information (STUI)	Same
12.4.3.65	Service Filtering Response	Same
12.4.3.66	Split Leg	Same
12.4.3.76	Trigger Data Status Report	Return Result from ManageTriggerData
12.4.3.77	Trigger Data Status Request	ManageTriggerData
SCF - SRF		
12.5.2.1	AssistRequestInstructions from SRF	AssistRequestInstructions
12.5.2.2	Cancel Announcement	Cancel(invokeID)
12.5.2.3	Collected User Information	Return Result from PromptAndCollectUserInformation
12.5.2.4	Message Received	Return Result from PromptandReceiveMessage
12.5.2.5	Play Announcement (PA)	Same

(continued)

Table 3-2 (continued): Information/operation mapping

ITU-T Recommendation Q.1224 [26] subclause	Information Flow	Operation
12.5.2.6	Prompt And Collect (P&C) User Information	Same
12.5.2.7	Prompt and Receive Message	Same
12.5.2.8	Script Close	Same
12.5.2.9	Script Event	Same
12.5.2.10	Script Information	Same
12.5.2.11	Script Run	Same
12.5.2.12	Specialized Resource Report	Same
SCF - SCF		
12.6.2.1	Activity Test	Activity Test
12.6.2.2	Activity Test Result	Return Result from ActivityTest
12.6.2.3	Additional Information Result	Return Result from Provide User Information
12.6.2.4	Confirmed Notification Provided	Same
12.6.2.5	Confirmed Report Charging Information	Same
12.6.2.6	Establish Charging Record	Same
12.6.2.7	Handling Information Referral	
12.6.2.8	Handling Information Request	Same
12.6.2.9	Handling Information Result	Same
12.6.2.10	Network Capability Request	NetworkCapability
12.6.2.11	Network Capability Result	Return Result from NetworkCapability
12.6.2.12	Notification Provided	Same
12.6.2.13	Notification Provided Confirmation	Return Result from ConfirmedNotificationProvided
12.6.2.14	Provide User Information	Same
12.6.2.15	Report Charging Information	Same
12.6.2.16	Report Charging Information Confirmation	Return Result from ConfirmedReportChargingInformation Confirmation
12.6.2.17	Request Notification	Same
	(continued)	

Table 3-2 (continued): Information/operation mapping

ITU-T Recommendation Q.1224 [26] subclause	Information Flow	Operation
12.6.2.18	SCF Bind Request	SCF Bind
12.5.2.19	SCF Bind Result	Return Result from SCF Bind
12.6.2.20	SCF Unbind Request	SCF Unbind
SCF - CUSF		
12.7.2.1	Activation Received and Authorized	InitialAssociationDP
12.7.2.2	Activity Test	Same
12.7.2.3	Activity Test Response	Return Result from ActivityTest
12.7.2.4	Association Release Requested	InitialAssociation or Basic Call Unrelated State Model (BCUSM) Event
12.7.2.6	Initiate Association	Same
12.7.2.7	Request Report BCUSM Event	Same
12.7.2.8	Release Association	Same
12.4.3.58	Reset Timer	Same
SCF - SDF		
12.8.2.1	Add Entry	Same
12.8.2.2	Add Entry Referral	Return Error from AddEntry
12.2.8.3	Add Entry Result	Return Result from AddEntry
12.8.2.4	Authenticate	Bind
12.8.2.5	Authenticate Result	Return Result from Bind
12.8.2.6	End Authenticated Relationship	Unbind
12.8.2.7	Execute	Same
12.8.2.8	Execute Referral	Return Error from Execute
12.8.2.9	Execute Result	Return Result from Execute
12.8.2.10	Modify Entry	Same
12.8.2.11	Modify Entry Referral	Return Error from ModifyEntry
12.8.2.12	Modify Entry Result	Return Result from ModifyEntry
12.8.2.13	Remove Entry	Same
12.8.2.14	Remove Entry Referral	Return Error from RemoveEntry
	(continued)	

Table 3-2 (continued): Information/operation mapping

ITU-T Recommendation Q.1224 [26] subclause	Information Flow	Operation
12.8.2.15	Remove Entry Result	Return Result from RemoveEntry
12.8.2.16	Search	Same
12.8.2.17	Search Referral	Return Error from Search
12.8.2.18	Search Result	Return Result from Search
SDF - SDF		
12.9.2.1	Authenticate	dSABind, dSAShadowBind
12.9.2.2	Authenticate Result	Return Result from dSABind or dSAShadowBind
12.9.2.3	Chaining Request	chained {OPERATION}
12.9.2.4	Chaining Result	Return Result from OPERATION
12.9.2.5	Copy Request	Coordinate Shadow Update Request Shadow Update
12.9.2.6	Copy Result	Return Result from Coordinate Shadow Update or from Request Shadow Update
12.9.2.7	End Authenticated Relationship	in-DSAUnbind, in-DSAShadowUnbind
12.9.2.8	Update Copy	Update Shadow
12.9.2.9	Update Copy Result	Return Result from Update Shadow

4.1.6 Compatibility mechanisms used for INAP

4.1.6.1 Introduction

This subclause specifies the compatibility mechanisms that shall be used to ensure consistent future versions of INAP.

There are three categories of compatibility:

- Minor changes to INAP in future standardized versions:
  A minor change can be defined as a change of a functionality which is not essential for the requested IN service. Where it is a modification of an existing function, it is acceptable that the addressed function is executed in either the older or the modified variant. If the change is purely additional, it is acceptable that it is not executed at all and that the peer Application Entity (AE) need not know about the effects of the change. For minor changes, a new AC is not required.

- Major changes to INAP in future standardized versions:
  A major change can be defined as a change of a functionality which is essential for the requested IN service. Where it is a modification of an existing function, both application entities shall have a shared knowledge about the addressed functional variant. If the change is purely additional, the requested IN service will not be provided if one of the application entities does not support the additional functionality. For major changes, a new AC is required.

- Network-specific changes to INAP:
  These additions may be of either the major or minor type for a service. No new AC is expected to be defined for this type of change. At the time of definition, the additions would not be expected to be included in identical form in future versions of ITU-T Recommendations.

4.1.6.2 Definition of INAP compatibility mechanisms

4.1.6.2.1 Procedures for major additions to INAP

In order to support the introduction of major functional changes, the protocol allows a synchronization between the two applications with regard to which functionality is to be performed. This synchronization takes place before the new function is invoked in either AE, in order to avoid complicated fall-back procedures. The solution chosen to achieve such a synchronization is to use the AC negotiation procedures provided in ETS 300 287-1 [7].

4.1.6.2.2 Procedures for minor additions to INAP

The extension mechanism marker shall be used for future standardized minor additions to INAP. This mechanism implements extensions differently by including an "extensions marker" in the type definition. The extensions are expressed by optional fields that are placed after the marker. When an entity receives unrecognized parameters that occur after the marker, they are ignored (see ITU-T Recommendation X.68x-series []).

4.1.6.2.3 Procedures for inclusion of network specific additions to INAP

This mechanism is based on the ability to explicitly declare fields of any type via the macro facility in ASN.1 at the outermost level of a type definition. It works by defining an "ExtensionField" that is placed at the end of the type definition. This extension field is defined as a set of extensions, where an extension can contain any type. Each extension is associated with a value that defines whether the terminating node should ignore the field if unrecognized, or reject the message, in a manner similar to the comprehension required mechanism described in subclause 4.1.6.2.2. Refer to ITU-T Recommendation Q.1400 [30] for a definition of this mechanism.

4.2 SACF/MACF rules

4.2.1 Reflection of TCAP AC

TCAP AC negotiation rules require that the proposed AC, if acceptable, is reflected in the first backwards message.

If the AC is not acceptable, and the TC-User does not wish to continue the dialogue, it may provide an alternate AC to the initiator which can be used to start a new dialogue.

TCAP AC negotiation applies only to the SCF interfaces.

Refer to the ITU-T Recommendations Q.77 X-Series (*Transaction capabilities application part*) [18] for a more detailed description of the TCAP AC negotiation mechanism.

4.2.2 Sequential/parallel execution of operations

In some cases it may be necessary to distinguish whether operations should be performed sequentially or in parallel (synchronized). Operations which may be synchronized are:

- charging operations; may be synchronized with any other operation.

The method of indicating that operations are to be synchronized is to include them in the same message. Where one of the operations identified above is not to be executed until some other operation has progressed to some extent or finished, the sending PE (usually SCP) shall control this by sending the operations in two separate messages.

This method does not imply that all operations sent in the same message have to be executed simultaneously, but simply that where it could make sense to do so (in the situations identified above) the operations should be synchronized.

In case of inconsistency between the above-mentioned generic rules and the FE-specific rules, as specified in clause 3, the FE-specific rules take precedence over the generic rules.

5 Common IN Capability Set 2 (CS2) Types

5.1 Data types

```
-- The Definition of Common Data Types follows

IN-CS2-datatypes { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20)
modules(0) in-cs2-datatypes (0) version1(0)}

DEFINITIONS IMPLICIT TAGS: : =
BEGIN
IMPORTS
tc-Messages, classes FROM IN-CS2-object-identifiers
        { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20)
module(0) in-cs2-object-identifiers(17) version1(0) }
    InvokeIdType
FROM TCAPMessages tc-Messages
    EXTENSION,
        PARAMETERS-BOUND,
        SupportedExtensions { }
FROM IN-CS2-classes classes
;
AccountNumber: : = NumericString (SIZE (1..151))
AChBillingChargingCharacteristics {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE
(bound.&minAChBillingChargingLength..bound.&maxAChBillingChargingLength))
-- The AChBillingChargingCharacteristics parameter specifies the charging related information
-- to be provided by the SSF and the conditions on which this information has to be reported
-- back to the SCF with the ApplyChargingReport operation.
-- Examples of charging related information to be provided by the SSF may be: bulk counter
-- values, costs, tariff change and time of charge, time stamps, durations, etc.
-- Examples of conditions on which the charging related information are to be reported may be:
-- threshold value reached, timer expiration, tariff change, end of connection configuration,
etc
ActionIndicator : : = ENUMERATED {
activate       (1) ,
deactivate     (2) ,
retrieve       (3)
    }
-- indicates the action to be performed by the ManageTriggerData operation (activate, deactivate
-- or retrieve the status of a TDP.

ActionPerformed : : = ENUMERATED {
activated      (1) ,
deactivated (2) ,
alreadyActive  (3) ,
alreadyInactive (4) ,
isActive       (5) ,
isInactive     (6)
    }
-- indicates the result of the operation ManageTriggerData
-- activated: response of activate TDP
-- deactivated: response of deactivate TDP
-- alreadyActive: response of activate TDP
-- alreadyInactive: response of deactivate TDP
-- isActive: response of retrieve status of TDP
-- isInactive: response of retrieve status of TDP

ActivableServices: : = BIT STRING {
    callingLineIdentificationPresentation (1),
    callingLineIdentificationRestriction (2),
    connectedLineIdentificationPresentation (3),
    connectedLineIdentificationRestriction (4),
    callForwardingOnNoReply (5),
    callForwardingUnconditional (6),
    callForwardingOnBusy (7),
    callForwardingOnNotReachable (8),
    reverseCharging (9),
    adviceOfChargeOnStart (10),
    adviceOfChargeAtEnd (11),
    adviceOfChargeDuringCall (12),
    timeDependentRouting (13),
    callingPartingDependentRouting (14),
    outgoingCallBarring (15),
    incomingCallBarring (16)
    }
```

```
AdditionalCallingPartyNumber {PARAMETERS-BOUND: bound}: : = Digits {bound}
-- Indicates the Additional Calling Party Number. Refer to ITU-T Recommendation Q.763 [22] for
encoding.
AlertingPattern: : = OCTET STRING (SIZE(3))
-- Indicates a specific pattern that is used to alert a subscriber (e.g. distinctive ringing,
tones, etc.).
-- Only the trailing OCTET is used, the remaining OCTETS should be sent as NULL (zero)
-- The receiving side ignores the leading two OCTETS.
ApplicationTimer: : =INTEGER (0..2047)
-- Used by the SCF to set a timer in the SSF. The timer is in seconds.
AssistingSSPIPRoutingAddress {PARAMETERS-BOUND: bound}: : = Digits {bound}
-- Indicates the destination address of the SRF for the assist procedure.
BackwardGVNS {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
                  bound.&minBackwardGVNSLength..bound.&maxBackwardGVNSLength))
-- Indicats the GVNS Backward information. Refer to Q.735, §6 for encoding.

BackwardServiceInteractionInd: : = SEQUENCE {
    conferenceTreatmentIndicator   [1] OCTET STRING (SIZE(1)) OPTIONAL,
    -- acceptConferenceRequest 'xxxx xx01'B
    -- rejectConferenceRequest 'xxxx xx10'B
    -- network default is accept conference request,
    callCompletionTreatmentIndicator   [2] OCTET STRING (SIZE(1)) OPTIONAL
    -- acceptCallCompletionServiceRequest  'xxxx xx01'B,
    -- rejectCallCompletionServiceRequest  'xxxx xx10'B
    -- network default is accept call completion service request
    }


BCSMEvent {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    eventTypeBCSM   [0] EventTypeBCSM,
    monitorMode [1] MonitorMode,
    legID   [2] LegID   OPTIONAL,
    dpSpecificCriteria [30] DpSpecificCriteria {bound}    OPTIONAL
    }
-- Indicates the BCSM Event information for monitoring.
BCUSMEvent: : = SEQUENCE{
    eventType   [0] EventTypeBCUSM,
    monitorMode [1] MonitorMode
    }

BearerCapabilities: : = BIT STRING {
    speech (0),
    bc64kbits (1),
    bc2x64kbits (2),
    bc384kbits (3),
    bc1536kbits (4),
    bc1920kbits (5),
    multirate (6),
    restrictedDigitalInfo (7),
    bc3-1khzAudio (8),
    bc7khzAudio (9),
    video (10)
    }
BearerCapability {PARAMETERS-BOUND: bound}: : = CHOICE {
    bearerCap   [0] OCTET STRING (SIZE(2..bound.&maxBearerCapabilityLength)),
    tmr [1] OCTET STRING (SIZE(1))
    }
-- Indicates the type of bearer capability connection to the user. For bearerCapability, either
-- DSS1 (EN 300 403-1) or the ISUP User Service Information (ITU-T Recommendation Q.763 [22])
encoding can be used. Refer
-- to the ITU-T Recommendation Q.763 [22] Transmission Medium Requirement parameter for tmr
encoding.

BothwayThroughConnectionInd: : = ENUMERATED {
    bothwayPathRequired(0),
    bothwayPathNotRequired (1)
    }
-- The default is as specified in EN 301 070-1

CallConditions {PARAMETERS-BOUND: bound}: : = CHOICE {
    userAbandon [0] NULL,
    callFailure [1] CauseValue,
    noReply [2] INTEGER, -- time expressed in seconds
    callRelease [3] NULL,
    ss-invocation   [4] InvokableService,
    creditLimitReached [5] INTEGER,
    callDuration    [6] INTEGER,
    calledNumber    [7] NumberMatch {bound},
    answeredCall    [8] NULL
```

```
    }

CalledPartyBCDNumber {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
            bound.&minCalledPartyBCDNumberLength ..
            bound.&maxCalledPartyBCDNumberLength))
-- Indicates the Called Party Number, including service selection information. Refer to
GSM 04.08 [13]
-- for encoding. This data type carries only the "type of number", "numbering plan
-- identification" and "number digit" fields defined in [13]; it does not carry the "called
party
-- BCD number IEI" or "length of called party BCD number contents".

CalledPartyNumber {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE
                    (bound.&minCalledPartyNumberLength..
bound.&maxCalledPartyNumberLength))
-- Indicates the Called Party Number. Refer to ITU-T Recommendation Q.763 [22] for encoding.
CallIdentifier : : = INTEGER (1..2147483647)
CallingPartyBusinessGroupID {PARAMETERS-BOUND: bound}: : = OCTET STRING(SIZE(
            bound.&minCallingPartyBusinessGroupIDLength ..
            bound.&maxCallingPartyBusinessGroupIDLength))
-- Indicates the business group of the calling party. The value of this octet string is network
-- operator specific.
CallingPartyNumber {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
            bound.&minCallingPartyNumberLength..
            bound.&maxCallingPartyNumberLength))
-- Indicates the Calling Party Number. Refer to ITU-T Recommendation Q.763 [22] for encoding.
CallingPartySubaddress {PARAMETERS-BOUND: bound}: : = OCTET STRING(SIZE(
            bound.&minCallingPartySubAddressLength ..
            bound.&maxCallingPartySubAddressLength))

-- Indicates the Calling Party Subaddress. Refer to EN 300 403-1 for encoding.
CallingPartysCategory: : = OCTET STRING (SIZE(1))
-- Indicates the type of calling party (e.g. operator, payphone, ordinary subscriber). Refer to
ITU-T Recommendation Q.763 [22]
-- for encoding.

CallRecord {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    callDuration     [0] Duration,
    callingPartyNumber   [1] CallingPartyNumber {bound},
    calledPartyNumber    [2] CalledPartyNumber {bound}
    }
CallResult {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (bound.&minCallResultLength..
            bound.&maxCallResultLength))

-- This parameter provides the SCF with the charging related information previously requested
-- using the ApplyCharging operation. This shall include the partyToCharge parameter as
-- received in the related ApplyCharging operation to correlate the result to the request
-- The remaining content is network operator specific.
-- Examples of charging related information to be provided by the SSF may be: bulk counter
values,
-- costs, tariff change and time of change, time stamps, durations, etc.
-- Examples of conditions on which the charging related information are to be reported may be:
-- threshold value reached, timer expiration, tariff change, end of connection configuration,
etc.
CallSegmentID {PARAMETERS-BOUND: bound}: : = INTEGER (1..bound.&numOfCSs)
initialCallSegment INTEGER: : = 1
-- the initial call segment represents the call segment that was there when the CSA was created,
ie. the CS where
-- the trigger took place or the CS that was created by an InitateCallAttempt within a TC-BEGIN
message.

Carrier {PARAMETERS-BOUND: bound}: : = OCTET STRING(SIZE(
        bound.&minCarrierLength ..
        bound.&maxCarrierLength))
-- This parameter contains the Transit Network Selection. Refer to
ITU-T Recommendation Q.763 [22] for encoding.

Cause {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (minCauseLength..
                        bound.&maxCauseLength))
-- Indicates the cause for interface related information. Refer to the
ITU-T Recommendation Q.763 [22] Cause parameter for encoding.
-- For the use of cause and location values refer to ITU-T Recommendation Q.850 [24]
CauseValue: : = OCTET STRING (SIZE (1)) --type extracted from Cause parameter in
ITU-T Recommendation Q.763 [22].
CGEncountered: : = ENUMERATED {
    noCGencountered(0),
    manualCGencountered(1),
    scpOverload(2)
    }
```

```
-- Indicates the type of automatic call gapping encountered, if any.
ChargingEvent {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    eventTypeCharging  [0] EventTypeCharging {bound},
    monitorMode [1] MonitorMode,
    legID   [2] LegID   OPTIONAL,
    eventTypeTariff [3] EventTypeTariff OPTIONAL
    }
-- This parameter indicates the charging event type and corresponding
-- monitor mode and LegID
ChargingParameters  {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    unitsPerInterval   [0] INTEGER (0..bound.&maxUnitsPerInterval),
    timePerInterval [1] INTEGER (0..bound.&maxTimePerInterval),
    scalingFactor   [2] INTEGER (0..bound.&maxScalingFactor),
    initialUnitIncrement   [3] INTEGER (0..bound.&maxInitialUnitIncrement)   OPTIONAL,
    unitsPerDataInterval   [4] INTEGER (0..bound.&maxUnitsPerDataInterval)   OPTIONAL,
    segmentsPerDataInterval    [5] INTEGER (0..bound.&maxSegmentsPerDataInterval)    OPTIONAL,
    initialTimeInterval [6] INTEGER (0..bound.&maxInitialTimeInterval)        OPTIONAL
    }
CollectedDigits: : = SEQUENCE {
    minimumNbOfDigits  [0] INTEGER (1..127)   DEFAULT 1,
    maximumNbOfDigits  [1] INTEGER (1..127),
    endOfReplyDigit [2] OCTET STRING (SIZE (1..2)) OPTIONAL,
    cancelDigit [3] OCTET STRING (SIZE (1..2)) OPTIONAL,
    startDigit  [4] OCTET STRING (SIZE (1..2)) OPTIONAL,
    firstDigitTimeOut  [5] INTEGER (1..127)   OPTIONAL,
    interDigitTimeOut  [6] INTEGER (1..127)   OPTIONAL,
    errorTreatment  [7] ErrorTreatment DEFAULT reportErrorToScf,
    interruptableAnnInd [8] BOOLEAN DEFAULT TRUE,
    voiceInformation   [9] BOOLEAN DEFAULT FALSE,
    voiceBack   [10] BOOLEAN    DEFAULT FALSE
    }
-- The use of voiceBack is network operator specific.
-- The endOfReplyDigit, cancelDigit, and startDigit parameters have been designated as OCTET
STRING,
-- and are to be encoded as BCD, one digit per octet only, contained
-- in the four least significant bits of each OCTET. The usage is service dependent.
CollectedInfo: : = CHOICE {
    collectedDigits [0] CollectedDigits,
    iA5Information  [1] BOOLEAN
    }

ConnectedNumberTreatmentInd: : = ENUMERATED {
    noINImpact  (0),
    presentationRestricted (1),
    presentCalledINNumber  (2),
    presentCalledINNumberRestricted    (3)
    }
-- The default is as specified in EN 301 070-1

Constraints: : = SEQUENCE {
    maximumNumberOfDigits  [1] INTEGER (1..127),
    minimumNumberOfDigits  [2] INTEGER (1..127),
    typeOfRequestedInfo [3] InfoType DEFAULT numericString,
    numberOfAllowedRetries [4] INTEGER (0..127) DEFAULT 0
    }

ControlType: : = ENUMERATED {
    sCPOverloaded(0),
    manuallyInitiated(1),
    destinationOverload(2)
    -- other values FOR FURTHER STUDY
    }

CorrelationID {PARAMETERS-BOUND: bound}: : = Digits {bound}
-- used by SCF for correlation with a previous operation. Refer to clause 17 for a description
of the procedures
-- associated with this parameter.
CounterAndValue: : = SEQUENCE {
        counterID   [0] CounterID,
        counterValue    [1] Integer4
        }
CounterID: : = INTEGER (0..99)
-- Indicates the counters to be incremented.
-- The counterIDs can be addressed by using the last digits of the dialed number.
CountersValue: : = SEQUENCE SIZE(0..numOfCounters) OF CounterAndValue
Credit {PARAMETERS-BOUND: bound}: : = CHOICE {
        currency    CurrencyValue {bound},
        units   CreditUnit
            }
```

```
CreditUnit: : = INTEGER (0..maxCreditUnit)
CriticalityType: : = ENUMERATED {
        ignore(0),
        abort(1)
        }
CSAID {PARAMETERS-BOUND: bound}: : = INTEGER (1..bound.&numOfCSAs)
-- Indicates the SSF CSA identifier
CUApplicationInd : : = CHOICE{
    localValue  [0] INTEGER ,
    global  [1] OBJECT IDENTIFIER  ,
    ...
    }
-- Parameter 'cUApplicationInd' identifies the triggered application. Possible applications:
CCBS, CCNR, Q.SIG, etc
CurrencyID: : = PrintableString (SIZE (3)) -- ISO 639 [15] code
CurrencyValue {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    currency    CurrencyID,
    amount  INTEGER (0..bound.&maxAmount)
    }
CutAndPaste: : = INTEGER (0..22)
-- Indicates the number of digits to be deleted.

DateAndTime: : = OCTET STRING (SIZE(6))
-- Indicates, amongst others, the start time for activate service filtering. Coded as
YYMMDDHHMMSS
-- with each digit coded BCD.
-- The first octet contains YY and the remaining items are sequenced following.
-- For example, 1993 September 30th, 12: 15: 01 would be encoded as:
-- Bits                HGFE        DCBA
-- leading octet       3   9
--              9   0
--              0   3
--              2   1
--              5   1
--              1   0
DestinationRoutingAddress {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE(1) OF
CalledPartyNumber {bound}
-- Indicates the Called Party Number.
Digits {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE
(bound.&minDigitsLength..bound.&maxDigitsLength))
-- Indicates the address signalling digits. Refer to the ITU-T Recommendation Q.763 [22] Generic
Number and Generic Digits parameters
-- for encoding. The coding of the subfields 'NumberQualifier' in Generic Number and
'TypeOfDigits' in
-- Generic Digits are irrelevant to the INAP, the ASN.1 tags are sufficient to identify the
parameter.
-- The ISUP format does not allow to exclude these subfields, therefore the value is network
operator specific.
-- The following parameters should use Generic Number:
-- CorrelationID for AssistRequestInstructions, AssistingSSPIPRoutingAddress for
EstablishTemporaryConnection,
-- calledAddressValue for all occurrences,callingAddressValue for all occurrences.
-- The following parameters should use Generic Digits: prefix, all
-- other CorrelationID occurrences, dialledNumber filtering criteria, callingLineID filtering
criteria, lineID for ResourceID
-- type, digitResponse for ReceivedInformationArg, iNServiceControlLow / iNServiceControlHighfor
MidCallInfoType,
-- iNServiceControlCode for MidCallInfo.
DisplayInformation {PARAMETERS-BOUND: bound}: : = IA5String (SIZE
                        (bound.&minDisplayInformationLength..
            bound.&maxDisplayInformationLength))
-- Indicates the display information.
-- Delivery of DisplayInformation parameter to Private Networks cannot be guaranteed due to
signalling
-- interworking problems, solutions are currently under study
DpSpecificCriteria {PARAMETERS-BOUND: bound}: : = CHOICE {
        numberOfDigits [0] NumberOfDigits,
        applicationTimer   [1] ApplicationTimer,
        midCallControlInfo [2] MidCallControlInfo {bound}
        }

-- The SCF may specify the number of digits to be collected by the SSF for the
EventReportBCSMEvent event.
-- When all digits are collected, the SSF reports the event to the SCF.
-- The SCF may set a timer in the SSF for the No Answer event. If the user does not answer the
call
-- within the allotted time, the SSF reports the event to the SCF
Duration: : = INTEGER (-2..86400)
-- Values are seconds
```

```
ElementaryMessageID: : = Integer4

Entry: : =CHOICE {
    agreements  [0] OBJECT IDENTIFIER,
    networkSpecific [1] Integer4
    }

ErrorTreatment: : = ENUMERATED {
        reportErrorToScf(0),
        help(1),
        repeatPrompt(2)
        }
-- reportErrorToScf means returning the "ImproperCallerResponse" error in the event of an error
-- condition during collection of user info.
EventSpecificInformationBCSM {PARAMETERS-BOUND: bound}: : = CHOICE {
    collectedInfoSpecificInfo  [0] SEQUENCE {
            calledPartynumber  [0] CalledPartyNumber {bound},
            ...
            },
    analysedInfoSpecificInfo   [1] SEQUENCE {
            calledPartynumber  [0] CalledPartyNumber {bound},
            ...
            },
    routeSelectFailureSpecificInfo [2] SEQUENCE {
            failureCause    [0] Cause {bound}  OPTIONAL,
            ...
            },
    oCalledPartyBusySpecificInfo   [3] SEQUENCE {
            busyCause   [0] Cause {bound}  OPTIONAL,
            ...
            },
    oNoAnswerSpecificInfo  [4] SEQUENCE {
            -- no specific info defined --
            ...
            },
    oAnswerSpecificInfo[5] SEQUENCE {
                        backwardGVNS  [0] BackwardGVNS {bound}
                          OPTIONAL,
            ...
            },
    oMidCallSpecificInfo   [6] SEQUENCE {
            connectTime [0] Integer4    OPTIONAL,
            oMidCallInfo    [1] MidCallInfo {bound}                                 OPTIONAL,
            ...
            },
    oDisconnectSpecificInfo    [7] SEQUENCE {
            releaseCause    [0] Cause {bound}                               OPTIONAL,
            connectTime [1] Integer4    OPTIONAL,
             ...
            },
    tBusySpecificInfo  [8] SEQUENCE {
            busyCause   [0] Cause {bound}                                   OPTIONAL,
             ...
            },
    tNoAnswerSpecificInfo  [9] SEQUENCE {
            -- no specific info defined --
             ...
            },
    tAnswerSpecificInfo[10] SEQUENCE {
            -- no specific info defined --
            ...
            },
    tMidCallSpecificInfo   [11] SEQUENCE {
            connectTime [0] Integer4    OPTIONAL,
            tMidCallInfo    [1] MidCallInfo {bound}                                 OPTIONAL,
             ...
            },
    tDisconnectSpecificInfo    [12] SEQUENCE {
            releaseCause    [0] Cause {bound}                               OPTIONAL,
            connectTime [1] Integer4    OPTIONAL,
             ...
            },
    oTermSeizedSpecificInfo    [13] SEQUENCE {
            -- no specific info defined
             ...
            },
```

```
    oSuspended  [14] SEQUENCE {
            -- no specific info defined
             ...
            },
    tSuspended  [15] SEQUENCE {
            -- no specific info defined
             ...
            },
    origAttemptAuthorized  [16] SEQUENCE {
            -- no specific info defined
             ...
            },
    oReAnswer   [17] SEQUENCE {
            -- no specific info defined
             ...
            },
    tReAnswer   [18] SEQUENCE {
            -- no specific info defined
             ...
            },
    facilitySelectedAndAvailable   [19] SEQUENCE {
            -- no specific info defined
             ...
            },
    callAccepted    [20] SEQUENCE {
            -- no specific info defined
             ...
            }},
    oAbandon    [21] SEQUENCE {
            abandonCause    [0] Cause {bound}  OPTIONAL,
            ...
            },
    tAbandon    [22] SEQUENCE {
            abandonCause    [0] Cause {bound}  OPTIONAL,
            ...
            }
    oNotReachable   [50] SEQUENCE {
            notReachableCause   [0] Cause {bound}  OPTIONAL,
            ...
            },
    tNotReachable   [51] SEQUENCE {
            notReachableCause   [0] Cause {bound}  OPTIONAL,
            ...
            }
    }

-- Indicates the call related information specific to the event.
-- The connectTime indicates the duration between the received answer indication from the called
party side
-- and the release of the connection for ODisconnect, OException, TDisconnect, or TException or
between
-- the received answer indication from the called party side and the time of detection of the
required
-- mid call event.
-- The unit for the connectTime is 100 milliseconds

EventSpecificInformationBCUSM {PARAMETERS-BOUND: bound}: : = CHOICE{
    associationReleaseRequestedSpecificInfo
            [1] SEQUENCE{
            associationReleaseInfo [0] OCTET STRING
            (SIZE(bound.&minAssReleaseInfoLength
                 ..bound.&maxAssReleaseInfoLength)
                            OPTIONAL,
            releaseCause    [1] Cause {bound}
                            OPTIONAL,
        ...
        }
    }
EventSpecificInformationCharging {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE
                            (bound.&minEventSpecificInformationChargingLength..
            bound.&maxEventSpecificInformationChargingLength))
-- defined by network operator.
-- Indicates the charging related information specific to the event.
-- An example data type definition for this parameter is given below:
--      chargePulses       [0] Integer4,
--      chargeMessages     [1] OCTET STRING (SIZE (min..max))
EventTypeBCSM: : = ENUMERATED {
    origAttemptAuthorized(1),
    collectedInfo(2),
```

```
    analysedInformation(3),
    routeSelectFailure(4),
    oCalledPartyBusy(5),
    oNoAnswer(6),
    oAnswer(7),
    oMidCall(8),
    oDisconnect(9),
    oAbandon(10),
    termAttemptAuthorized(12),
    tBusy(13),
    tNoAnswer(14),
    tAnswer(15),
    tMidCall(16),
    tDisconnect(17),
    tAbandon(18),
    oTermSeized(19),
    oSuspended(20),
    tSuspended(21),
    origAttempt(22),
    termAttempt(23),
    oReAnswer(24),
    tReAnswer(25),
    facilitySelectedAndAvailable(26),
    callAccepted(27),
    oNotReachable(50),
    tNotReachable(51)

    }

-- Indicates the BCSM detection point event. Refer to Q.1224 for additional information on the
events.
-- Values origAttempt and termAttempt can only be used for TDPs

EventTypeBCUSM: : = ENUMERATED{
        componentReceived(127),
        associationReleaseRequested(126),
        activationReceivedAndAuthorized(125)
        }

EventTypeCharging {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
                            bound.&minEventTypeChargingLength..
            bound.&maxEventTypeChargingLength))
-- This parameter indicates the charging event type. Its content is network operator specific.
--
-- An example data type definition for this parameter is given below:
-- EventTypeCharging: : = ENUMERATED {
--              chargePulses (0),
--              chargeMessages (1)
--              }
EventTypeTariff: : = ENUMERATED {
    chargingTariffInformation  (0),
    chargingPriceInformation   (1),
    chargingAcknowledgementInformation (2),
    chargingAcknowledgeTimerExpired    (3)
    }
ExtensionField {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    type    EXTENSION.&id ({SupportedExtensions {bound}}),
            -- shall identify the value of an EXTENSION type
    criticality CriticalityType DEFAULT ignore,
    value   [1] EXTENSION.&ExtensionType
               ({SupportedExtensions {bound}}{@type})
    }
--This parameter indicates an extension of an argument data type. Its content is network
operator specific
FCIBillingChargingCharacteristics {PARAMETERS-BOUND: bound}: : = CHOICE {
    fCIBCCcs1   OCTET STRING (SIZE (bound.&minFCIBCcs1Length..
                bound.&maxFCIBCcs1Length)),
    fCIBCCsequencecs2  [1] SEQUENCE {
        fCIBCC  [0] OCTET STRING (SIZE (bound.&minFCIBCCcs2Length ..
               bound.&maxFCIBCCcs2Length)),
        tariff  [1] CHOICE {
            crgt    [0] ChargingTariffInformation,
            crgp    [1] ChargingPriceInformation
            } OPTIONAL
        }
    }

-- This parameter indicates the billing and/or charging characteristics. Its content is network
operator specific.
```

```
-- An example datatype definition for this parameter is given below:
-- FCIBillingChargingCharacteristics: : = CHOICE {
--  completeChargingrecord     [0] OCTET STRING (SIZE (min..max)),
--  correlationID              [1] CorrelationID,
--  scenario2Dot3              [2] SEQUENCE {
--                          chargeParty    [0] LegID   OPTIONAL,
--                          chargeLevel    [1] OCTET STRING (SIZE (min..max))
--                                                 OPTIONAL,
--                          chargeItems    [2] SET OF Attribute   OPTIONAL
--                          }
--  }
FilteredCallTreatment {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    sFBillingChargingCharacteristics  [0] SFBillingChargingCharacteristics {bound},
    informationToSend  [1] InformationToSend {bound}  OPTIONAL,
    maximumNumberOfCounters    [2] MaximumNumberOfCounters    OPTIONAL,
    releaseCause    [3] Cause {bound}  OPTIONAL,
    sFTariffMessage [4] CHOICE {
        crgt       [0] ChargingTariffInformation}  OPTIONAL
            }
    }
-- If releaseCause is not present, the default value is the same as the ISUP cause value
decimal 31.
-- If informationToSend is present, the call will be released after the end of the announcement
-- with the indicated or default releaseCause.
-- If maximumNumberOfCounters is not present, ServiceFilteringResponse will be sent with
-- CountersValue: : = SEQUENCE SIZE (0) OF CountersAndValue
FilteringCharacteristics: : = CHOICE {
    interval    [0] INTEGER (-1..32000),
    numberOfCalls  [1] Integer4
    }
-- Indicates the severity of the filtering and the point in time when the
ServiceFilteringResponse is to be sent.
-- If = interval, every interval of time the next call leads to an InitialDP and a
ServiceFilteringResponse is sent to
-- the SCF. The interval is specified in seconds.
-- If = NumberOfCalls, every N calls the Nth call leads to an InitialDP and a
ServiceFilteringResponse
-- is sent to the SCF.
-- If ActivateServiceFiltering implies several counters - filtering on several dialled numbers -
,
-- the numberOfCalls would include calls to all the dialled numbers.
FilteringCriteria {PARAMETERS-BOUND: bound}: : = CHOICE {
    serviceKey  [2] ServiceKey,
    addressAndService  [30] SEQUENCE {
            calledAddressValue [0] Digits {bound},
            serviceKey  [1] ServiceKey,
            callingAddressValue[2] Digits {bound}
                       OPTIONAL,
            locationNumber  [3] LocationNumber {bound}
    OPTIONAL
            }
    }
-- In case calledAddressValue is specified, the numbers to be filtered are from
calledAddressValue
-- up to and including calledAddressValue + maximumNumberOfCounters-1.
-- The last two digits of calledAddressvalue can not exceed 100-maximumNumberOfCounters.
FilteringTimeOut: : = CHOICE {
    duration    [0] Duration,
    stopTime    [1] DateAndTime
    }
-- Indicates the maximum duration of the filtering. When the timer expires, a
ServiceFilteringResponse
-- is sent to the SCF.
ForwardCallIndicators: : = OCTET STRING (SIZE(2))
-- Indicates the Forward Call Indicators. Refer to ITU-T Recommendation Q.763 [22] for encoding
ForwardGVNS {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
            bound.&minForwardGVNSLength..
            bound.&maxForwardGVNSLength))
-- Indicats the GVNS Forward information. Refer to Q.735, §6 for encoding.
ForwardServiceInteractionInd: : = SEQUENCE {
    conferenceTreatmentIndicator   [1] OCTET STRING (SIZE(1)) OPTIONAL,
    -- acceptConferenceRequest 'xxxx xx01',B
    -- rejectConferenceRequest 'xxxx xx10'B
    -- network default is accept conference request
    callDiversionTreatmentIndicator    [2] OCTET STRING (SIZE(1)) OPTIONAL,
    -- callDiversionAllowed    'xxxx xx01'B
    -- callDiversionNotAllowed 'xxxx xx10'B
    -- network default is Call Diversion allowed
    callOfferingTreatmentIndicator [3] OCTET STRING (SIZE(1)) OPTIONAL,
```

```
    -- callOfferingNotAllowed  'xxxx xx01'B,
    -- callOfferingAllowed     'xxxx xx10'B
    -- network default is Call Offering not allowed
    callingPartyRestrictionIndicator   [4] OCTET STRING (SIZE(1)) OPTIONAL
    -- noINImpact              'xxxx xx01'B,
    -- presentationRestricted      'xxxx xx10'B
    -- network default is noINImpact
    }

GapCriteria {PARAMETERS-BOUND: bound}: : = CHOICE {
    calledAddressValue [0] Digits {bound},
    gapOnService    [2] GapOnService,
    gapAllInTraffic [3] NULL,
    calledAddressAndService    [29] SEQUENCE {
            calledAddressValue [0] Digits {bound},
            serviceKey  [1] ServiceKey
            },
    callingAddressAndService   [30] SEQUENCE {
            callingAddressValue[0] Digits {bound},
            serviceKey  [1] ServiceKey,
            locationNumber  [2] LocationNumber {bound}                          OPTIONAL
            }
    }
-- Both calledAddressValue and callingAddressValue can be
-- incomplete numbers, in the sense that a limited amount of digits can be given.
--
-- For the handling of numbers starting with the same digit string refer to the detailed
procedure
-- of the CallGap operation
GapOnService: : = SEQUENCE {
    serviceKey  [0] ServiceKey
    }
GapIndicators: : = SEQUENCE {
    duration    [0] Duration,
    gapInterval [1] Interval
    }
-- Indicates the gapping characteristics. No gapping when gapInterval equals 0, and gap all
calls when
-- gapInterval equals -1.
GapTreatment {PARAMETERS-BOUND: bound}: : = CHOICE {
    informationToSend  [0] InformationToSend {bound},
    releaseCause    [1] Cause {bound},
    both    [2] SEQUENCE {
             informationToSend [0] InformationToSend {bound},
             releaseCause   [1] Cause {bound}
             }
    }
-- The default value for Cause is the same as in ISUP.
GenericName {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
               bound.&minGenericNameLength..
               bound.&maxGenericNameLength))
-- Refer to EN 300 403-1 Display Information for encoding
GenericNumber {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
               bound.&minGenericNumberLength..
               bound.&maxGenericNumberLength))
-- Refer to ITU-T Recommendation Q.763 [22] Generic Number for encoding.
GenericNumbers {PARAMETERS-BOUND: bound}: : = SET SIZE(1..bound.&numOfGenericNumbers) OF
GenericNumber {bound}
HighLayerCompatibilities: : = BIT STRING {
    telephony (0),
    facsimileGroup2-3 (1),
    facsimileGroup4classeI (2),
    teletexMixedMode (3),
    teletexProcessableMode (4),
    teletexBasicMode (5),
    syntaxBasedVideotex (6),
    internationalVideotex (7),
    telexService (8),
    messageHandlingSystem (9),
    osiApplication (10),
    audioVisual (11)
    }
HighLayerCompatibility: : = OCTET STRING (SIZE (highLayerCompatibilityLength))
-- Indicates the teleservice. For encoding, DSS1 (EN 300 403-1) is used.
InbandInfo {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    messageID   [0] MessageID {bound},
    numberOfRepetitions[1] INTEGER (1..127)   OPTIONAL,
    duration    [2] INTEGER (0..32767) OPTIONAL,
    interval    [3] INTEGER (0.. 32767)    OPTIONAL
```

```
    }
-- Interval is the time in seconds between each repeated announcement. Duration is the total
-- amount of time in seconds, including repetitions and intervals.
-- The end of announcement is either the end of duration or numberOfRepetitions, whatever comes
first.
-- duration with value 0 indicates infinite duration
InformationToRecord {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    messageID   [0] ElementaryMessageID    OPTIONAL,
    messageDeletionTimeOut [1] INTEGER (1..3600)  OPTIONAL,
    timeToRecord    [3] INTEGER (0..bound.&maxRecordingTime)  OPTIONAL,
    controlDigits   [4] SEQUENCE {
        endOfRecordingDigit[0] OCTET STRING (SIZE(1..2))  OPTIONAL,
        cancelDigit [1] OCTET STRING (SIZE(1..2))  OPTIONAL,
        replayDigit [2] OCTET STRING (SIZE(1..2))  OPTIONAL,
        restartRecordingDigit  [3] OCTET STRING (SIZE(1..2))  OPTIONAL,
        restartAllowed  [4] BOOLEAN     DEFAULT FALSE,
        replayAllowed   [5] BOOLEAN     DEFAULT FALSE
        }
    }
InformationToSend {PARAMETERS-BOUND: bound}: : = CHOICE {
    inbandInfo  [0] InbandInfo {bound},
    tone    [1] Tone,
    displayInformation [2] DisplayInformation {bound}
    }
InfoToSend {PARAMETERS-BOUND: bound}: : = CHOICE {
    messageID   [0] MessageID {bound},
    toneId  [1] ToneId,
    displayInformation [2] DisplayInformation {bound}
    }
InfoType: : = ENUMERATED {
    numericString (0),
    characterString (1),
    iA5String (2)
    }
INServiceCompatibilityIndication {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE
(1..bound.&numOfInServiceCompatibilityIndLength) OF Entry
INServiceCompatibilityResponse : : = Entry
Integer4: : = INTEGER(0..2147483647)
Interval: : = INTEGER (-1..60000)
-- Units are milliseconds.
InvokableService: : = ENUMERATED {
    callingLineIdentificationRestriction (1),
    connectedLineIdentificationRestriction (2),
    callWaiting (3),
    callHold (4),
    reverseCharging (5),
    explicitCallTransfer (6),
    callCompletionOnBusySubscriber (7)
    }
InvokeID: : = InvokeIdType
-- Operation invoke identifier.
IPAvailable {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
            bound.&minIPAvailableLength..bound.&maxIPAvailableLength))
-- defined by network operator.
-- Indicates that the resource is available.
IPRoutingAddress {PARAMETERS-BOUND: bound}: : = CalledPartyNumber {bound}
-- Indicates the routing address for the IP.
IPSSPCapabilities {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
            bound.&minIPSSPCapabilitiesLength..
            bound.&maxIPSSPCapabilitiesLength))
-- defined by network operator.
-- Indicates the SRF resources available at the SSP.
ISDNAccessRelatedInformation {PARAMETERS-BOUND: bound}: : = OCTET STRING(SIZE(
            bound.&minISDNAccessRelatedInformationLength ..
            bound.&maxISDNAccessRelatedInformationLength))

-- Indicates the destination user network interface related information. Refer to the
ITU-T Recommendation Q.763 [22] Access
-- Transport parameter for encoding.
Language: : = PrintableString (SIZE (3)) -- ISO 639 [15] codes only;

LegID: : = CHOICE {
    sendingSideID   [0] LegType,
    receivingSideID [1] LegType
    }
```

```
-- Indicates a reference to a specific party in a call. OPTIONAL denotes network operator
specific use
-- with a choice of unilateral ID assignment or bilateral ID assignment.
-- OPTIONAL for LegID also denotes the following:
--when only one party exists in the call, this parameter is not needed (as no ambiguity exists);
--when more than one party exists in the call, one of the following alternatives applies:
--   1. LegID is present and indicates which party is concerned.
--   2. LegID is not present and a default value is assumed (e.g. calling party in the case of
the
--    ApplyCharging operation).
-- Choice between these two alternatives is kept a network operator option.
LegType: : = OCTET STRING (SIZE(1))
leg1 LegType: : = '01'H
leg2 LegType: : = '02'H
LocationNumber {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
           bound.&minLocationNumberLength..
           bound.&maxLocationNumberLength))
-- Indicates the Location Number for the calling party. Refer to ITU-T Recommendation Q.763 [22]
(White book) for encoding.
MailBoxID {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
               bound.&minMailBoxIDLength..bound.&maxMailBoxIDLength))
MaximumNumberOfCounters: : = INTEGER (1..numOfCounters)
Media: : = ENUMERATED {
    voiceMail (0),
    faxGroup3 (1),
    faxGroup4 (2)
    }

MessageID {PARAMETERS-BOUND: bound}: : = CHOICE {
    elementaryMessageID[0] Integer4,
    text    [1] SEQUENCE {
             messageContent[0] IA5String (SIZE (
    bound.&minMessageContentLength..
            bound.&maxMessageContentLength)),
             attributes[1] OCTET STRING (SIZE (              bound.&minAttributesLength..
                bound.&maxAttributesLength))
                              OPTIONAL
             },
    elementaryMessageIDs   [29] SEQUENCE SIZE (1.. bound.&numOfMessageIDs) OF Integer4,
    variableMessage[30] SEQUENCE {
             elementaryMessageID   [0] Integer4,
             variableParts  [1] SEQUENCE SIZE (1..5)               OF VariablePart {bound}
             }
    }
-- OPTIONAL denotes network operator specific use.
MidCallControlInfo {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE (
        bound.&minMidCallControlInfoNum ..
        bound.&maxMidCallControlInfoNum) OF SEQUENCE {
            midCallInfoType    [0] MidCallInfoType {bound},
            midCallReportType      [1] ENUMERATED {
                                       inMonitoringState(0),
                                       inAnyState(1)
                                   } DEFAULT inMonitoringState
            }
MidCallInfo {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    iNServiceControlCode   [0] Digits {bound}
    }
MidCallInfoType {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    iNServiceControlCodeLow    [0] Digits {bound},
    iNServiceControlCodeHigh   [1] Digits {bound} OPTIONAL
    }
MiscCallInfo: : = SEQUENCE {
    messageType[0] ENUMERATED {
             request(0),
             notification(1)
             }
    }
-- Indicates detection point related information.
MonitorMode : : = ENUMERATED {
    interrupted(0),
    notifyAndContinue(1),
    transparent(2)
    }
-- Indicates the event is relayed and/or processed by the SSP.
-- If this parameter is used in the context of charging events, the following definitions apply
for the
-- handling of charging events:
-- Interrupted means that the SSF notifies the SCF of the charging event using
-- EventNotificationCharging, does not process the event but discard it.
```

```
-- NotifyAndContinue means that SSF notifies the SCF of the charging event using
-- EventNotificationCharging, and continues processing the event or signal without waiting for
SCF instructions.
-- Transparent means that the SSF does not notify the SCF of the event. This value is used to
end the monitoring
-- of a previously requested charging event. Previously requested charging events are monitored
-- until ended by a transparent monitor mode, or until the end of the connection configuration.
-- For the use of this parameter in the context of BCSM events refer to clause 17.
Notification: : = ENUMERATED {
    userAbandon (0),
    callFailure (1),
    noReply (2),
    callRelease (3),
    ssInvocation (4),
    creditLimitReached (5),
    callDuration (6),
    calledNumber (7),
    answeredCall (8)
    }
NotificationInformation {PARAMETERS-BOUND: bound}: : = CHOICE {
    userAbandonSpecificInfo    [0] SEQUENCE {...},
    callFailureSpecificInfo    [1] SEQUENCE {
            failureCause    [0] Cause {bound}      OPTIONAL,
            ...},
    noReplySpecificInfo[2] SEQUENCE {...},
    callReleaseSpecificInfo    [3] SEQUENCE {
            releaseCause    [0] Cause {bound}      OPTIONAL,
            timeStamp   [1] DateAndTime    OPTIONAL,
            ...},
    ssInvocationSpecificInfo   [4] SEQUENCE {
            invokedService  [0] InvokableService,
            ...},
    creditLimitReachedSpecificInfo [5] SEQUENCE {
            timeStamp   [0] DateAndTime    OPTIONAL,
            ...},
    callDurationSpecificInfo   [6] SEQUENCE {
            timeStamp   [0] DateAndTime    OPTIONAL,
            ...},
    calledNumberSpecificInfo   [7] SEQUENCE {
            calledNumber    [0] CalledPartyNumber {bound}  OPTIONAL,
            ...},
    answeredCallSpecificInfo   [8] SEQUENCE {
            timeStamp   [0] DateAndTime    OPTIONAL,
            ...}
    }
NumberOfDigits: : = INTEGER (1..255)
-- Indicates the number of digits to be collected

OCSIApplicacable: : = NULL
-- Indicates that the OCSI, if present, shall be applied on the outgoing call leg created with a
Connect operation
OperationCode: : = CHOICE {
    globalCode  OBJECT IDENTIFIER,
    local   INTEGER
    }
-- contains the operation value, or error value (object identifier), or problem value of the
FACILITY IE,
-- and the argument, the result, or the reject part of the same FACILITY IE that are received
with DSS1
-- message from the user. (see Q.932 8.2.2 for encoring)

OriginalCalledPartyID {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE(
            bound.&minOriginalCalledPartyIDLength..
            bound.&maxOriginalCalledPartyIDLength))
-- Indicates the original called number. Refer to the ITU-T Recommendation Q.763 [22] Original
Called Number for encoding.
ProfileIdentifier {PARAMETERS-BOUND: bound} : : = CHOICE {
access  [0] CalledPartyNumber {bound},
group       [1] FacilityGroup
 }
-- Please note that 'CalledPartyNumber' is used to address a subscriber access line.
--The data type was reused from the existing types to avoid the definition of a new one.

Reason {PARAMETERS-BOUND: bound}: : = OCTET STRING(SIZE(
       bound.&minReasonLength..bound.&maxReasonLength))
ReceivedInformation {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE (
            bound.&minReceivedInformationLength..
            bound.&maxReceivedInformationLength) OF IA5String
```

```
ReceivedStatus: : =ENUMERATED {
    messageComplete (0),
    messageInterrupted (1),
    messageTimeOut (2)
    }
RecordedMessageID: : = Integer4
RedirectingPartyID {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
           bound.&minRedirectingPartyIDLength..
           bound.&maxRedirectingPartyIDLength))
-- Indicates redirecting number. Refer to the ITU-T Recommendation Q.763 [22] Redirecting number
for encoding.
RedirectionInformation: : = OCTET STRING (SIZE(2))
-- Indicates redirection information. Refer to the ITU-T Recommendation Q.763 [22] Redirection
Information for encoding.
RegistratorIdentifier {PARAMETERS-BOUND: bound} : : = OCTET STRING(SIZE(
           bound.&min RegistratorIdentifierLength ..
           bound.&max RegistratorIdentifierLength))


RequestedInformationList {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE (1..numOfInfoItems) OF
RequestedInformation {bound}
RequestedInformationTypeList: : = SEQUENCE SIZE (1..numOfInfoItems) OF RequestedInformationType
RequestedInformation {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    requestedInformationType   [0] RequestedInformationType,
    requestedInformationValue  [1] RequestedInformationValue {bound}
    }
RequestedInformationType: : = ENUMERATED {
    callAttemptElapsedTime(0),
    callStopTime(1),
    callConnectedElapsedTime(2),
    calledAddress(3),
    releaseCause(30)
    }
RequestedInformationValue {PARAMETERS-BOUND: bound}: : = CHOICE {
    callAttemptElapsedTimeValue    [0] INTEGER (0..255),
    callStopTimeValue  [1] DateAndTime,
    callConnectedElapsedTimeValue  [2] Integer4,
    calledAddressValue [3] Digits {bound},

    releaseCauseValue  [30] Cause {bound}
    }

-- The callAttemptElapsedTimeValue is specified in seconds. The unit for the
-- callConnectedElapsedTimeValue is 100 milliseconds
RequestedNotifications {PARAMETERS-BOUND: bound}: : = SET OF CallConditions {bound}
RequestedType: : = INTEGER (0 .. 127)
RequestedUTSI {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    uSIServiceIndicator[0] USIServiceIndicator {bound},
    uSImonitorMode  [1] USIMonitorMode,
    }

RequestedUTSIList {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE (bound.&minRequestedUTSINum..
           bound.&maxRequestedUTSINum) OF RequestedUTSI {bound}


ResponseCondition: : = ENUMERATED {
    intermediateResponse(0),
    lastResponse(1)
-- additional values are for further study
    }
-- ResponseCondition is used to identify the reason why ServiceFilteringResponse operation is
sent.
-- intermediateresponse identifies that service filtering is running and the interval time is
expired and
-- a call is received, or that service filtering is running and the threshold value is reached.
-- lastResponse identifies that the duration time is expired and service filtering has been
finished or
-- that the stop time is met and service filtering has been finished.
RouteList {PARAMETERS-BOUND: bound}: : = SEQUENCE SIZE(1..3) OF OCTET STRING (SIZE
(bound.&minRouteListLength..bound.&maxRouteListLength))
-- Indicates a list of trunk groups or a route index. See Q.1224 for additional information on
this item.
RoutingAddress {PARAMETERS-BOUND: bound}: : = CHOICE {
    routingProhibited  [0] NULL,
    destinationRoutingAddress  [1] DestinationRoutingAddress {bound}
    }
ScfAddress {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE
(bound.&minScfAddressLength..bound.&maxScfAddressLength))
-- ISDN address
```

```
ScfID {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE
    (bound.&minScfIDLength..bound.&maxScfIDLength))
-- defined by network operator.
-- Indicates the SCF identity.
-- Used to derive the INAP address of the SCF to establish a connection between a requesting FE
-- and the specified SCF.
-- When ScfID is used in an operation which may cross an internetwork boundary, its encoding
must
-- be understood in both networks; this requires bilateral agreement on the encoding.
-- Refer to 3.5/ETS 300 009-1 "calling party address" parameter for encoding. It indicates the
SCCP address of the SCF.
-- Other encoding schemes are also possible as an operator specific option.
SCIBillingChargingCharacteristics {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
            bound.&minSCIBillingChargingLength..
            bound.&maxSCIBillingChargingLength))
-- This parameter indicates the billing and/or charging characteristics. Its content is network
operator specific.
-- An example datatype definition for this parameter is given below:
-- SCIBillingChargingCharacteristics : : = CHOICE {
--  chargeLevel        [0] OCTET STRING (SIZE (min..max),
--  chargePulses           [1] Integer4,
--  chargeMessages  [2] OCTET STRING (SIZE (min..max)
--  }
ServiceInteractionIndicators {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
                           bound.&minServiceInteractionIndicatorsLength..
            bound.&maxServiceInteractionIndicatorsLength))
-- Indicators which are exchanged between SSP and SCP to resolve interactions between IN based
services
-- and network based services, respectively between different IN based services.
-- The contents are network specific.
-- Note this parameter is kept in CS2 for backward compatibility to CS1R, for CS2 see new
-- parameter ServiceInteractionIndicatorsTwo
ServiceInteractionIndicatorsTwo: : =  SEQUENCE {
    forwardServiceInteractionInd   [0] ForwardServiceInteractionInd   OPTIONAL,
    -- applicable to operations IDP, CON, ICA.
    backwardServiceInteractionInd  [1] BackwardServiceInteractionInd  OPTIONAL,
    -- applicable to operations IDP, CON, CTR, ETC.
    bothwayThroughConnectionInd    [2] BothwayThroughConnectionInd    OPTIONAL,
    -- applicable to operations CTR, ETC.
    suspendTimer    [3] SuspendTimer   OPTIONAL,
    -- applicable to operations CON, ICA.
    connectedNumberTreatmentInd    [4] ConnectedNumberTreatmentInd    OPTIONAL,
    -- applicable to operations CON, CTR, ETC.
    suppressCallDiversionNotification  [5] BOOLEAN    OPTIONAL,
    -- applicable to CON, ICA
    suppressCallTransferNotification   [6] BOOLEAN    OPTIONAL,
    -- applicable to CON, ICA
    allowCdINNoPresentationInd [7] BOOLEAN OPTIONAL,
    -- applicable to CON, ICA
    -- indicates whether the Number Presentation not allowed indicator of the ISUP
    -- "called IN number" shall be set to presentation allowed (TRUE) or presentation not
allowed (FALSE)
    userDialogueDurationInd    [8] BOOLEAN DEFAULT TRUE,
    -- applicable when interaction with the user is required, if the interaction
    -- TRUE means the user interaction may last longer than 90 seconds. Otherwise the
    -- indicator should be set to FALSE.
    -- used for delaying ISUP T9 timer.
    ...
    }
-- Indicators which are exchanged between SSP and SCP to resolve interactions between IN based
services
-- and network based services, respectively between different IN based services.

ServiceKey: : = Integer4
-- Information that allows the SCF to choose the appropriate service logic.
SFBillingChargingCharacteristics {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
                           bound.&minSFBillingChargingLength..
            bound.&maxSFBillingChargingLength))
-- This parameter indicates the billing and/or charging characteristics for filtered calls.
-- Its content is network operator specific
SubscriberId {PARAMETERS-BOUND: bound}: : = GenericNumber {bound}
SupplementaryServices: : = BIT STRING {
    callingLineIdentificationPresentation (1),
    callingLineIdentificationRestriction (2),
    connectedLineIdentificationPresentation (3),
    connectedLineIdentificationRestriction (4),
    callForwardingOnNoReply (5),
    callForwardingUnconditional (6),
    callForwardingOnBusy (7),
```

```
    callForwardingOnNotReachable (8),
    callWaiting (9),
    callHold (10),
    reverseCharging (11),
    explicitCallTransfer (12),
    callCompletionOnBusySubscriber (13),
    adviceOfChargeOnStart (14),
    adviceOfChargeAtEnd (15),
    adviceOfChargeDuringCall (16),
    timeDependentRouting (17),
    callingPartingDependentRouting (18),
    outgoingCallBarring (19),
    incomingCallBarring (20)
    }
SuspendTimer: : = INTEGER (-1..120) --value in seconds
-- The default is as specified in EN 301 070-1
-- The value -1 indicates the network specific suspend timer (T6) must be used.
TerminalType: : = ENUMERATED {
    unknown(0),
    dialPulse(1),
    dtmf(2),
    isdn(3),
    isdnNoDtmf(4),
    spare(16)
    }
-- Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of
capability
-- (voice recognition, DTMF, display capability, etc.).
TimerID: : = ENUMERATED {
    tssf(0),
    tcusf(1)
    -- others for further study
    }
-- Indicates the timer to be reset.
TimerValue: : = Integer4
-- Indicates the timer value (in seconds).
Tone: : = SEQUENCE {
    toneID  [0] Integer4,
    duration    [1] Integer4    OPTIONAL
    }
-- The duration specifies the length of the tone in seconds, value 0 indicates infinite
duration.
ToneId: : = CHOICE {
    local   [0] Integer4,
    global  [1] OBJECT IDENTIFIER
    }
TraceInformation {PARAMETERS-BOUND: bound}: : = SEQUENCE OF TraceItem { bound}
TraceItem {PARAMETERS-BOUND: bound}: : = SET {scf [0] ScfID { bound},...}
TriggerDataIdentifier {PARAMETERS-BOUND: bound}: : = SEQUENCE {
    triggerID   [0] EventTypeBCSM,
    profileIdentifier   [1] ProfileIdentifier {bound}
    }
-- It is for further study whether al TDP types really apply

UnavailableNetworkResource: : = ENUMERATED {
    unavailableResources(0),
    componentFailure(1),
    basicCallProcessingException(2),
    resourceStatusFailure(3),
    endUserFailure(4)
    }
-- Indicates the network resource that failed.
UserCredit {PARAMETERS-BOUND: bound}: : = Credit {bound}
UserInteractionModes: : = BIT STRING {
    voiceMessage (0),
    tone (1),
    display (2)
    }
USIInformation {PARAMETERS-BOUND: bound}: : = OCTET STRING (SIZE (
            bound.&minUSIInformationLength..bound.&maxUSIInformationLength))
-- Indicates the length of the USIInformation element, maxUSIInformationlength will depend on
the constraints
-- imposed by the network signalliing used to transport the USI information.
USIMonitorMode: : = ENUMERATED {
    monitoringActive   (0),
    monitoringInactive (1)
    }
```

```
-- Indicates if the monitoring relationship for the specified UTSI IE should be
activated or deactivated.

USIServiceIndicator {PARAMETERS-BOUND: bound}: : = CHOICE{
    global  OBJECT IDENTIFIER
    local   OCTET STRING (SIZE (
            bound.&minUSIServiceIndicatorLength..
            bound.&maxUSIServiceIndicatorLength))
            }


VariablePart {PARAMETERS-BOUND: bound}: : = CHOICE {
    integer [0] Integer4,
    number  [1] Digits { bound},    -- Generic digits
    time    [2] OCTET STRING (SIZE(2)),    -- HH: MM, BCD coded
    date    [3] OCTET STRING (SIZE(3)),    -- YYMMDD, BCD coded
    price   [4] OCTET STRING (SIZE(4))
    }
-- Indicates the variable part of the message.
-- BCD coded variable parts are encoded as described in the examples below.
-- For example, time = 12: 15 would be encoded as:
--  Bits                    HGFE        DCBA
--  leading octet   2   1
--          5   1
-- date = 1993 September 30th would be encoded as:
--  Bits                    HGFE        DCBA
--  leading octet   3   9
--          9   0
--          0   3

-- The Definition of range of constants Follows
highLayerCompatibilityLength   INTEGER: : = 2
minCauseLength  INTEGER: : = 2
numOfCounters   INTEGER: : = 100
numOfInfoItems  INTEGER: : = 5
maxCreditUnit   INTEGER: : = 65536
END
```

5.2 Error types

```
IN-CS2-errortypes { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
CS2(20) modules(0) in-cs2-errortypes (1) version1(0)}
-- This module contains the type definitions for the IN CS2 errors.
-- Where a parameter of type CHOICE is tagged with a specific tag value, the tag is
automatically
-- replaced with an EXPLICIT tag of the same value.

DEFINITIONS IMPLICIT TAGS: : =
BEGIN
IMPORTS
ros-InformationObjects, datatypes, errorcodes FROM IN-CS2-object-identifiers
        { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20)
module(0) in-cs2-object-identifiers(17) version1(0) }
    ERROR
FROM Remote-Operations-Information-Objects ros-InformationObjects
    InvokeID,
    UnavailableNetworkResource
FROM IN-CS2-datatypes datatypes
    errcode-canceled,
    errcode-cancelFailed,
    errcode-chainingRefused,
    errcode-eTCFailed,
    errcode-improperCallerResponse,
    errcode-missingCustomerRecord,
    errcode-missingParameter,
    errcode-parameterOutOfRange,
    errcode-requestedInfoError,
    errcode-systemFailure,
    errcode-taskRefused,
    errcode-unavailableResource,
    errcode-unexpectedComponentSequence,
    errcode-unexpectedDataValue,
    errcode-unexpectedParameter,
    errcode-unknownLegID,
    errcode-unknownResource
FROM IN-CS2-errorcodes errorcodes;
```

```
-- TYPE DEFINITION FOR  IN CS2  ERRORS FOLLOWS

canceled ERROR: : = {
    CODE    errcode-canceled
    }
-- The operation has been canceled.
cancelFailed ERROR: : = {
    PARAMETER   SEQUENCE {
            problem [0] ENUMERATED {
                unknownOperation(0),
                tooLate(1),
                operationNotCancellable(2)
                },
            operation   [1] InvokeID
            }
    CODE    errcode-cancelFailed
    }
-- The operation failed to be canceled.
chainingRefused ERROR: : = {
    CODE    errcode-chainingRefused
    }


eTCFailed ERROR: : = {
    CODE    errcode-eTCFailed
    }
-- The establish temporary connection failed.
improperCallerResponse ERROR: : = {
    CODE    errcode-improperCallerResponse
    }
-- The caller response was not as expected.
missingCustomerRecord ERROR: : = {
    CODE    errcode-missingCustomerRecord
    }
-- The Service Logic Program could not be found in the SCF.
missingParameter ERROR: : = {
    CODE    errcode-missingParameter
    }
-- An expected optional parameter was not received.
parameterOutOfRange ERROR: : = {
    CODE    errcode-parameterOutOfRange
    }
-- The parameter was not as expected (e.g. missing or out of range).
requestedInfoError ERROR: : = {
    PARAMETER    ENUMERATED {
            unknownRequestedInfo(1),
            requestedInfoNotAvailable(2)
            -- other values FOR FURTHER STUDY
            }
    CODE    errcode-requestedInfoError
    }
-- The requested information cannot be found.
systemFailure ERROR: : = {
    PARAMETER   UnavailableNetworkResource
    CODE    errcode-systemFailure
    }
-- The operation could not be completed due to a system failure at the serving physical entity.
taskRefused ERROR: : = {
    PARAMETER   ENUMERATED {
            generic(0),
            unobtainable (1),
            congestion(2)
            --other values FOR FURTHER STUDY
            }
    CODE    errcode-taskRefused
    }
-- An entity normally capable of the task requested cannot or chooses not to perform the task at
this
-- time. This includes error situations like congestion and unobtainable address as used in e.g.
the
-- connect operation.)
unavailableResource ERROR: : = {
    CODE    errcode-unavailableResource
    }
-- A requested resource is not available at the serving entity.
unexpectedComponentSequence ERROR: : = {
    CODE    errcode-unexpectedComponentSequence
    }
```

```
-- An incorrect sequence of Components was received (e.g."DisconnectForwardConnection"
-- followed by "PlayAnnouncement").
unexpectedDataValue ERROR: : = {
    CODE    errcode-unexpectedDataValue
    }
-- The data value was not as expected (e.g. routing number expected but billing number received)
unexpectedParameter ERROR: : = {
    CODE    errcode-unexpectedParameter
    }
-- A parameter received was not expected.
unknownLegID ERROR: : = {
    CODE    errcode-unknownLegID
    }
-- Leg not known to the SSF.
unknownResource ERROR: : = {
    CODE    errcode-unknownResource
    }
-- Resource whose status is being requested is not known to the serving entity.

END
```

5.3 Operation codes

```
IN-CS2-operationcodes { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
CS2(20) modules(0) in-cs2-operationcodes (2) version1(0)}
DEFINITIONS: : =
BEGIN

IMPORTS
ros-InformationObjects FROM IN-CS2-object-identifiers
       { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20)
modules(0) in-cs2-object-identifiers(17) version1(0) }
    Code
FROM Remote-Operations-Information-Objects ros-InformationObjects
;

-- the operations are grouped by the identified operation packages.
-- SCF activation Package
    opcode-initialDP       Code: : = local: 0

-- SCF/SRF activation of assist Package
    opcode-assistRequestInstructions      Code: : = local: 16
-- Assist connection establishment Package
    opcode-establishTemporaryConnection   Code: : = local: 17
-- Generic disconnect resource Package
    opcode-disconnectForwardConnection Code: : = local: 18
    opcode-dFCWithArgument     Code: : = local: 86
-- Non-assisted connection establishment Package
    opcode-connectToResource       Code: : = local: 19
-- Connect Package (elementary SSF function)
    opcode-connect     Code: : = local: 20
-- Call handling Package (elementary SSF function)
    opcode-releaseCall     Code: : = local: 22
-- BCSM Event handling Package
    opcode-requestReportBCSMEvent  Code: : = local: 23
    opcode-eventReportBCSM     Code: : = local: 24
-- Charging Event handling Package
    opcode-requestNotificationChargingEvent   Code: : = local: 25
    opcode-eventNotificationCharging   Code: : = local: 26
-- SSF call processing Package
    opcode-collectInformation      Code: : = local: 27
    opcode-continue    Code: : = local: 31

-- SCF call initiation Package
    opcode-initiateCallAttempt     Code: : = local: 32
-- Timer Package
    opcode-resetTimer      Code: : = local: 33
-- Billing Package
    opcode-furnishChargingInformation  Code: : = local: 34
-- Charging Package
    opcode-applyCharging       Code: : = local: 35
    opcode-applyChargingReport     Code: : = local: 36
-- Traffic management Package
    opcode-callGap     Code: : = local: 41
-- Service management Package
    opcode-activateServiceFiltering       Code: : = local: 42
```

```
    opcode-serviceFilteringResponse       Code: : = local: 43
-- Call report Package
    opcode-callInformationReport      Code: : = local: 44
    opcode-callInformationRequest     Code: : = local: 45
-- Signalling control Package
    opcode-sendChargingInformation Code: : = local: 46
-- Specialized resource control Package
    opcode-playAnnouncement       Code: : = local: 47
    opcode-promptAndCollectUserInformation Code: : = local: 48
    opcode-specializedResourceReport   Code: : = local: 49
 -- Cancel Package
    opcode-cancel      Code: : = local: 53
-- Activity Test Package
    opcode-activityTest    Code: : = local: 55
-- CPH Response Package
    opcode-continueWithArgument                       Code: : = local: 88
    opcode-createCallSegmentAssociation   Code: : = local: 89
    opcode-disconnectLeg       Code: : = local: 90
    opcode-mergeCallSegments       Code: : = local: 91
    opcode-moveCallSegments        Code: : = local: 92
    opcode-moveLeg     Code: : = local: 93
    opcode-reconnect       Code: : = local: 94
    opcode-splitLeg    Code: : = local: 95
-- Exception Inform Package
    opcode-entityReleased      Code: : = local: 96
-- Trigger Management Package
    opcode-manageTriggerData       Code: : = local: 97
-- USI Handling Package
    opcode-requestReportUTSI       Code: : = local: 98
    opcode-sendSTUI    Code: : = local: 100
    opcode-reportUTSI      Code: : = local: 101


-- SRF/SCF interface
    opcode-promptAndReceiveMessage Code: : = local: 107
    opcode-scriptInformation       Code: : = local: 108
    opcode-scriptEvent     Code: : = local: 109
    opcode-scriptRun       Code: : = local: 110
    opcode-scriptClose     Code: : = local: 111

-- SCF/SCF interface
    opcode-establishChargingRecord     Code: : = local: 112
    opcode-handlingInformationRequest  Code: : = local: 113
    opcode-handlingInformationResult   Code: : = local: 114
    opcode-networkCapability       Code: : = local: 115
    opcode-notificationProvided        Code: : = local: 116
    opcode-confirmedNotificationProvided   Code: : = local: 117
    opcode-provideUserInformation      Code: : = local: 118
    opcode-confirmedReportChargingInformation Code: : = local: 119
    opcode-reportChargingInformation   Code: : = local: 120
    opcode-requestNotification     Code: : = local: 121

-- CUSF/SCF interface
    opcode-initiateAssociation     Code: : = local: 123
    opcode-releaseAssociation      Code: : = local: 126
    opcode-requestReportBCUSMEvent Code: : = local: 127
    opcode-initialAssociationDP        Code: : = local: 131
    opcode-connectAssociation      Code: : = local: 132
    opcode-continueAssociation     Code: : = local: 133
    opcode-eventReportBCUSM                       Code: : = local: 134
END
```

5.4 Error codes

```
IN-CS2-errorcodes { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
CS2(20) modules(0) in-cs2-errorcodes (3) version1(0)}
DEFINITIONS: : =
BEGIN
IMPORTS
ros-InformationObjects FROM IN-CS2-object-identifiers
        { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20)
modules(0) in-cs2-object-identifiers(17) version1(0) }
    Code
FROM Remote-Operations-Information-Objects ros-InformationObjects
;
    errcode-canceled       Code: : = local: 0
```

```
    errcode-cancelFailed       Code: : = local: 1
    errcode-eTCFailed      Code: : = local: 3
    errcode-improperCallerResponse     Code: : = local: 4
    errcode-missingCustomerRecord      Code: : = local: 6
    errcode-missingParameter       Code: : = local: 7
    errcode-parameterOutOfRange        Code: : = local: 8
    errcode-requestedInfoError     Code: : = local: 10
    errcode-systemFailure      Code: : = local: 11
    errcode-taskRefused    Code: : = local: 12
    errcode-unavailableResource        Code: : = local: 13
    errcode-unexpectedComponentSequence    Code: : = local: 14
    errcode-unexpectedDataValue        Code: : = local: 15
    errcode-unexpectedParameter        Code: : = local: 16
    errcode-unknownLegID       Code: : = local: 17
    errcode-unknownResource        Code: : = local: 18

-- Error codes for the new IN CS2 error types follows
    errcode-scfReferral    Code: : = local: 21
    errcode-scfTaskRefused     Code: : = local: 22
    errcode-chainingRefused        Code: : = local: 23

END
```

5.5 Classes

```
IN-CS2-classes { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20)
modules(0) in-cs2-classes (4) version1(0)}
DEFINITIONS: : =
BEGIN

IMPORTS

    ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, Code, OPERATION,
    CONNECTION-PACKAGE
FROM Remote-Operations-Information-Objects ros-InformationObjects
    emptyBind, emptyUnbind
FROM Remote-Operations-Useful-Definitions ros-UsefulDefinitions

    id-package-emptyConnection,
    id-rosObject-scf,
    id-rosObject-cusf,
    id-rosObject-dssp,
    id-rosObject-srf,
    id-rosObject-ssf,
    ros-InformationObjects,
    ros-UsefulDefinitions,
    ssf-scf-Protocol,
    scf-cusf-Protocol,
    scf-scf-Protocol,
    scf-srf-Protocol,
    scf-sdf-Protocol,
    datatypes
FROM IN-CS2-object-identifiers { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) CS2(20) modules(0) in-cs2-object-identifiers (17) version1(0)}

    inCs2AssistHandofor further studysfToScf,
    inCs2ScfToSsfGeneric,
    inCs2ScfToSsfTrafficManagement,
    inCs2SsfToScfGeneric,
    inCs2SsfToScfServiceManagement
FROM IN-CS2-SSF-SCF-pkgs-contracts-acs ssf-scf-Protocol

    cusf-scf-contract,
    scf-cusf-contract
FROM IN-CS2-SCF-CUSF-pkgs-contracts-acs scf-cusf-Protocol

    dsspContract,
    scf-scfContract
FROM IN-CS2-SCF-SCF-pkgs-contracts-acs scf-scf-Protocol

    srf-scf-contract
FROM IN-CS2-SCF-SRF-pkgs-contracts-acs scf-srf-Protocol

    dapContract
    FROM IN-CS2-SCF-SDF-Protocol scf-sdf-Protocol
    CriticalityType
FROM IN-CS2-datatypes datatypes
```

```
;

ssf ROS-OBJECT-CLASS: : = {
    INITIATES   {inCs2SsfToScfGeneric|
            inCs2AssistHandofor further studysfToScf|
            inCs2SsfToScfServiceManagement}
    RESPONDS    {inCs2ScfToSsfGeneric|
            inCs2ScfToSsfTrafficManagement|
            inCs2SsfToScfServiceManagement|
    ID      id-rosObject-ssf}

srf ROS-OBJECT-CLASS: : = {
    INITIATES   {srf-scf-contract}
    ID      id-rosObject-srf
    }
cusf ROS-OBJECT-CLASS: : = {
    INITIATES          {cusf-scf-contract}
    RESPONDS           {scf-cusf-contract }
    ID                 id-rosObject-cusf}
dssp ROS-OBJECT-CLASS: : = {
    BOTH    {dsspContract}
    ID      id-rosObject-dssp
    }

scf ROS-OBJECT-CLASS: : = {
    INITIATES   {inCs2ScfToSsfGeneric|
            inCs2ScfToSsfTrafficManagement|
            inCs2ScfToSsfServiceManagement|
            inCs2ScfToSsfTriggerManagement|
-- scf to cusf contracts
            scf-cusf-contract |
-- scf to scf contracts
            scf-scfContract |
            dsspContract |
-- sdf to scf contracts
            dapContract
            }
    RESPONDS    {inCs2SsfToScfGeneric|
            inCs2AssistHandofor further studysfToScf|
            inCs2SsfToScfServiceManagement|
-- cusf to scf contracts
            cusf-scf-contract |
-- srf to scf contracts
            srf-scf-contract |
-- scf to scf contracts
            scf-scfContract |
            dsspContract

            }
    ID      id-rosObject-scf}

EXTENSION: : = CLASS {
    &ExtensionType,
    &criticality    CriticalityType DEFAULT ignore,
    &id Code
    }
WITH SYNTAX {
    EXTENSION-SYNTAX    &ExtensionType
    CRITICALITY &criticality
    IDENTIFIED BY   &id
    }

-- Example of addition of an extension named 'Some Network Specific Indicator' of type
-- BOOLEAN, with criticality 'abort' and to be identified as extension number 1
-- Example of definition using the above information object class:
--
-- SomeNetworkSpecificIndicator EXTENSION: : = {
--  EXTENSION-SYNTAX  BOOLEAN
--  CRITICALITY     abort
--  IDENTIFIED BY    local: 1
--  }

-- Example of transfer syntax, using the ExtensionField datatype as specified in subclause 4.1.
-- Assuming the value of the extension is set to TRUE, the extensions parameter
-- becomes a Sequence of type INTEGER: : = 1, criticality ENUMERATED: : = 1 and value [1]
-- EXPLICIT BOOLEAN: : = TRUE.
--
-- Use of Q.1400 defined Extension is for further study
```

```
-- In addition the extension mechanism marker is used to identify the future minor additions to
INAP.
firstExtension EXTENSION: : = {
    EXTENSION-SYNTAX   NULL
    CRITICALITY ignore
    IDENTIFIED BY   local: 1
    }
-- firstExtension is just an example.
SupportedExtensions {PARAMETERS-BOUND: bound} EXTENSION: : = {firstExtension , ...-- full set of
network operator extensions --}
-- SupportedExtension is the full set of the network operator extensions.
UISCRIPT: : = CLASS {
    &SpecificInfo   OPTIONAL,
    &Result OPTIONAL,
    &id Code
    }
WITH SYNTAX {
    [WITH-SPECIFICINFO &SpecificInfo]
    [WITH-RESULT    &Result]
    IDENTIFIED BY   &id
}

firstScript UISCRIPT: : =
    {
    IDENTIFIED BY local: 1
    }
-- firstScript is just an example.
SupportedUIScripts {PARAMETERS-BOUND: bound} UISCRIPT: : = {firstScript , ...-- full set of User
Interaction script --}
-- SupportedUIScripts is the full set of User Interaction scripts.

inEmptyUnbind OPERATION: : = {
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE }
emptyConnectionPackage CONNECTION-PACKAGE: : = {
    BIND    emptyBind
    UNBIND  inEmptyUnbind
    RESPONDER UNBIND    TRUE
    ID      id-package-emptyConnection
    }
PARAMETERS-BOUND: : = CLASS
{
    &minAChBillingChargingLength       INTEGER,
    &maxAChBillingChargingLength       INTEGER,
    &minAssReleaseInfoLength       INTEGER,
    &maxAssReleaseInfoLength       INTEGER,
    &minAttributesLength       INTEGER,
    &maxAttributesLength       INTEGER,
    &minBackwardGVNSLength     INTEGER,
    &maxBackwardGVNSLength     INTEGER,
    &maxBearerCapabilityLength     INTEGER,
    &minCalledPartyBCDNumberLength     INTEGER,
    &maxCalledPartyBCDNumberLength     INTEGER,
    &minCalledPartyNumberLength        INTEGER,
    &maxCalledPartyNumberLength        INTEGER,
    &minCallingPartyBusinessGroupIDLength  INTEGER,
    &maxCallingPartyBusinessGroupIDLength  INTEGER,
    &minCallingPartyNumberLength       INTEGER,
    &maxCallingPartyNumberLength       INTEGER,
    &minCallingPartySubAddressLength       INTEGER,
    &maxCallingPartySubAddressLength       INTEGER,
    &minCallResultLength       INTEGER,
    &maxCallResultLength       INTEGER,
    &minCarrierLength      INTEGER,
    &maxCarrierLength      INTEGER,
    &maxCauseLength    INTEGER,
    &minDigitsLength       INTEGER,
    &maxDigitsLength       INTEGER,
    &minDisplayInformationLength       INTEGER,
    &maxDisplayInformationLength       INTEGER,
    &minEventSpecificInformationChargingLength INTEGER,
    &maxEventSpecificInformationChargingLength INTEGER,
    &minEventTypeChargingLength        INTEGER,
    &maxEventTypeChargingLength        INTEGER,
    &minFCIBillingChargingLength       INTEGER,
    &maxFCIBillingChargingLength       INTEGER,
    &minForwardGVNSLength      INTEGER,
    &maxForwardGVNSLength      INTEGER,
    &minGenericNameLength      INTEGER,
```

```
    &maxGenericNameLength      INTEGER,
    &minGenericNumberLength        INTEGER,
    &maxGenericNumberLength        INTEGER,
    &maxInitialTimeInterval        INTEGER,
    &maxINServiceCompatibilityIndLength   INTEGER,
    &minIPAvailableLength      INTEGER,
    &maxIPAvailableLength      INTEGER,
    &minIPSSPCapabilitiesLength        INTEGER,
    &maxIPSSPCapabilitiesLength        INTEGER,
    &minISDNAccessRelatedInformationLength INTEGER,
    &maxISDNAccessRelatedInformationLength INTEGER,
    &minLocationNumberLength       INTEGER,
    &maxLocationNumberLength       INTEGER,
    &minMailBoxIDLength    INTEGER,
    &maxMailBoxIDLength    INTEGER,
    &minMessageContentLength       INTEGER,
    &maxMessageContentLength       INTEGER,
    &minMidCallControlInfoNum      INTEGER,
    &maxMidCallControlInfoNum      INTEGER,
    &minOriginalCalledPartyIDLength        INTEGER,
    &maxOriginalCalledPartyIDLength        INTEGER,
    &minReasonLength       INTEGER,
    &maxReasonLength       INTEGER,
    &minReceivedInformationLength      INTEGER,
    &maxReceivedInformationLength      INTEGER,
    &maxRecordedMessageUnits       INTEGER,
    &maxRecordingTime      INTEGER,
    &minRedirectingPartyIDLength       INTEGER,
    &maxRedirectingPartyIDLength       INTEGER,
    &minRegistratorIdentifierLength        INTEGER,
    &maxRegistratorIdentifierLength        INTEGER,
    &minRequestedUTSINum       INTEGER,
    &maxRequestedUTSINum       INTEGER,
    &minRouteListLength    INTEGER,
    &maxRouteListLength    INTEGER,
    &minScfIDLength    INTEGER,
    &maxScfIDLength    INTEGER,
    &minScfAddressLength       INTEGER,
    &maxScfAddressLength       INTEGER,
    &minSCIBillingChargingLength       INTEGER,
    &maxSCIBillingChargingLength       INTEGER,
    &minServiceInteractionIndicatorsLength INTEGER,
    &maxServiceInteractionIndicatorsLength INTEGER,
    &minSFBillingChargingLength        INTEGER,
    &maxSFBillingChargingLength        INTEGER,
    &minUSIInformationLength       INTEGER,
    &maxUSIInformationLength       INTEGER,
    &minUSIServiceIndicatorLength      INTEGER,
    &maxUSIServiceIndicatorLength      INTEGER,
    &numOfBCSMEvents       INTEGER,
    &numOfBCUSMEvents      INTEGER,
    &numOfChargingEvents       INTEGER,
    &numOfCSAs     INTEGER,
    &numOfCSs      INTEGER,
    &numOfExtensions       INTEGER,
    &numOfGenericNumbers       INTEGER,
    &numOfInServiceCompatibilityIndLength  INTEGER,
    &numOfLegs     INTEGER,
    &numOfMessageIDs       INTEGER,
    &maxAmount     INTEGER,
    &maxInitialUnitIncrement       INTEGER,
    &maxScalingFactor      INTEGER,
    &maxSegmentsPerDataInterval        INTEGER,
    &maxTimePerInterval    INTEGER,
    &maxUnitsPerDataInterval       INTEGER,
    &maxUnitsPerInterval       INTEGER,
    &ub-maxUserCredit      INTEGER,
    &ub-nbCall     INTEGER
}
WITH SYNTAX
{
    MINIMUM-FOR-ACH-BILLING-CHARGING   &minAChBillingChargingLength
    MAXIMUM-FOR-ACH-BILLING-CHARGING   &maxAChBillingChargingLength
    MINIMUM-FOR-ASSOCIATION-RELEASE-INFO   &minAssReleaseInfoLength
    MAXIMUM-FOR-ASSOCIATION-RELEASE-INFO   &maxAssReleaseInfoLength
    MINIMUM-FOR-ATTRIBUTES     &minAttributesLength
    MAXIMUM-FOR-ATTRIBUTES     &maxAttributesLength
    MAXIMUM-FOR-BACKWARD-GVNS  &minBackwardGVNSLength
    MAXIMUM-FOR-BACKWARD-GVNS  &maxBackwardGVNSLength
```

```
MAXIMUM-FOR-BEARER-CAPABILITY  &maxBearerCapabilityLength
MINIMUM-FOR-CALLED-PARTY-BCD-NUMBER    &minCalledPartyBCDNumberLength
MAXIMUM-FOR-CALLED-PARTY-BCD-NUMBER    &maxCalledPartyBCDNumberLength
MINIMUM-FOR-CALLED-PARTY-NUMBER    &minCalledPartyNumberLength
MAXIMUM-FOR-CALLED-PARTY-NUMBER    &maxCalledPartyNumberLength
MINIMUM-FOR-CALLING-PARTY-BUSINESS-GROUP-ID   &minCallingPartyBusinessGroupIDLength
MAXIMUM-FOR-CALLING-PARTY-BUSINESS-GROUP-ID   &maxCallingPartyBusinessGroupIDLength
MINIMUM-FOR-CALLING-PARTY-NUMBER   &minCallingPartyNumberLength
MAXIMUM-FOR-CALLING-PARTY-NUMBER   &maxCallingPartyNumberLength
MINIMUM-FOR-CALLING-PARTY-SUBADDRESS   &minCallingPartySubAddressLength
MAXIMUM-FOR-CALLING-PARTY-SUBADDRESS   &maxCallingPartySubAddressLength
MINIMUM-FOR-CALL-RESULT        &minCallResultLength
MAXIMUM-FOR-CALL-RESULT        &maxCallResultLength
MINIMUM-FOR-CARRIER    &minCarrierLength
MAXIMUM-FOR-CARRIER    &maxCarrierLength
MAXIMUM-FOR-CAUSE      &maxCauseLength
MINIMUM-FOR-DIGITS     &minDigitsLength
MAXIMUM-FOR-DIGITS     &maxDigitsLength
MINIMUM-FOR-DISPLAY    &minDisplayInformationLength
MAXIMUM-FOR-DISPLAY    &maxDisplayInformationLength
MINIMUM-FOR-EVENT-SPECIFIC-CHARGING    &minEventSpecificInformationChargingLength
MAXIMUM-FOR-EVENT-SPECIFIC-CHARGING    &maxEventSpecificInformationChargingLength
MINIMUM-FOR-EVENT-TYPE-CHARGING    &minEventTypeChargingLength
MAXIMUM-FOR-EVENT-TYPE-CHARGING    &maxEventTypeChargingLength
MINIMUM-FOR-FCI-BILLING-CHARGING   &minFCIBillingChargingLength
MAXIMUM-FOR-FCI-BILLING-CHARGING   &maxFCIBillingChargingLength
MINIMUM-FOR-FORWARD-GVNS       &minForwardGVNSLength
MAXIMUM-FOR-FORWARD-GVNS       &maxForwardGVNSLength
MINIMUM-FOR-GENERIC-NAME       &minGenericNameLength
MAXIMUM-FOR-GENERIC-NAME       &maxGenericNameLength
MINIMUM-FOR-GENERIC-NUMBER &minGenericNumberLength
MAXIMUM-FOR-GENERIC-NUMBER &maxGenericNumberLength
MAXIMUM-FOR-INITIAL-TIME-INTERVAL  &maxInitialTimeInterval
MAXIMUM-FOR-IN-SERVICE-COMPATIBILITY   &maxINServiceCompatibilityIndLength
MINIMUM-FOR-IP-AVAILABLE       &minIPAvailableLength
MAXIMUM-FOR-IP-AVAILABLE       &maxIPAvailableLength
MINIMUM-FOR-IP-SSP-CAPABILITIES    &minIPSSPCapabilitiesLength
MAXIMUM-FOR-IP-SSP-CAPABILITIES    &maxIPSSPCapabilitiesLength
MINIMUM-FOR-ISDN-ACCESS-RELATED-INFO   &minISDNAccessRelatedInformationLength
MAXIMUM-FOR- ISDN-ACCESS-RELATED-INFO &maxISDNAccessRelatedInformationLength
MINIMUM-FOR-LOCATION-NUMBER    &minLocationNumberLength
MAXIMUM-FOR-LOCATION-NUMBER    &maxLocationNumberLength
MINIMUM-FOR-MAIL-BOX-ID        &minMailBoxIDLength
MAXIMUM-FOR-MAIL-BOX-ID        &maxMailBoxIDLength
MINIMUM-FOR-MESSAGE-CONTENT    &minMessageContentLength
MAXIMUM-FOR-MESSAGE-CONTENT    &maxMessageContentLength
MINIMUM-FOR-MID-CALL-CONTROL-INFO  &minMidCallControlInfoNum
MAXIMUM-FOR-MID-CALL-CONTROL-INFO  &maxMidCallControlInfoNum
MINIMUM-FOR-ORIGINAL-CALLED-PARTY-ID   &minOriginalCalledPartyIDLength
MAXIMUM-FOR-ORIGINAL-CALLED-PARTY-ID   &maxOriginalCalledPartyIDLength
MINIMUM-FOR-REASON     &minReasonLength
MAXIMUM-FOR-REASON     &maxReasonLength
MINIMUM-FOR-RECEIVED-INFORMATION   &minReceivedInformationLength
MAXIMUM-FOR-RECEIVED-INFORMATION   &maxReceivedInformationLength
MAXIMUM-FOR-RECORDED-MESSAGE-UNITS &maxRecordedMessageUnits
MAXIMUM-FOR-RECORDING-TIME &maxRecordingTime
MINIMUM-FOR-REDIRECTING-ID     &minRedirectingPartyIDLength
MAXIMUM-FOR-REDIRECTING-ID     &maxRedirectingPartyIDLength
MINIMUM-FOR-REGISTRATOR-IDENTIFIER &minRegistratorIdentifierLength
MAXIMUM-FOR- REGISTRATOR-IDENTIFIER    &maxRegistratoridentifierLength
MINIMUM-FOR-REQUESTED-UTSI-NUM &minRequestedUTSINum
MAXIMUM-FOR-REQUESTED-UTSI-NUM &maxRequestedUTSINum
MINIMUM-FOR-ROUTE-LIST     &minRouteListLength
MAXIMUM-FOR-ROUTE-LIST     &maxRouteListLength
MINIMUM-FOR-SCF-ID     &minScfIDLength
MAXIMUM-FOR-SCF-ID     &maxScfIDLength
MINIMUM-FOR-SCF-ADDRESS        &minScfAddressLength
MAXIMUM-FOR-SCF-ADDRESS        &maxScfAddressLength
MINIMUM-FOR-SCI-BILLING-CHARGING   &minSCIBillingChargingLength
MAXIMUM-FOR-SCI-BILLING-CHARGING   &maxSCIBillingChargingLength
MINIMUM-FOR-SII    &minServiceInteractionIndicatorsLength
MAXIMUM-FOR-SII    &maxServiceInteractionIndicatorsLength
MINIMUM-FOR-SF-BILLING-CHARGING    &minSFBillingChargingLength
MAXIMUM-FOR-SF-BILLING-CHARGING    &maxSFBillingChargingLength
MINIMUM-FOR-USI-INFORMATION    &minUSIInformationLength
MAXIMUM-FOR-USI-INFORMATION    &maxUSIInformationLength
MINIMUM-FOR-USI-SERVICE-INDICATOR  &minUSIServiceIndicatorLength
MAXIMUM-FOR-USI-SERVICE-INDICATOR  &maxUSIServiceIndicatorLength
NUM-OF-BCSM-EVENT      &numOfBCSMEvents
```

```
    NUM-OF-BCUSM-EVENT      &numOfBCUSMEvents
    NUM-OF-CHARGING-EVENT       &numOfChargingEvents
    NUM-OF-CSAS     &numOfCSAs
    NUM-OF-CSS      &numOfCSs
    NUM-OF-EXTENSIONS       &numOfExtensions
    NUM-OF-GENERIC-NUMBERS      &numOfGenericNumbers
    NUM-OF-IN-SERVICE-COMPATIBILITY-ID &numOfInServiceCompatibilityIndLength
    NUM-OF-LEGS     &numOfLegs
    NUM-OF-MESSAGE-IDS      &numOfMessageIDs
    MAXIMUM-FOR-AMOUNT      &maxAmount
    MAXIMUM-FOR-INITIAL-UNIT-INCREMENT &maxInitialUnitIncrement
    MAXIMUM-FOR-SCALING-FACTOR &maxScalingFactor
    MAXIMUM-FOR-SEGMENTS-PER-DATA-INTERVAL &maxSegmentsPerDataInterval
    MAXIMUM-FOR-TIME-PER-INTERVAL  &maxTimePerInterval
    MAXIMUM-FOR-UNITS-PER-DATA-INTERVAL    &maxUnitsPerDataInterval
    MAXIMUM-FOR-UNITS-PER-INTERVAL &maxUnitsPerInterval
    MAXIMUM-FOR-UB-USER-CREDIT      &ub-maxUserCredit
    MAXIMUM-FOR-UB-NB-CALL      &ub-nbCall
}


-- The following instance of the parameter bound is just an example
networkSpecificBoundSet PARAMETERS-BOUND: : =
{
    MINIMUM-FOR-ACH-BILLING-CHARGING   1   -- example value
    MAXIMUM-FOR-ACH-BILLING-CHARGING   5   -- example value
    MINIMUM-FOR-ASSOCIATION-RELEASE-INFO   1   -- example value
    MAXIMUM-FOR-ASSOCIATION-RELEASE-INFO   5   -- example value
    MINIMUM-FOR-ATTRIBUTES      1   -- example value
    MAXIMUM-FOR-ATTRIBUTES      5   -- example value
    MAXIMUM-FOR-BACKWARD-GVNS  1   -- example value
    MAXIMUM-FOR-BACKWARD-GVNS  5   -- example value
    MAXIMUM-FOR-BEARER-CAPABILITY  5   -- example value
    MINIMUM-FOR-CALLED-PARTY-BCD-NUMBER    1   -- example value
    MAXIMUM-FOR-CALLED-PARTY-BCD-NUMBER    5   -- example value
    MINIMUM-FOR-CALLED-PARTY-NUMBER    1   -- example value
    MAXIMUM-FOR-CALLED-PARTY-NUMBER    5   -- example value
    MINIMUM-FOR-CALLING-PARTY-BUSINESS-GROUP-ID   1   -- example value
    MAXIMUM-FOR-CALLING-PARTY-BUSINESS-GROUP-ID   5   -- example value
    MINIMUM-FOR-CALLING-PARTY-NUMBER   1   -- example value
    MAXIMUM-FOR-CALLING-PARTY-NUMBER   5   -- example value
    MINIMUM-FOR-CALLING-PARTY-SUBADDRESS   1   -- example value
    MAXIMUM-FOR-CALLING-PARTY-SUBADDRESS   5   -- example value
    MINIMUM-FOR-CALL-RESULT        1   -- example value
    MAXIMUM-FOR-CALL-RESULT        5   -- example value
    MINIMUM-FOR-CARRIER    1   -- example value
    MAXIMUM-FOR-CARRIER    5   -- example value
    MAXIMUM-FOR-CAUSE      4   -- example value
    MINIMUM-FOR-DIGITS     1   -- example value
    MAXIMUM-FOR-DIGITS     5   -- example value
    MINIMUM-FOR-DISPLAY    1   -- example value
    MAXIMUM-FOR-DISPLAY    5   -- example value
    MINIMUM-FOR-EVENT-SPECIFIC-CHARGING    1   -- example value
    MAXIMUM-FOR-EVENT-SPECIFIC-CHARGING    5   -- example value
    MINIMUM-FOR-EVENT-TYPE-CHARGING    1   -- example value
    MAXIMUM-FOR-EVENT-TYPE-CHARGING    5   -- example value
    MINIMUM-FOR-FCI-BILLING-CHARGING   1   -- example value
    MAXIMUM-FOR-FCI-BILLING-CHARGING   5   -- example value
    MINIMUM-FOR-FORWARD-GVNS       1   -- example value
    MAXIMUM-FOR-FORWARD-GVNS       5   -- example value
    MINIMUM-FOR-GENERIC-NAME       1   -- example value
    MAXIMUM-FOR-GENERIC-NAME       5   -- example value
    MINIMUM-FOR-GENERIC-NUMBER 1   -- example value
    MAXIMUM-FOR-GENERIC-NUMBER 5   -- example value
    MAXIMUM-FOR-INITIAL-TIME-INTERVAL  5   -- example value
    MAXIMUM-FOR-IN-SERVICE-COMPATIBILITY   5   -- example value
    MINIMUM-FOR-IP-AVAILABLE       1   -- example value
    MAXIMUM-FOR-IP-AVAILABLE       5   -- example value
    MINIMUM-FOR-IP-SSP-CAPABILITIES    1   -- example value
    MAXIMUM-FOR-IP-SSP-CAPABILITIES    5   -- example value
    MINIMUM-FOR-ISDN-ACCESS-RELATED-INFO   1   -- example value
    MAXIMUM-FOR- ISDN-ACCESS-RELATED-INFO  5   -- example value
    MINIMUM-FOR-LOCATION-NUMBER    1   -- example value
    MAXIMUM-FOR-LOCATION-NUMBER    5   -- example value
    MINIMUM-FOR-MAIL-BOX-ID        1   -- example value
    MAXIMUM-FOR-MAIL-BOX-ID        5   -- example value
    MINIMUM-FOR-MESSAGE-CONTENT    1   -- example value
    MAXIMUM-FOR-MESSAGE-CONTENT    5   -- example value
    MINIMUM-FOR-MID-CALL-CONTROL-INFO  1   -- example value
```

```
    MAXIMUM-FOR-MID-CALL-CONTROL-INFO  5   -- example value
    MINIMUM-FOR-ORIGINAL-CALLED-PARTY-ID   1   -- example value
    MAXIMUM-FOR-ORIGINAL-CALLED-PARTY-ID   5   -- example value
    MINIMUM-FOR-REASON     1   -- example value
    MAXIMUM-FOR-REASON     5   -- example value
    MINIMUM-FOR-RECEIVED-INFORMATION   1   -- example value
    MAXIMUM-FOR-RECEIVED-INFORMATION   5   -- example value
    MAXIMUM-FOR-RECORDED-MESSAGE-UNITS 5   -- example value
    MAXIMUM-FOR-RECORDING-TIME 5   -- example value
    MINIMUM-FOR-REDIRECTING-ID     1   -- example value
    MAXIMUM-FOR-REDIRECTING-ID     5   -- example value
    MINIMUM-FOR-REGISTRATOR-IDENTIFIER 1   -- example value
    MAXIMUM-FOR- REGISTRATOR-IDENTIFIER    5   -- example value
    MINIMUM-FOR-REQUESTED-UTSI-NUM 1   -- example value
    MAXIMUM-FOR-REQUESTED-UTSI-NUM 5   -- example value
    MINIMUM-FOR-ROUTE-LIST     1   -- example value
    MAXIMUM-FOR-ROUTE-LIST     5   -- example value
    MINIMUM-FOR-SCF-ID     1   -- example value
    MAXIMUM-FOR-SCF-ID     5   -- example value
    MINIMUM-FOR-SCF-ADDRESS        1   -- example value
    MAXIMUM-FOR-SCF-ADDRESS        5   -- example value
    MINIMUM-FOR-SCI-BILLING-CHARGING   1   -- example value
    MAXIMUM-FOR-SCI-BILLING-CHARGING   5   -- example value
    MINIMUM-FOR-SII    1   -- example value
    MAXIMUM-FOR-SII    5   -- example value
    MINIMUM-FOR-SF-BILLING-CHARGING    1   -- example value
    MAXIMUM-FOR-SF-BILLING-CHARGING    5   -- example value
    MINIMUM-FOR-USI-INFORMATION    1   -- example value
    MAXIMUM-FOR-USI-INFORMATION    5   -- example value
    MINIMUM-FOR-USI-SERVICE-INDICATOR  1   -- example value
    MAXIMUM-FOR-USI-SERVICE-INDICATOR  5   -- example value
    NUM-OF-BCSM-EVENT      4   -- example value
    NUM-OF-BCUSM-EVENT     4   -- example value
    NUM-OF-CHARGING-EVENT      4   -- example value
    NUM-OF-CSAS     2   -- example value
    NUM-OF-CSS      2   -- example value
    NUM-OF-EXTENSIONS      1   -- example value
    NUM-OF-GENERIC-NUMBERS     2   -- example value
    NUM-OF-IN-SERVICE-COMPATIBILITY-ID 2   -- example value
    NUM-OF-LEGS     2   -- example value
    NUM-OF-MESSAGE-IDS     2   -- example value
    MAXIMUM-FOR-AMOUNT     2   -- example value
    MAXIMUM-FOR-INITIAL-UNIT-INCREMENT 2   -- example value
    MAXIMUM-FOR-SCALING-FACTOR 2   -- example value
    MAXIMUM-FOR-SEGMENTS-PER-DATA-INTERVAL 5   -- example value
    MAXIMUM-FOR-TIME-PER-INTERVAL  5   -- example value
    MAXIMUM-FOR-UNITS-PER-DATA-INTERVAL    5   -- example value
    MAXIMUM-FOR-UNITS-PER-INTERVAL 5   -- example value
    MAXIMUM-FOR-UB-USER-CREDIT     5   -- example value
    MAXIMUM-FOR-UB-NB-CALL     5   -- example value
}
END
```

5.6 Object IDentifiers (IDs)

```
IN-CS2-object-identifiers { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) CS2(20) modules(0) in-cs2-object-identifiers(17) version1(0)}
DEFINITIONS: : =
BEGIN
-- This module assigns object identifiers for Modules, Packages, Contracts and AC
-- for IN CS2

-- For Modules from TCAP, ROS,
tc-Messages OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation q 773 modules(2) messages(1) version3(3)}
tc-NotationExtensions  OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation q 775 modules(2) notation-extension (4) version1(1)}
ros-InformationObjects OBJECT IDENTIFIER: : =
    {joint-iso-ccitt remote-operations(4) informationObjects(5) version1(0)}
ros-genericPDUs OBJECT IDENTIFIER: : =
    {joint-iso-ccitt remote-operations(4) generic-ROS-PDUs(6) version1(0)}
ros-UsefulDefinitions  OBJECT IDENTIFIER: : =
    {joint-iso-ccitt remote-operations(4) useful-definitions(7) version1(0)}
sese-APDUs  OBJECT IDENTIFIER: : =
    {joint-iso-ccitt genericULS(20) modules(1) seseAPDUs(6) }
guls-Notation   OBJECT IDENTIFIER: : =
    {joint-iso-ccitt genericULS (20) modules (1) notation (1)}
```

```
guls-SecurityTransformations   OBJECT IDENTIFIER: : =
    {joint-iso-itu-t genericULS (20) modules (1) gulsSecurityTransformations (3) }
ds-UsefulDefinitions   OBJECT IDENTIFIER: : =
    {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 3}
spkmGssTokens  OBJECT IDENTIFIER: : =
    { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) spkm(1)
spkmGssTokens(10)}

-- For IN-CS1 Modules
contexts    OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules (0)
contexts (8) selectedContexts (1) version (1)}



-- For IN CS2 Modules
datatypes  OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-datatypes (0) version1(0)}
errortypes  OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-errortypes (1) version1(0)}
operationcodes  OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-operationcodes (2) version1(0)}
errorcodes  OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-errorcodes (3) version1(0)}
classes OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-classes (4) version1(0)}
ssf-scf-Operations OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-ssf-scf-ops-args (5) version1(0)}
ssf-scf-Protocol   OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-ssf-scf-pkgs-contracts-acs (6) version1(0)}
scf-srf-Operations OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-srf-ops-args (7) version1(0)}
scf-srf-Protocol   OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-srf-pkgs-contracts-acs(8)
version1(0)}
sdf-InformationFramework   OBJECT IDENTIFIER: : =
    { ccitt ITU-T Recommendation Q.1228 module(0) sdfInformationFramework(9) version1(0) }
sdf-BasicAccessControl     OBJECT IDENTIFIER: : =
    { ccitt ITU-T Recommendation Q.1228 module(0) sdfBasicAccessControl(10) version1(0) }
scf-sdf-Operations OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation Q.1228 module(0) scf-sdf-operations(11) version1(0) }
scf-sdf-Protocol   OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation q 1218 modules(0) in-scf-sdf-protocol(12) version1(0)}
scf-scf-Operations OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-scf-ops-args (13) version1(0)}
scf-scf-Protocol   OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-scf-pkgs-contracts-acs (14)
version1(0)}
scf-cusf-OperationsOBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-scf-cusf-ops-args (15) version1(0)}
scf-cusf-Protocol  OBJECT IDENTIFIER: : =
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-scf-cusf-pkgs-contracts-acs (16) version1(0)}
object-identifiers OBJECT IDENTIFIER: : =
    {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-identifiers(17) version1(0)}
sdf-sdf-Protocol   OBJECT IDENTIFIER: : =
    { ccitt ITU-T Recommendation Q.1228 module(0) in-cs2-sdf-sdf-Protocol(18) version1(0) }

id-cs2      OBJECT IDENTIFIER: : = {ccitt ITU-T Recommendation Q.1228 cs2 (2)}
id-cs20E    OBJECT IDENTIFIER: : = {ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) CS2(20)}



id-ac           OBJECT IDENTIFIER: : = {id-cs2 ac(3)}
id-acE          OBJECT IDENTIFIER: : = {id-cs20E ac(3)}
id-as           OBJECT IDENTIFIER: : = {id-cs2 as(5)}
id-asE          OBJECT IDENTIFIER: : = {id-cs20E as(5)}
id-rosObject    OBJECT IDENTIFIER: : = {id-cs2 rosObject(25)}
id-contract OBJECT IDENTIFIER: : = {id-cs2 contract(26)}
id-contractE    OBJECT IDENTIFIER: : = {id-cs20E contract(26)}
```

```
id-package  OBJECT IDENTIFIER: : = {id-cs2 package(27)}
id-packageE OBJECT IDENTIFIER: : = {id-cs20E package(27)}

-- for ac, as, rosObject, contract and package, the values are identical to Q1218

id-package-scf-scfConnection   OBJECT IDENTIFIER: : = {id-package 46}
id-package-dsspConnection  OBJECT IDENTIFIER: : = {id-package 47}


-- ROS Objects

id-rosObject-scf    OBJECT IDENTIFIER: : = {id-rosObject 4}
id-rosObject-ssf    OBJECT IDENTIFIER: : = {id-rosObject 5}
id-rosObject-srf    OBJECT IDENTIFIER: : = {id-rosObject 6}
id-rosObject-cusf   OBJECT IDENTIFIER: : = {id-rosObject 7}
id-rosObject-dssp   OBJECT IDENTIFIER: : = {id-rosObject 8}
id-rosObject-sdf    OBJECT IDENTIFIER: : = {id-rosObject 9}

id-rosObject-dua    OBJECT IDENTIFIER: : = {id-rosObject 1}
id-rosObject-directory     OBJECT IDENTIFIER: : = {id-rosObject 2}
id-rosObject-dapDSA    OBJECT IDENTIFIER: : = {id-rosObject 3}

id-rosObject-dspDSA OBJECT IDENTIFIER: : = { id-rosObject 10 }
id-rosObject-initiatingConsumerDSA OBJECT IDENTIFIER: : = { id-rosObject 11 }
id-rosObject-respondingSupplierDSA OBJECT IDENTIFIER: : = { id-rosObject 12 }
id-rosObject-respondingConsumerDSA OBJECT IDENTIFIER: : = { id-rosObject 13 }
id-rosObject-initiatingSupplierDSA OBJECT IDENTIFIER: : = { id-rosObject 14 }



-- ssf/scf AC

id-ac-cs2-ssf-scfGenericAC     OBJECT IDENTIFIER: : = {id-acE 4}
id-ac-cs2-ssf-scfAssistHandoffAC       OBJECT IDENTIFIER: : = {id-acE 6}
id-ac-cs2-ssf-scfServiceManagementAC       OBJECT IDENTIFIER: : = {id-acE 7}
id-ac-cs2-scf-ssfGenericAC     OBJECT IDENTIFIER: : = {id-acE 8}
id-ac-cs2-scf-ssfTrafficManagementAC       OBJECT IDENTIFIER: : = {id-acE 10}
id-ac-cs2-scf-ssfServiceManagementAC       OBJECT IDENTIFIER: : = {id-acE 11}
id-ac-cs2-scf-ssfTriggerManagementAC       OBJECT IDENTIFIER: : = {id-acE 13}

-- srf/scf AC
id-ac-srf-scf   OBJECT IDENTIFIER: : = {id-ac 14}

--  SCF-SDF - ACs --
id-ac-indirectoryAccessAC  OBJECT IDENTIFIER: : = {id-ac 1}
id-ac-indirectoryAccessWith3seAC   OBJECT IDENTIFIER: : = {id-ac 2}
id-ac-inExtendedDirectoryAccessAC  OBJECT IDENTIFIER: : = {id-ac 3}
id-ac-inExtendedDirectoryAccessWith3seAC     OBJECT IDENTIFIER: : = {id-ac 27}

--  SDF - SDF ACs
id-ac-indirectorySystemAC  OBJECT IDENTIFIER: : = { id-ac 15 }
id-ac-inShadowSupplierInitiatedAC  OBJECT IDENTIFIER: : = { id-ac 16 }
id-ac-inShadowConsumerInitiatedAC  OBJECT IDENTIFIER: : = { id-ac 17 }
id-ac-indirectorySystemWith3seAC   OBJECT IDENTIFIER: : = { id-ac 18 }
id-ac-inShadowSupplierInitiatedWith3seAC   OBJECT IDENTIFIER: : = { id-ac 19 }
id-ac-inShadowConsumerInitiatedWith3seAC   OBJECT IDENTIFIER: : = { id-ac 20 }

-- scf/scf AC
id-ac-scf-scfOperationsAC  OBJECT IDENTIFIER: : = {id-ac 21}
id-ac-distributedSCFSystemAC   OBJECT IDENTIFIER: : = {id-ac 22}
id-ac-scf-scfOperationsWith3seAC   OBJECT IDENTIFIER: : = {id-ac 23}
id-ac-distributedSCFSystemWith3seAC    OBJECT IDENTIFIER: : = {id-ac 24}

-- cusf/scf AC
id-ac-scf-cusf  OBJECT IDENTIFIER: : = {id-acE 25}
id-ac-cusf-scf  OBJECT IDENTIFIER: : = {id-acE 26}


-- ssf/scf Contracts

id-inCs2SsfToScfGeneric    OBJECT IDENTIFIER: : = {id-contractE 3}
id-inCs2AssistHandofor further studysfToScf   OBJECT IDENTIFIER: : = {id-contractE 5}
id-inCs2ScfToSsfGeneric    OBJECT IDENTIFIER: : = {id-contractE 6}
id-inCs2ScfToSsfTrafficManagement  OBJECT IDENTIFIER: : = {id-contractE 8}
id-inCs2ScfToSsfServiceManagement  OBJECT IDENTIFIER: : = {id-contractE 9}
id-inCs2SsfToScfServiceManagement  OBJECT IDENTIFIER: : = {id-contractE 10}
id-inCs2ScfToSsfTriggerManagement  OBJECT IDENTIFIER: : = {id-contractE 12}

-- srf/scf Contracts
```

```
id-contract-srf-scfOBJECT IDENTIFIER: : = {id-contract 13}

--  SCF-SDF contracts --
id-contract-dap    OBJECT IDENTIFIER: : = {id-contract 1}
id-contract-dapExecute OBJECT IDENTIFIER: : = {id-contract 2 }

--  SDF - SDF Contracts.
id-contract-indsp  OBJECT IDENTIFIER: : = { id-contract 14 }
id-contract-shadowConsumer OBJECT IDENTIFIER: : = { id-contract 15 }
id-contract-shadowSupplier OBJECT IDENTIFIER: : = { id-contract 17 }

-- scf/scf Contracts
id-contract-scf-scfOBJECT IDENTIFIER: : = {id-contract 18}
id-contract-dssp   OBJECT IDENTIFIER: : = {id-contract 19}

-- cusf/scf Contracts
id-contract-scf-cusf   OBJECT IDENTIFIER: : = {id-contractE 20}
id-contract-cusf-scf   OBJECT IDENTIFIER: : = {id-contractE 21}


-- ssf/scf Operation Packages

id-package-scfActivation       OBJECT IDENTIFIER: : =     {id-package 11}
id-package-srf-scfActivationOfAssist      OBJECT IDENTIFIER: : = {id-package 15}
id-package-assistConnectionEstablishment      OBJECT IDENTIFIER: : = {id-package 16}
id-package-genericDisconnectResource      OBJECT IDENTIFIER: : = {id-package 17}
id-package-nonAssistedConnectionEstablishment OBJECT IDENTIFIER: : = {id-package 18}
id-package-connect     OBJECT IDENTIFIER: : = {id-package 19}
id-package-callHandling        OBJECT IDENTIFIER: : = {id-packageE 20}
id-package-bcsmEventHandling       OBJECT IDENTIFIER: : = {id-package 21}
id-package-chargingEventHandling       OBJECT IDENTIFIER: : = {id-package 23}
id-package-ssfCallProcessing       OBJECT IDENTIFIER: : = {id-packageE 24}
id-package-scfCallInitiation       OBJECT IDENTIFIER: : = {id-package 25}
id-package-timer       OBJECT IDENTIFIER: : = {id-package 26}
id-package-billing     OBJECT IDENTIFIER: : = {id-package 27}
id-package-charging    OBJECT IDENTIFIER: : = {id-package 28}
id-package-trafficManagement       OBJECT IDENTIFIER: : = {id-package 29}
id-package-serviceManagementActivate      OBJECT IDENTIFIER: : = {id-package 30}
id-package-serviceManagementResponse      OBJECT IDENTIFIER: : = {id-package 31}
id-package-callReport      OBJECT IDENTIFIER: : = {id-package 32}
id-package-signallingControl       OBJECT IDENTIFIER: : = {id-package 33}
id-package-activityTest        OBJECT IDENTIFIER: : = {id-package 34}
id-package-cancel      OBJECT IDENTIFIER: : = {id-packageE 36}
id-package-cphResponse     OBJECT IDENTIFIER: : = {id-packageE 37}
id-package-entityReleased      OBJECT IDENTIFIER: : = {id-package 38}
id-package-triggerManagement       OBJECT IDENTIFIER: : = {id-package 39}
id-package-uSIHandling     OBJECT IDENTIFIER: : = {id-package 40}

-- srf/scf Operation Packages
id-package-specializedResourceControl  OBJECT IDENTIFIER: : = { id-package 42}
id-package-srf-scfCancel   OBJECT IDENTIFIER: : = { id-package 43}
id-package-messageControl  OBJECT IDENTIFIER: : = { id-package 44}
id-package-scriptControl   OBJECT IDENTIFIER: : = { id-package 45}

--  SCF-SDF packages --
id-package-search  OBJECT IDENTIFIER: : = {id-package 2}
id-package-modify  OBJECT IDENTIFIER: : = {id-package 3}
id-package-dapConnection   OBJECT IDENTIFIER: : ={id-package 10}
id-package-execute OBJECT IDENTIFIER: : ={id-package 4 }

--  SDF - SDF Packages.
id-package-dspConnection   OBJECT IDENTIFIER: : = { id-package 47 }
id-package-inchainedModify OBJECT IDENTIFIER: : = { id-package 48 }
id-package-inchainedSearch OBJECT IDENTIFIER: : = { id-package 49 }
id-package-chainedExecute  OBJECT IDENTIFIER: : = { id-package 50 }
id-package-dispConnection  OBJECT IDENTIFIER: : = { id-package 51 }
id-package-shadowConsumer  OBJECT IDENTIFIER: : = { id-package 52 }
id-package-shadowSupplier  OBJECT IDENTIFIER: : = { id-package 53 }

-- scf/scf Operation Packages
id-package-handlingInformation OBJECT IDENTIFIER: : = {id-package 54}
id-package-notification    OBJECT IDENTIFIER: : = {id-package 55}
id-package-chargingInformation OBJECT IDENTIFIER: : = {id-package 56}
id-package-userInformation OBJECT IDENTIFIER: : = {id-package 57}
id-package-networkCapability   OBJECT IDENTIFIER: : = {id-package 58}
id-package-chainedSCFOperations    OBJECT IDENTIFIER: : = {id-package 59}

-- cusf/scf Operation Packages
id-package-emptyConnection OBJECT IDENTIFIER: : = { id-package 60}
```

```
id-package-basic-cusf-scf  OBJECT IDENTIFIER: : = { id-packageE 61}
id-package-basic-scf-cusf  OBJECT IDENTIFIER: : = { id-packageE 62}


-- ssf/scf Abstract Syntaxes

id-as-ssf-scfGenericAS OBJECT IDENTIFIER: : =     {id-asE 4}
id-as-assistHandoff-ssf-scfAS  OBJECT IDENTIFIER: : =     {id-asE 6}
id-as-scf-ssfGenericAS OBJECT IDENTIFIER: : =     {id-asE 7}
id-as-scf-ssfTrafficManagementAS   OBJECT IDENTIFIER: : =     {id-asE 9}
id-as-scf-ssfServiceManagementAS   OBJECT IDENTIFIER: : =     {id-asE 10}
id-as-ssf-scfServiceManagementAS   OBJECT IDENTIFIER: : =     {id-asE 11}
id-as-scf-ssfTriggerManagementAS   OBJECT IDENTIFIER: : =     {id-asE 13}


-- srf/scf Abstract Syntaxes
id-as-basic-srf-scfOBJECT IDENTIFIER: : = { id-as 14}
id-as-basic-scf-srfOBJECT IDENTIFIER: : = { id-as 15}

--  SCF-SDF - abstract syntaxes --
id-as-indirectoryOperationsAS  OBJECT IDENTIFIER: : = {id-as 1}
id-as-indirectoryBindingAS OBJECT IDENTIFIER: : = {id-as 2}
id-as-inExtendedDirectoryOperationsAS  OBJECT IDENTIFIER: : = {id-as 3 }
id-as-inSESEAS  OBJECT IDENTIFIER: : = {id-as 25 }

-- SDF-SDF - abstract syntaxes
id-as-indirectorySystemAS  OBJECT IDENTIFIER: : = { id-as 16 }
id-as-indirectoryDSABindingAS  OBJECT IDENTIFIER: : = { id-as 17 }
id-as-indirectoryShadowAS  OBJECT IDENTIFIER: : = { id-as 18 }
id-as-indsaShadowBindingAS OBJECT IDENTIFIER: : = { id-as 19 }

-- scf/scf Abstract Syntaxes
id-as-scf-scfOperationsAS  OBJECT IDENTIFIER: : = {id-as 20}
id-as-distributedSCFSystemAS   OBJECT IDENTIFIER: : = {id-as 21}
id-as-scf-scfBindingAS OBJECT IDENTIFIER: : = {id-as 22}


-- cusf/scf Abstract Syntaxes

id-as-basic-cusf-scf   OBJECT IDENTIFIER: : = { id-asE 23}
id-as-basic-scf-cusf   OBJECT IDENTIFIER: : = { id-asE 24}



-- Object Identifiers for SDF-SDF interface.
-- useful definitions
in-ds       OBJECT IDENTIFIER: : = {ccitt ITU-T Recommendation Q.1228 sdf-objects (10)}

id-avc          OBJECT IDENTIFIER: : = {in-ds 29}
id-aca          OBJECT IDENTIFIER: : = {in-ds 24}
id-soa          OBJECT IDENTIFIER: : = {in-ds 21}

-- Object Identifiers for SDF-SDF interface.

--  SDF Attributes
id-soa-methodRuleUse   OBJECT IDENTIFIER: : = {id-soa 1}
id-aca-prescriptiveACI     OBJECT IDENTIFIER: : = { id-aca 4 }
id-aca-entryACI    OBJECT IDENTIFIER: : = { id-aca 5 }
id-aca-subentryACI OBJECT IDENTIFIER: : = { id-aca 6 }

--  SDF Attribute Value Contexts
id-avc-assignment  OBJECT IDENTIFIER: : = {id-avc 1}

END
```

6 SSF/CCF - SCF Interface

6.1 Operations and arguments

```
IN-CS2-SSF-SCF-ops-args { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
CS2(20) modules(0) in-cs2-ssf-scf-ops-args (5) version1(0)}
DEFINITIONS IMPLICIT TAGS:: =
BEGIN
IMPORTS
```

```
    errortypes, datatypes, operationcodes, classes, ros-InformationObjects
FROM IN-CS2-object-identifiers
    { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0)
in-cs2-object-identifiers(17) version1(0)}
    OPERATION
FROM Remote-Operations-Information-Objects ros-InformationObjects

tc-Messages, classes FROM IN-CS2-object-identifiers
       { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) module(0)
in-cs2-object-identifiers(17) version1(0) }
    InvokeIdType
FROM TCAPMessages tc-Messages

    ChargingPriceInformation,
    ChargingTariffInformation,
    ChargingMessageType

FROM Tarriffing-Data Types {itu-t(0) identified organization etsi (0) xxx("EN number")
version1(1)};

-- Editor note: exact EN number (xxx) still to be defined for DEN/SPS-01049 "ISDN User Part
(ISUP) Support of charging"


    IMSI,
    Ext-BasicServiceCode

FROM MAP-CommonDataTypes { ccitt(0) identified-organization(4) etsi(0) mobileDomain(0)
gsm-Network(1) modules(3) map-CommonDataTypes(18) version3(3)}

    LocationInformation,
    SubscriberState

FROM MAP-MS-DataTypes { ccitt(0) identified-organization(4) etsi(0) mobileDomain(0)
gsm-Network(1) modules(3) map-MS-DataTypes(11) version3(3)}

    CallReferenceNumber,
    SuppressionOfAnnouncement

FROM MAP-CH-DataTypes { ccitt(0) identified-organization(4) etsi(0) mobileDomain(0)
gsm-Network(1) modules(3) map-CH-DataTypes(13) version3(3)};

    ISDN-AddressString,

FROM MAP-CommonDataTypes { ccitt identified-organization(4) etsi(0) mobileDomain(0)
gsm-Network (1) modules (3) map-CommonDataTypes (18) version3 (3)};


       PARAMETERS-BOUND
FROM IN-CS2-classes classes
    opcode-activateServiceFiltering,
    opcode-activityTest,
    opcode-applyCharging,
    opcode-applyChargingReport,
    opcode-assistRequestInstructions,
    opcode-callGap,
    opcode-callInformationReport,
    opcode-callInformationRequest,
    opcode-cancel,
    opcode-collectInformation,
    opcode-connect,
    opcode-connectToResource,
    opcode-continue,
    opcode-continueWithArgument,
    opcode-createCallSegmentAssociation,
    opcode-disconnectForwardConnection,
    opcode-dFCWithArgument,
    opcode-disconnectLeg,
    opcode-entityReleased,
    opcode-establishTemporaryConnection,
    opcode-eventNotificationCharging,
    opcode-eventReportBCSM,
    opcode-furnishChargingInformation,
    opcode-initialDP,
    opcode-initiateCallAttempt,
    opcode-manageTriggerData,
    opcode-mergeCallSegments,
    opcode-moveCallSegments,
    opcode-moveLeg,
```

```
    opcode-releaseCall,
    opcode-reportUTSI,
    opcode-requestNotificationChargingEvent,
    opcode-requestReportBCSMEvent,
    opcode-requestReportUTSI,
    opcode-resetTimer,
    opcode-sendChargingInformation,
    opcode-sendSTUI,
    opcode-serviceFilteringResponse,
    opcode-splitLeg
FROM IN-CS2-operationcodes operationcodes
    ActionIndicator,
    ActionPerformed,
    AChBillingChargingCharacteristics {},
    AdditionalCallingPartyNumber {},
    AlertingPattern,
    AssistingSSPIPRoutingAddress {},
    BackwardGVNS {},
    BCSMEvent {},
    BearerCapability {},
    CalledPartyNumber {},
    CallingPartyBusinessGroupID{},
    CallingPartyNumber {},
    CallingPartysCategory,
    CallingPartySubaddress{},
    CallResult {},
    CallSegmentID {},
    Carrier{},
    Cause {},
    CGEncountered,
    ChargingEvent {},
    ControlType,
    CorrelationID {},
    CountersValue,
    CSAID {},
    CutAndPaste,
    DateAndTime,
    DestinationRoutingAddress {},
    Digits {},
    DisplayInformation {},
    EventSpecificInformationBCSM {},
    EventSpecificInformationCharging {},
    EventTypeBCSM,
    EventTypeCharging {},
    ExtensionField {},
    FCIBillingChargingCharacteristics {},
    FilteredCallTreatment {},
    FilteringCharacteristics,
    FilteringCriteria {},
    FilteringTimeOut,
    ForwardCallIndicators,
    ForwardGVNS {},
    GapCriteria {},
    GapIndicators,
    GapTreatment {},
    GenericName {},
    GenericNumbers {},
    HighLayerCompatibility,
    initialCallSegment,
    INServiceCompatibilityIndication {},
    INServiceCompatibilityResponse,
    Integer4,
    InvokeID,
    IPAvailable {},
    IPRoutingAddress {},
    IPSSPCapabilities {},
    ISDNAccessRelatedInformation{},
    LegID,
    leg1,
    LocationNumber {},
    MiscCallInfo,
    MonitorMode,
    NumberingPlan,
    OriginalCalledPartyID {},
    Reason {},
    RedirectingPartyID {},
    RedirectionInformation,
    RegistratorIdentifier{},
    RequestedInformationList {},
```

```
    RequestedInformationTypeList,
    RequestedUTSIList {},
    ResponseCondition,
    RouteList {},
    ScfID {},
    SCIBillingChargingCharacteristics {},
    ServiceInteractionIndicators {},
    ServiceInteractionIndicatorsTwo,
    ServiceKey,
    TerminalType,
    TimerID,
    TimerValue,
    TriggerDataIdentifier {},
    USIInformation {},
    USIServiceIndicator {}
FROM IN-CS2-datatypes datatypes
    cancelFailed,
    eTCFailed,
    improperCallerResponse,
    missingCustomerRecord,
    missingParameter,
    parameterOutOfRange,
    requestedInfoError,
    systemFailure,
    taskRefused,
    unavailableResource,
    unexpectedComponentSequence,
    unexpectedDataValue,
    unexpectedParameter,
    unknownLegID,
    unknownResource
FROM IN-CS2-errortypes errortypes
;

activateServiceFiltering {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ActivateServiceFilteringArg {bound}
    RETURN RESULT   TRUE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedParameter
            }
    CODE    opcode-activateServiceFiltering
    }
-- Direction: SCF -> SSF, Timer: Tasf
-- When receiving this operation, the SSF handles calls to destination in a specified manner
-- without sending queries for every detected call. It is used for example for providing
-- televoting or mass calling services. Simple registration functionality (counters) and
-- announcement control may be located at the SSF. The operation initializes the specified
-- counters in the SSF.

ActivateServiceFilteringArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    filteredCallTreatment  [0] FilteredCallTreatment {bound},
    filteringCharacteristics   [1] FilteringCharacteristics,
    filteringTimeOut   [2] FilteringTimeOut ,
    filteringCriteria  [3] FilteringCriteria {bound},
    startTime   [4] DateAndTime OPTIONAL,
    extensions  [5] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
                ExtensionField {bound} OPTIONAL,
    ...
    }

activityTest OPERATION:: = {
    RETURN RESULT TRUE
    CODE    opcode-activityTest
    }
-- Direction: SCF -> SSF, Timer: Tat
-- This operation is used to check for the continued existence of a relationship between the SCF
-- and SSF. If the relationship is still in existence, then the SSF will respond. If no reply is
-- received, then the SCF will assume that the SSF has failed in some way and will take the
-- appropriate action.

applyCharging {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ApplyChargingArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            unexpectedComponentSequence |
```

```
            unexpectedParameter |
            unexpectedDataValue |
            parameterOutOfRange |
            systemFailure |
            taskRefused|
            unknownLegID}
    CODE    opcode-applyCharging
    }
-- Direction: SCF -> SSF, Timer: Tac
-- This operation is used for interacting from the SCF with the SSF charging mechanisms. The
ApplyChargingReport
-- operation provides the feedback from the SSF to the SCF.

ApplyChargingArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    aChBillingChargingCharacteristics  [0] AChBillingChargingCharacteristics {bound},
    partyToCharge   [2] LegID   OPTIONAL,
    extensions  [3] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
            ExtensionField {bound}     OPTIONAL,
    ...
    }

-- The partyToCharge parameter indicates the party in the call to which the ApplyCharging
operation
-- should be applied.

applyChargingReport {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ApplyChargingReportArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            unexpectedComponentSequence |
            unexpectedParameter |
            unexpectedDataValue |
            parameterOutOfRange |
            systemFailure |
            taskRefused}
    CODE    opcode-applyChargingReport
    }

-- Direction: SSF -> SCF, Timer: Tacr
-- This operation is used by the SSF to report to the SCF the occurrence of a specific
charging event
-- as requested
-- by the SCF using the ApplyCharging operation.

ApplyChargingReportArg {PARAMETERS-BOUND: bound}:: = CallResult {bound}
-- Note: When the SSF sends the ApplyChargingReport operation as the last event from the Call
Segment, the
--  lastEventIndicator parameter is needed for indicating whether
--  the event is last to the SCF. However, because there is no consideration for the parameter
expansion in the
--  CS1, this parameter cannot be added. There are two alternatives for the solution. One is to
be included
--  into the CallResult parameter. And the other is to specify a new operation with this
parameter. The latter is
--  for further study.


assistRequestInstructions {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    AssistRequestInstructionsArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingCustomerRecord|
            missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter}
    CODE    opcode-assistRequestInstructions
    }
-- Direction: SSF -> SCF or SRF -> SCF, Timer: Tari
-- This operation is used when there is an assist or a hand-off procedure and may be sent by the
SSF
-- or SRF to the SCF. This operation is sent by the assisting SSF to SCF, when the initiating
SSF has
-- set up a connection to the SRF or to the assisting SSF as a result of receiving an
EstablishTemporaryConnection
-- or Connect operation (in the case of hand-off) from the SCF.
-- Refer to clause 17 for a description of the procedures associated with this operation.
```

```
AssistRequestInstructionsArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    correlationID   [0] CorrelationID {bound},
    iPAvailable [1] IPAvailable {bound}    OPTIONAL,
    iPSSPCapabilities   [2] IPSSPCapabilities {bound}  OPTIONAL,
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
             ExtensionField {bound}    OPTIONAL,
    ...
    }
-- OPTIONAL denotes network operator specific use. The value of the correlationID may be the
-- Called Party Number supplied by the initiating SSF.

callGap {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    CallGapArg {bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS    FALSE
    CODE    opcode-callGap
    }
-- Direction: SCF -> SSF, Timer: Tcg
-- This operation is used to request the SSF to reduce the rate at which specific service
requests are sent to
-- the SCF..

CallGapArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    gapCriteria [0] GapCriteria {bound},
    gapIndicators   [1] GapIndicators,
    controlType [2] ControlType OPTIONAL,
    gapTreatment    [3] GapTreatment {bound}   OPTIONAL,
    extensions  [4] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
              ExtensionField {bound}  OPTIONAL,
    ...
    }
-- OPTIONAL denotes network operator optional. If gapTreatment is not present, the SSF will use
-- a default treatment depending on network operator implementation.
callInformationReport {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    CallInformationReportArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-callInformationReport
    }
-- Direction: SSF -> SCF, Timer: Tcirp
-- This operation is used to send specific call information for a single call to the SCF as
requested by the SCF
-- in a previous CallInformationRequest.

CallInformationReportArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    requestedInformationList   [0] RequestedInformationList {bound},
    extensions  [2] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
              ExtensionField {bound}  OPTIONAL,
    legID   [3] LegID   OPTIONAL,
    ...
    }
-- OPTIONAL denotes network operator optional.
callInformationRequest {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    CallInformationRequestArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            requestedInfoError |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter|
            unknownLegID}
    CODE    opcode-callInformationRequest
    }
-- Direction: SCF -> SSF, Timer: Tcirq
-- This operation is used to request the SSF to record specific information about a single call
and report it to
-- the SCF (with a CallInformationReport operation).

CallInformationRequestArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    requestedInformationTypeList   [0] RequestedInformationTypeList,
    extensions  [2] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
              ExtensionField {bound}    OPTIONAL,
    legID   [3] LegID   OPTIONAL,
    ...
    }
-- OPTIONAL denotes network operator optional.
```

```
cancel {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    CancelArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {cancelFailed |
            missingParameter |
            taskRefused}
    CODE    opcode-cancel
    }
-- Direction: SCF -> SSF, or SCF -> SRF, Timer: Tcan
-- This operation cancels the correlated previous operation or all previous requests. The
following operations can be
-- canceled: PlayAnnouncement, PromptAndCollectUserInformation, PromptAndReceiveMessage.
CancelArg {PARAMETERS-BOUND: bound}:: = CHOICE {
    invokeID    [0] InvokeID,
    allRequests [1] NULL,
    callSegmentToCancel[2] SEQUENCE {
            invokeID    [0] InvokeID,
            callSegmentID   [1] CallSegmentID {bound}
            }
    }
-- The InvokeID has the same value as that which was used for the operation to be cancelled.
collectInformation {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    CollectInformationArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter}
    CODE    opcode-collectInformation
    }
-- Direction: SCF -> SSF, Timer: Tci
-- This operation is used to request the SSF to perform the originating basic call processing
actions to prompt
-- a calling party for destination information, then collect destination information according
to a specified
-- numbering plan (e.g. for virtual private networks).

CollectInformationArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    extensions  [4] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    ...
    }


connect {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ConnectArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter}
    CODE    opcode-connect
    }
-- Direction: SCF -> SSF, Timer: Tcon
-- This operation is used to request the SSF to perform the call processing actions to route or
forward a call to
-- a specified destination. To do so, the SSF may or may not use destination information from
the calling party
-- (e.g. dialed digits) and existing call setup information (e.g. route index to a list of trunk
groups), depending on
-- the information provided by the SCF.
--  - When address information is only included in the Connect operation, call processing
resumes at the
--    Analyzed_Information PIC in the O-BCSM.
--  - When address information and routing information is included, call processing resumes at
the
--    Select_Route PIC.

ConnectArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    destinationRoutingAddress  [0] DestinationRoutingAddress { bound},
    alertingPattern [1] AlertingPattern OPTIONAL,
    correlationID   [2] CorrelationID { bound} OPTIONAL,
    cutAndPaste [3] CutAndPaste OPTIONAL,
    iSDNAccessRelatedInformation   [5] ISDNAccessRelatedInformation{ bound}  OPTIONAL,
```

```
    originalCalledPartyID  [6] OriginalCalledPartyID { bound} OPTIONAL,
    routeList   [7] RouteList { bound} OPTIONAL,
    scfID   [8] ScfID { bound} OPTIONAL,
    extensions  [10] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
                ExtensionField {bound}     OPTIONAL,
    carrier [11] Carrier{ bound}   OPTIONAL,
    serviceInteractionIndicators   [26] ServiceInteractionIndicators { bound}OPTIONAL,
    callingPartyNumber [27] CallingPartyNumber { bound}   OPTIONAL,
    callingPartysCategory  [28] CallingPartysCategory OPTIONAL,
    redirectingPartyID [29] RedirectingPartyID { bound}   OPTIONAL,
    redirectionInformation [30] RedirectionInformation    OPTIONAL,
    displayInformation [12] DisplayInformation { bound}   OPTIONAL,
    forwardCallIndicators  [13] ForwardCallIndicators OPTIONAL,
    genericNumbers  [14] GenericNumbers { bound}   OPTIONAL,
    serviceInteractionIndicatorsTwo    [15] ServiceInteractionIndicatorsTwo   OPTIONAL,
    iNServiceCompatibilityResponse [16] INServiceCompatibilityResponse    OPTIONAL,
    forwardGVNS [17] ForwardGVNS { bound}  OPTIONAL,
    backwardGVNS    [18] BackwardGVNS { bound} OPTIONAL,
    callSegmentID   [20] CallSegmentID {bound} OPTIONAL,
    legToBeCreated  [21] LegID  OPTIONAL,
    locationNumber  [50] LocationNumber { bound}   OPTIONAL,
    bearerCapability    [51] BearerCapability { bound} OPTIONAL,
    suppressionOfAnnouncement  [55] SuppressionOfAnnouncement OPTIONAL,
    oCSIApplicable  [56] OCSIApplicableOPTIONAL,
    ...
    }


connectToResource {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ConnectToResourceArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter|
            unknownLegID}
    CODE    opcode-connectToResource
    }
-- Direction: SCF -> SSF, Timer: Tctr
-- This operation is used to connect a call from the SSP to the physical entity containing the
SRF.
-- Refer to clause 17 for a description of the procedures associated with this operation.

ConnectToResourceArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    resourceAddress CHOICE {
        ipRoutingAddress    [0] IPRoutingAddress { bound},
        legID   [1] LegID,
        ipAddressAndLegID   [2] SEQUENCE {
             ipRoutingAddress   [0] IPRoutingAddress {bound},
             legID  [1] LegID
             },
        none    [3] NULL,
        callSegmentID   [5] CallSegmentID { bound} ,
        ipAddressAndCallSegment    [6] SEQUENCE {
             ipRoutingAddress   [0] IPRoutingAddress {bound},
             callSegmentID  [1] CallSegmentID { bound}
             }
        },
    extensions  [4] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    serviceInteractionIndicators   [30] ServiceInteractionIndicators { bound}OPTIONAL,
    serviceInteractionIndicatorsTwo    [7] ServiceInteractionIndicatorsTwo    OPTIONAL,
    ...
    }

continue OPERATION:: = {
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-continue
    }
-- Direction: SCF -> SSF, Timer: Tcue
-- This operation is used to request the SSF to proceed with call processing at the DP at which
it
-- previously suspended call processing to await SCF instructions (i.e. proceed to the next
point
-- in call in the BCSM). The SSF continues call processing without substituting new data from
SCF.
```

```
-- This operation is not valid for a single call segment CSA with more than 2 legs or a multi
call segment CSA

continueWithArgument {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ContinueWithArgumentArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            unknownLegID |
            unexpectedComponentSequence |
            unexpectedParameter |
            unexpectedDataValue
            }
    CODE    opcode-continueWithArgument}
-- Direction: SCF -> SSF, Timer: Tcwa
-- This operation is used to request the SSF to proceed with call processing at the DP a which
it previously
-- suspended call processing to await SCF instructions.
-- It is also used to provide additional service related information to a User (Called Party or
Calling Party) whilst
-- the call processing proceeds.

ContinueWithArgumentArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    legorCSID   CHOICE{
    legID   [0] LegID
    csID    [50] CallsegmentID { bound}}   DEFAULT                              sendingSideID:
leg1,
    alertingPattern [1] AlertingPattern OPTIONAL,
    genericName [2] GenericName { bound}   OPTIONAL,
    iNServiceCompatibilityResponse [3] INServiceCompatibilityResponse OPTIONAL,
    forwardGVNS [4] ForwardGVNS { bound}   OPTIONAL,
    backwardGVNS    [5] BackwardGVNS { bound}  OPTIONAL,
    extensions  [6] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
              ExtensionField {bound}  OPTIONAL,
    serviceInteractionIndicatorsTwo    [7] ServiceInteractionIndicatorsTwo    OPTIONAL,
    locationNumber  [51] LocationNumber { bound}   OPTIONAL,
    ...
    }

createCallSegmentAssociation {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    CreateCallSegmentAssociationArg { bound}
    RESULT  CreateCallSegmentAssociationResult { bound}
    ERRORS  { missingParameter |
            systemFailure|
            taskRefused|
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-createCallSegmentAssociation
    }
-- Direction SCF -> SSF, Timer Tcsa
-- This operation is used to create a new CSA. The new CSA will not contain any Call Segments
after creation.
-- The SSF is responsible for specifying a new CSA identifier for the created CSA which is
unique within
-- the SSF.
CreateCallSegmentAssociationArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    extensions              [0] SEQUENCE SIZE {1..bound.&numOfExtensions) OF
                               ExtensionField {bound} OPTIONAL,
    ...
    }
CreateCallSegmentAssociationResult {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    newCallSegmentAssociation  [0] CSAID { bound},
    extensions  [1] SEQUENCE SIZE {1..bound.&numOfExtensions) OF
              ExtensionField {bound} OPTIONAL,
    ...
    }

disconnectForwardConnection OPERATION:: = {
    RETURN RESULT   FALSE
    ERRORS  {systemFailure |
            taskRefused |
            unexpectedComponentSequence }
    CODE    opcode-disconnectForwardConnection
    }
-- Direction: SCF -> SSF, Timer: Tdfc
-- This operation is used to disconnect a forward temporary connection or a connection to a
resource.
-- Refer to clause 17 for a description of the procedures associated with this operation.
```

```
-- This operation is not valid for a multi call segment CSA.

disconnectForwardConnectionWithArgument {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    DisconnectForwardConnectionWithArgumentArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter |
            unknownLegID}
    CODE    opcode-dFCWithArgument
    }
-- Direction: SCF -> SSF, Timer: Tdfcwa
-- This operation is used to disconnect a forward temporary connection or a connection to a
resource.
-- Refer to clause 17 for a description of the procedures associated with this operation.

DisconnectForwardConnectionWithArgumentArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {

    partyToDisconnect  CHOICE {
            legID   [0] LegID,
            callSegmentID   [1] CallSegmentID { bound}
            },
    extensions  [2] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
               ExtensionField {bound} OPTIONAL,
    ...
    }


disconnectLeg {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    DisconnectLegArg { bound}
    RETURN RESULT TRUE
    ERRORS  {missingParameter|
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter|
            unknownLegID}
    CODE    opcode-disconnectLeg
    }

--  Direction: SCF -> SSF. Timer: T dl
--  This operation is issued by the SCF to release a specific leg associated with the call and
retain any
--  other legs not specified in the DisconnectLeg. Any leg may be disconnected, including the
controlling
--  leg, without completely releasing all legs.
--  Refer to clause 17 for a description of the procedures associated with this operation.

DisconnectLegArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    legToBeReleased [0] LegID,
    releaseCause    [1] Cause { bound} OPTIONAL,
    extensions  [2] SEQUENCE SIZE (1..bound.&numOfExtensions) OF ExtensionField {bound}
    OPTIONAL,
    ...
    }


entityReleased {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    EntityReleasedArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS     FALSE
    CODE    opcode-entityReleased
    }

-- Direction SSF -> SCF, Timer: Ter
-- This operation is used by SSF to inform the SCF of an error/exception

EntityReleasedArg {PARAMETERS-BOUND: bound}:: = CHOICE {
    cSFailure   [0] SEQUENCE{
            callSegmentID   [0] CallSegmentID { bound},
            reason  [1] Reason { bound}OPTIONAL,
            cause   [2] Cause { bound} OPTIONAL
            },
    bCSMFailure [1] SEQUENCE{
            legID   [0] LegID,
```

```
            reason  [1] Reason { bound} OPTIONAL,
            cause   [2] Cause { bound} OPTIONAL
            }
    }
establishTemporaryConnection {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    EstablishTemporaryConnectionArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {eTCFailed |
            missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter|
            unknownLegID}
    CODE    opcode-establishTemporaryConnection
    }
-- Direction: SCF -> SSF, Timer: Tetc
-- This operation is used to create a connection to a resource for a limited period of time
-- (e.g. to play an announcement, to collect user information); it implies the use of the assist
-- procedure. Refer to clause 17 for a description of the procedures associated with this
operation.

EstablishTemporaryConnectionArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    assistingSSPIPRoutingAddress   [0] AssistingSSPIPRoutingAddress { bound},
    correlationID   [1] CorrelationID { bound} OPTIONAL,
    partyToConnect  CHOICE {
            legID   [2] LegID,
            callSegmentID   [7] CallSegmentID { bound}
            }   OPTIONAL,
    scfID   [3] ScfID { bound} OPTIONAL,
    extensions  [4] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
               ExtensionField {bound}  OPTIONAL,
    serviceInteractionIndicators   [30] ServiceInteractionIndicators { bound}OPTIONAL,
    serviceInteractionIndicatorsTwo    [6] ServiceInteractionIndicatorsTwo    OPTIONAL,
    ...
    }

eventNotificationCharging {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    EventNotificationChargingArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-eventNotificationCharging
    }
-- Direction: SSF -> SCF, Timer: Tenc
-- This operation is used by the SSF to report to the SCF the occurence of a specific charging
event
-- type as previously requested by the SCF in a RequestNotificationChargingEvent
operation.

EventNotificationChargingArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    eventTypeCharging  [0] EventTypeCharging { bound},
    eventSpecificInformationCharging   [1] EventSpecificInformationCharging { bound} OPTIONAL,
    legID   [2] LegID   OPTIONAL,
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    eventTypeTariff [4] EventTypeTariffOPTIONAL,
    eventSpecificInformationTariff [5] ChargingMessageType    OPTIONAL,
    monitorMode [30] MonitorMode   DEFAULT notifyAndContinue,
    ...
    }
-- OPTIONAL denotes network operator specific use.

eventReportBCSM {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    EventReportBCSMArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-eventReportBCSM
    }
-- Direction: SSF -> SCF, Timer: Terb
-- This operation is used to notify the SCF of a call-related event (e.g. BCSM events such as
busy or
-- no answer) previously requested by the SCF in a RequestReportBCSMEvent operation.

EventReportBCSMArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    eventTypeBCSM   [0] EventTypeBCSM,
    eventSpecificInformationBCSM   [2] EventSpecificInformationBCSM { bound} OPTIONAL,
    legID   [3] LegID   OPTIONAL,
    miscCallInfo    [4] MiscCallInfo    DEFAULT
```

```
                {messageType request},
    extensions  [5] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    ...
    }


furnishChargingInformation {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    FurnishChargingInformationArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter}
    CODE    opcode-furnishChargingInformation
    }
-- Direction: SCF -> SSF, Timer: Tfci
-- This operation is used to request the SSF to generate, register a call record or to include
some information
-- in the default call
-- record. The registered call record is intended for off line charging of the call.
FurnishChargingInformationArg {PARAMETERS-BOUND: bound}:: = FCIBillingChargingCharacteristics
{bound}

initialDP {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    InitialDPArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingCustomerRecord |
            missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-initialDP
    }
-- Direction: SSF -> SCF, Timer: Tidp
-- This operation is used after a TDP to indicate request for service.

InitialDPArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    serviceKey  [0] ServiceKey  ,
    calledPartyNumber   [2] CalledPartyNumber { bound} OPTIONAL,
    callingPartyNumber [3] CallingPartyNumber { bound}    OPTIONAL,
    callingPartyBusinessGroupID    [4] CallingPartyBusinessGroupID    OPTIONAL,
    callingPartysCategory   [5] CallingPartysCategory   OPTIONAL,
    cGEncountered   [7] CGEncountered   OPTIONAL,
    iPSSPCapabilities   [8] IPSSPCapabilities { bound} OPTIONAL,
    iPAvailable [9] IPAvailable { bound}   OPTIONAL,
    locationNumber  [10] LocationNumber { bound}   OPTIONAL,
    originalCalledPartyID   [12] OriginalCalledPartyID {bound} OPTIONAL,
    terminalType    [14] TerminalType   OPTIONAL,
    extensions  [15] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
                ExtensionField {bound}     OPTIONAL,
    highLayerCompatibility [23] HighLayerCompatibility    OPTIONAL,
    serviceInteractionIndicators    [24] ServiceInteractionIndicators { bound}OPTIONAL,
    additionalCallingPartyNumber    [25] AdditionalCallingPartyNumber { bound}OPTIONAL,
    forwardCallIndicators   [26] ForwardCallIndicators OPTIONAL,
    bearerCapability    [27] BearerCapability { bound} OPTIONAL,
    eventTypeBCSM   [28] EventTypeBCSM OPTIONAL,
    redirectingPartyID [29] RedirectingPartyID { bound}   OPTIONAL,
    redirectionInformation [30] RedirectionInformation    OPTIONAL,
    cause   [17] Cause { bound}OPTIONAL,
    iSDNAccessRelatedInformation    [21] ISDNAccessRelatedInformation  OPTIONAL,
    iNServiceCompatibilityIndication    [22] INServiceCompatibilityIndication { bound}
    OPTIONAL,
    genericNumbers  [31] GenericNumbers { bound}   OPTIONAL,
    serviceInteractionIndicatorsTwo    [32] ServiceInteractionIndicatorsTwo   OPTIONAL,
    forwardGVNS [33] ForwardGVNS { bound}  OPTIONAL,
    createdCallSegmentAssociation  [34] CSAID { bound}OPTIONAL,
    uSIServiceIndicator[35] USIServiceIndicator { bound}  OPTIONAL,
    uSIInformation  [36] USIInformation { bound}   OPTIONAL,
    carrier [37] Carrier    OPTIONAL,
    iMSI    [50] IMSI   OPTIONAL,
    subscriberState [51] SubscriberState    OPTIONAL,
    locationInformation[52] LocationInformation    OPTIONAL,
    ext-basicServiceCode    [53] Ext-BasicServiceCode  OPTIONAL,
```

```
    callReferenceNumber[54] CallReferenceNumber   OPTIONAL,
    mscAddress  [55] ISDN-AddressString    OPTIONAL,
    calledPartyBCDNumber    [56] CalledPartyBCDNumber  OPTIONAL,
    ...
    }


-- OPTIONAL for iPSSPCapabilities, iPAvailable, cGEncountered, and miscCallInfo denotes network
-- operator specific use.
-- OPTIONAL for callingPartyNumber, and callingPartysCategory refer to clause 17 for the trigger
-- detection point processing rules to specify when these parameters are included in the
message.

initiateCallAttempt {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    InitiateCallAttemptArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-initiateCallAttempt
    }
-- Direction: SCF -> SSF, Timer: T_ica
-- This operation is used to request the SSF to create a new call to one call party using
address
-- information provided by the SCF.

InitiateCallAttemptArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    destinationRoutingAddress  [0] DestinationRoutingAddress { bound},
    alertingPattern [1] AlertingPatternOPTIONAL,
    iSDNAccessRelatedInformation   [2] ISDNAccessRelatedInformation   OPTIONAL,
    extensions  [4] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    serviceInteractionIndicators   [29] ServiceInteractionIndicators { bound}OPTIONAL,
    callingPartyNumber [30] CallingPartyNumber { bound}   OPTIONAL,
    legToBeCreated  [5] LegID DEFAULT sendingSideID: leg1,
    newCallSegment  [6] CallSegmentID { bound} DEFAULT initialCallSegment,
    iNServiceCompatibilityResponse [7] INServiceCompatibilityResponse OPTIONAL,
    serviceInteractionIndicatorsTwo    [8] ServiceInteractionIndicatorsTwo    OPTIONAL,
    carrier [50] Carrier { bound}  OPTIONAL,
    locationNumber  [51] LocationNumber { bound}   OPTIONAL,
    bearerCapability    [52] BearerCapability { bound} OPTIONAL,
    ...
    }

manageTriggerData {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ManageTriggerDataArg { bound}
    RESULT  ManageTriggerDataResultArg { bound}
    ERRORS  {missingParameter |
            missingCustomerRecord |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-manageTriggerData
    }
-- Direction: SCF -> SSF, Class 1, Timer: T_mtd
-- This operation is used to activate, deactivate or retrieve
-- the status of a trigger detection point linked to a subscriber profile known at the switch,
e.g. related to an access line.
ManageTriggerDataArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    actionIndicator [0] ActionIndicator,
    triggerDataIdentifier  [1] TriggerDataIdentifier { bound},
    registratorIdentifier  [2] RegistratorIdentifier OPTIONAL,
    extensions  [3] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
                ExtensionField {bound} OPTIONAL,
    ...
    }
```

```
ManageTriggerDataResultArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    actionPerformed [0] ActionPerformed,
    extensions  [1] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
                ExtensionField {bound} OPTIONAL,
    ...
    }


mergeCallSegments {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    MergeCallSegmentsArg { bound}
    RETURN RESULT TRUE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-mergeCallSegments
    }
--  Direction: SCF -> SSF. Timer: T mc
--  This operation is issued by the SCF to merge two associated CSs with a single controlling
leg into one
--  CS with that controlling leg.
--  For additional information on this operation, refer to Q.1224.

MergeCallSegmentsArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    sourceCallSegment  [0] CallSegmentID {bound},
    targetCallSegment  [1] CallSegmentID {bound} DEFAULT initialCallSegment,
    extensions  [2] SEQUENCE SIZE (1..bound.&numOfExtensions)
              OF ExtensionField {bound}    OPTIONAL,
    ...
    }

moveCallSegments {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    MoveCallSegmentsArg { bound}
    RETURN RESULT TRUE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter|
            unknownLegID
            }
    CODE    opcode-moveCallSegments
    }
-- Direction: SCF -> SSF, Timer Tmcs
--This operation moves a CS from the source CSA to the target CSA
MoveCallSegmentsArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    targetCallSegmentAssociation   [0] CSAID { bound},
-- assignement of CSAID by SSF/SCF is for further study
    callSegments    [1] SEQUENCE SIZE (1..bound.&numOfCSs) OF SEQUENCE {
        sourceCallSegment  [0] CallSegmentID { bound} DEFAULT initialCallSegment,
        newCallSegment [1] CallSegmentID { bound}
        },
    legs    [2] SEQUENCE SIZE (1..bound.&numOfLegs) OF SEQUENCE {
        sourceLeg  [0] LegID,
        newLeg  [1] LegID
        },
    extensions  [2] SEQUENCE SIZE (1..bound.&numOfExtensions)
                OF ExtensionField {bound}  OPTIONAL,
    ...
    }
-- The double use of the tag value 2 is from pure ASN.1 point of view correct. From the way the
tagging is used within
-- INAP however it is incorrect(sequential tagging is used). As the possibility that ITU-T
adapts its tagging
-- is rather small no adaptation will be done here (yet).
moveLeg {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    MoveLegArg { bound}
    RETURN RESULT TRUE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter|
            unknownLegID
```

```
            }
    CODE    opcode-moveLeg
    }
-- Direction: SCF ->SSF, Timer: T ml
-- This operation is issued by the SCF to move a leg from one CS to another with which it is
associated.
MoveLegArg {PARAMETERS-BOUND: bound}:: =SEQUENCE {
    legIDToMove [0] LegID,
    targetCallSegment  [1] CallSegmentID { bound} DEFAULT 1,
    extensions  [2] SEQUENCE SIZE (1..bound.&numOfExtensions)     OF
                ExtensionField {bound} OPTIONAL,
    ...
    }
-- For the OPTIONAL parameters, refer to clause 17 for the trigger
-- detection point processing rules to specify when these parameters are
-- included in the message.

releaseCall {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ReleaseCallArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-releaseCall
    }
-- Direction: SCF -> SSF, Timer: Trc
-- This operation is used to tear down an existing call at any phase of the call for all parties
-- involved in the call.
ReleaseCallArg {PARAMETERS-BOUND: bound}:: = CHOICE {
    initialCallSegment Cause { bound},
    associatedCallSegment  [1] SEQUENCE {
        callSegment [0] INTEGER (2..bound.&numOfCSs),
        releaseCause    [1] Cause { bound} OPTIONAL
        },
    allCallSegments [2] SEQUENCE {
         releaseCause   [0] Cause { bound} OPTIONAL
        }
    }
-- A default value of decimal 31 (normal unspecified) should be coded appropriately.


reportUTSI {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ReportUTSIArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-reportUTSI
    }
-- Direction: SSF -> SCF. Timer: Tru
-- This operation is issued by the SSF in the context of the USI feature. It is used to report
the receipt
-- of a User to Service Information (UTSI) IE to the SCF.

ReportUTSIArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    uSIServiceIndicator [0] USIServiceIndicator { bound},
    legID   [1] LegID   DEFAULT receivingSideID: leg1,
    uSIInformation  [2] USIInformation { bound},
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
                ExtensionField {bound}  OPTIONAL,
    ...
    }

requestNotificationChargingEvent {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    RequestNotificationChargingEventArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-requestNotificationChargingEvent
    }
-- Direction: SCF->SSF, Timer: Trnc
-- This operation is used by the SCF to instruct the SSF on how to manage the charging events
-- which are received rom other FE's and not under control of the service logic instance.
RequestNotificationChargingEventArg {PARAMETERS-BOUND: bound}:: = SEQUENCE
SIZE(1..bound.&numOfChargingEvents) OF
         ChargingEvent {bound}
requestReportBCSMEvent {PARAMETERS-BOUND: bound} OPERATION:: = {
```

```
    ARGUMENT    RequestReportBCSMEventArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-requestReportBCSMEvent
    }
-- Direction: SCF -> SSF, Timer: Trrb
-- This operation is used to request the SSF to monitor for a call-related event (e.g. BCSM
events such as
-- busy or no answer), then send a notification back to the SCF when the event is detected.
-- NOTE:    -
    - Every EDP must be explicitly armed by the SCF via a RequestReportBCSMEvent operation. No
--      implicit arming of EDPs at the SSF after reception of any operation (different from
--      RequestReportBCSMEvent) from the SCF is allowed.

RequestReportBCSMEventArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    bcsmEvents  [0] SEQUENCE SIZE(1..bound.&numOfBCSMEvents) OF
                BCSMEvent {bound},
    extensions  [2] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    ...
    }
-- Indicates the BCSM related events for notification.
requestReportUTSI {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    RequestReportUTSIArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-requestReportUTSI
    }

-- Direction: SCF -> SSF. Timer: Trru
-- This operation is issued by the SCF in the context of the USI feature to request the SSF to
monitor for
-- a User to Service Information (UTSI) information element, which are received from a user.
RequestReportUTSIArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    requestedUTSIList  [0] RequestedUTSIList { bound},
    extensions  [1] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound} OPTIONAL,
    legID   [2] LegID   DEFAULT
                receivingSideID: leg1,
    ...
    }


resetTimer {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ResetTimerArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-resetTimer
    }
-- Direction: SCF->SSF, Timer: Trt
-- This operation is used to request the SSF to refresh an application timer in the SSF.

ResetTimerArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    timerID [0] TimerID DEFAULT tssf,
    timervalue  [1] TimerValue,
    extensions  [2] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    callSegmentID   [3] CallSegmentID { bound} OPTIONAL,
    ...
    }
```

```
sendChargingInformation {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    SendChargingInformationArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            unexpectedComponentSequence |
            unexpectedParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unknownLegID
            }
    CODE    opcode-sendChargingInformation
    }
-- Direction: SCF -> SSF, Timer: Tsci
-- This operation is used to instruct the SSF on the charging information to send by the SSF.
-- The charging information can either be sent back by means of signalling or internal
-- if the SSF is located in the local exchange. In the local exchange
-- this information may be used to update the charge meter or to create a standard call record.

SendChargingInformationArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    sCIBillingChargingCharacteristics  [0] SCIBillingChargingCharacteristics { bound},
    partyToCharge   [1] LegID,
    extensions  [2] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    tariffMessage   [3] ChargingMessageType    OPTIONAL,
    ...
    }

sendSTUI {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    SendSTUIArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            unexpectedComponentSequence |
            unexpectedParameter |
            unexpectedDataValue |
            systemFailure |
            taskRefused |
            unknownLegID
            }
    CODE    opcode-sendSTUI
    }
-- Direction: SCF -> SSF. Timer: Tss
-- This operation is issued by the SCF in the context of the USI feature. It is used to request
the SSF
-- to send a Service to User Information (STUI) information element to the indicated user.

SendSTUIArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    uSIServiceIndicator[0] USIServiceIndicator { bound},
    legID   [1] LegID   DEFAULT sendingSideID: leg1,
    uSIInformation  [2] USIInformation { bound},
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
                ExtensionField {bound} OPTIONAL,
    ...
    }

serviceFilteringResponse {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ServiceFilteringResponseArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-serviceFilteringResponse
    }
-- Direction: SSF->SCF, Timer: Tsfr
-- This operation is used to send back to the SCF the values of counters specified in a previous
-- ActivateServiceFiltering operation

ServiceFilteringResponseArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    countersValue   [0] CountersValue,
    filteringCriteria   [1] FilteringCriteria { bound},
    extensions  [2] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}     OPTIONAL,
    responseCondition   [3] ResponseCondition   OPTIONAL,
    ...
    }

splitLeg {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    SplitLegArg { bound}
    RETURN RESULT TRUE
    ERRORS  {missingParameter |
```

```
            unexpectedComponentSequence |
            unexpectedParameter |
            unexpectedDataValue |
            systemFailure |
            taskRefused |
            unknownLegID
            }
    CODE    opcode-splitLeg
    }
-- Direction: SCF -> SSF. Timer: T sl
-- This operation is issued by the SCF to separate one joined leg from a multi-way connection
-- or to interrupt the bearer connection between the involved legs of a single 2 party Call
segment.
SplitLegArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    legToBeSplit    [0] LegID,
    newCallSegment  [1] INTEGER (2..bound.&numOfCSs),
    extensions  [2] SEQUENCE SIZE (1..bound.&numOfExtensions) OF ExtensionField {bound}
    OPTIONAL,
    ...
    }

END
```

The following value ranges do apply for operation specific timers in INAP:

short: 1 - 10 s

medium: 1 - 60 s

long: 1 s - 30 minutes

for further study: For Further Study

Table 5-1 lists all operation timers and the value range for each timer. The definitive value for each operation timer may be network specific and has to be defined by the network operator.

Table 5-1: Timer value ranges

Operation Name	Timer	value range
ActivateServiceFiltering	T_{asf}	medium
ActivityTest	T_{at}	short
ApplyCharging	T_{ac}	short
ApplyChargingReport	T_{acr}	short
AssistRequestInstructions	T_{ari}	short
CallGap	T_{cg}	short
CallInformationReport	T_{cirp}	short
CallInformationRequest	T_{cirq}	short
Cancel	T_{can}	short
CollectInformation	T_{ci}	medium
Connect	T_{con}	short
ConnectToResource	T_{ctr}	short
Continue	T_{cue}	short
ContinueWithArgument	Tcwa	short
CreateCallSegmentAssociation	T_{csa}	short
DisconnectForwardConnection	T_{dfc}	short
DisconnectForwardConnectionWithArgument	T_{dfcwa}	short
DisconnectLeg	T_{dl}	short
EntityReleased	T_{er}	short
EstablishTemporaryConnection	T_{etc}	medium
EventNotificationCharging	T_{enc}	short
EventReportBCSM	T_{erb}	short
FurnishChargingInformation	T_{fci}	short
InitialDP	T_{idp}	short
InitiateCallAttempt	T_{ica}	short
ManageTriggerData	T_{mtd}	medium
MergeCallSegments	T_{mc}	short
MoveCallSegments	T_{mcs}	short
MoveLeg	T_{ml}	short
ReleaseCall	T_{rc}	short
ReportUTSI	T_{ru}	short
RequestNotificationChargingEvent	T_{rnc}	short
RequestReportBCSMEvent	T_{rrb}	short
RequestReportUTSI	T_{rru}	short
ResetTimer	T_{rt}	short
SendChargingInformation	T_{sci}	short
SendSTUI	T_{ss}	short
ServiceFilteringResponse	T_{sfr}	short
SplitLeg	T_{sl}	short

6.2 SSF/SCF packages, contracts and ACs

6.2.1 Protocol overview

The **inCs2SsfToScfGeneric** contract expresses the form of the service in which the SSF, a Remote Operations Service (ROS)-object of class **ssf**, initiates the generic triggering approach contract. A ROS-object of class **scf**, responds to this contract.

```
inCs2SsfToScfGeneric CONTRACT:: = {
-- dialogue initiated by SSF with InitialDP Operation
    INITIATOR CONSUMER OF  {exceptionInformPackage {networkSpecificBoundSet} |
            scfActivationPackage {networkSpecificBoundSet} }
    RESPONDER CONSUMER OF  {activityTestPackage|
            assistConnectionEstablishmentPackage {networkSpecificBoundSet} |
            bcsmEventHandlingPackage {networkSpecificBoundSet} |
            billingPackage {networkSpecificBoundSet} |
            callHandlingPackage {networkSpecificBoundSet} |
            callReportPackage {networkSpecificBoundSet} |
            cancelPackage {networkSpecificBoundSet} |
```

```
            chargingEventHandlingPackage {networkSpecificBoundSet} |
            chargingPackage {networkSpecificBoundSet} |
            connectPackage {networkSpecificBoundSet} |
            cphResponsePackage {networkSpecificBoundSet} |
            genericDisconnectResourcePackage {networkSpecificBoundSet} |
            nonAssistedConnectionEstablishmentPackage {networkSpecificBoundSet} |
            scfCallInitiationPackage {networkSpecificBoundSet} |
            signallingControlPackage {networkSpecificBoundSet} |
            specializedResourceControlPackage {networkSpecificBoundSet} |
            scriptControlPackage {networkSpecificBoundSet} |
            messageControlPackage {networkSpecificBoundSet} |
            ssfCallProcessingPackage {networkSpecificBoundSet} |
            timerPackage {networkSpecificBoundSet} |
            trafficManagementPackage {networkSpecificBoundSet} |
            uSIHandlingPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2SsfToScfGeneric
    }
```

The **inCs2AssistHandofor further studysfToScf** contract expresses the form of the service in which the SSF, a ROS-object of class **ssf**, initiates the Assist or Hand-off contract. A ROS-object of class **scf**, responds to this contract.

```
inCs2AssistHandofor further studysfToScf CONTRACT:: = {
-- dialogue initiated by SSF with AssistRequestInstructions
    INITIATOR CONSUMER OF  {srf-scfActivationOfAssistPackage {networkSpecificBoundSet} }
    RESPONDER CONSUMER OF  {activityTestPackage|
            billingPackage {networkSpecificBoundSet} |
            callHandlingPackage {networkSpecificBoundSet} |
            cancelPackage {networkSpecificBoundSet} |
            chargingPackage {networkSpecificBoundSet} |
            genericDisconnectResourcePackage {networkSpecificBoundSet} |
            nonAssistedConnectionEstablishmentPackage {networkSpecificBoundSet} |
            specializedResourceControlPackage {networkSpecificBoundSet} |
            scriptControlPackage {networkSpecificBoundSet} |
            messageControlPackage {networkSpecificBoundSet} |
            timerPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2AssistHandofor further studysfToScf
    }
```

The **inCs2ScfToSsfGeneric** contract expresses the form of the service in which the SCF, a ROS-object of class **scf**, initiates the generic messaging approach for the SCF Initiate Call Attempt contract. A ROS-object of class **ssf** responds to this contract.

```
inCs2ScfToSsfGeneric CONTRACT:: = {
-- dialogue initiated by SCF with InitiateCallAttempt, Generic Case
    INITIATOR CONSUMER OF  {activityTestPackage|
            assistConnectionEstablishmentPackage {networkSpecificBoundSet} |
            bcsmEventHandlingPackage {networkSpecificBoundSet} |
            billingPackage {networkSpecificBoundSet} |
            callHandlingPackage {networkSpecificBoundSet} |
            callReportPackage {networkSpecificBoundSet} |
            cancelPackage {networkSpecificBoundSet} |
            chargingPackage {networkSpecificBoundSet} |
            connectPackage {networkSpecificBoundSet} |
            cphResponsePackage {networkSpecificBoundSet} |
            genericDisconnectResourcePackage {networkSpecificBoundSet} |
            nonAssistedConnectionEstablishmentPackage {networkSpecificBoundSet} |
            scfCallInitiationPackage {networkSpecificBoundSet} |
            signallingControlPackage {networkSpecificBoundSet} |
            specializedResourceControlPackage {networkSpecificBoundSet} |
            scriptControlPackage {networkSpecificBoundSet} |
            messageControlPackage {networkSpecificBoundSet} |
            ssfCallProcessingPackage {networkSpecificBoundSet} |
            timerPackage {networkSpecificBoundSet} |
            uSIHandlingPackage {networkSpecificBoundSet}
            }
    RESPONDER CONSUMER OF  {exceptionInformPackage {networkSpecificBoundSet} }
    ID      id-inCs2ScfToSsfGeneric
    }
```

The **inCs2ScfToSsfTrafficManagement** contract expresses the form of the service in which the SCF, a ROS-object of class **scf**, initiates the Traffic Management related contract. A ROS-object of class **ssf** responds to this contract.

```
inCs2ScfToSsfTrafficManagement CONTRACT:: = {
-- dialogue initiated by SCF with CallGap
    INITIATOR CONSUMER OF  {trafficManagementPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2ScfToSsfTrafficManagement
    }
```

The **inCs2ScfToSsfServiceManagement** contract expresses the form of the service in which the SCF, a ROS-object of class **scf**, initiates the Service Management related contract. A ROS-object of class **ssf**, in the context of a separate contract, responds to this initiation.

```
inCs2ScfToSsfServiceManagement CONTRACT:: = {
-- dialogue initiated by SCF with ActivateServiceFiltering
    INITIATOR CONSUMER OF  {serviceManagementActivatePackage {networkSpecificBoundSet}
            }
    ID      id-inCs2ScfToSsfServiceManagement
    }
```

The **inCs2SsfToScfServiceManagement** expresses the form of the service in which the SSF, a ROS-object of class **ssf**, initiates the Service Management related contract for reporting Service Management results.

```
inCs2SsfToScfServiceManagement CONTRACT:: = {
-- dialogue initiated/ended by SSF with ServiceFilteringResponse
    INITIATOR CONSUMER OF  {serviceManagementResponsePackage {networkSpecificBoundSet}
            }
    ID      id-inCs2SsfToScfServiceManagement
    }
```

The **inCs2ScfToSsfTriggerManagement** contract expresses the form of the service in which the SCF, a ROS-object of class **scf**, initiates the Trigger Management related contract. A ROS-object of class **ssf**, in the context of a separate contract, responds to this initiation.

```
inCs2ScfToSsfTriggerManagement CONTRACT:: = {
-- dialogue initiated by SCF with Manage Trigger Data
    INITIATOR CONSUMER OF  {triggerManagementPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2ScfToSsfTriggerManagement
    }
```

6.2.1.1 SSF/SCF operation packages

The operation packages below are defined as information objects of class OPERATION-PACKAGE. The operations of these packages are defined in subclause 5.1.

```
scfActivationPackage {PARAMETERS-BOUND: bound}  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {initialDP {bound}}
    ID      id-package-scfActivation}
 srf-scfActivationOfAssistPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {assistRequestInstructions {bound}}
    ID      id-package-srf-scfActivationOfAssist}
assistConnectionEstablishmentPackage {PARAMETERS-BOUND: bound}   OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {establishTemporaryConnection {bound}}
    ID      id-package-assistConnectionEstablishment}
genericDisconnectResourcePackage {PARAMETERS-BOUND: bound}   OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {bound}}
    ID      id-package-genericDisconnectResource}
nonAssistedConnectionEstablishmentPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {connectToResource {bound}}
    ID      id-package-nonAssistedConnectionEstablishment}
connectPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {connect {bound}}
    ID      id-package-connect}
callHandlingPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   { releaseCall {bound}}
    ID      id-package-callHandling}
bcsmEventHandlingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestReportBCSMEvent {bound}}
    SUPPLIER INVOKES   {eventReportBCSM {bound}}
    ID      id-package-bcsmEventHandling}
chargingEventHandlingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestNotificationChargingEvent {bound}}
    SUPPLIER INVOKES   {eventNotificationCharging {bound}}
    ID      id-package-chargingEventHandling}
ssfCallProcessingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
```

```
    CONSUMER INVOKES   {collectInformation {bound} | continue}
    ID      id-package-ssfCallProcessing}
scfCallInitiationPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {initiateCallAttempt {bound}}
    ID      id-package-scfCallInitiation}
timerPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {resetTimer {bound}}
    ID      id-package-timer}
billingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {furnishChargingInformation {bound}}
    ID      id-package-billing}
chargingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {applyCharging {bound}}
    SUPPLIER INVOKES   {applyChargingReport {bound}}
    ID      id-package-charging}
trafficManagementPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {callGap {bound}}
    ID      id-package-trafficManagement}
serviceManagementActivatePackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {activateServiceFiltering {bound}}
    ID      id-package-serviceManagementActivate}
serviceManagementResponsePackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {serviceFilteringResponse {bound}}
    ID      id-package-serviceManagementResponse}
callReportPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {callInformationRequest {bound}}
    SUPPLIER INVOKES   {callInformationReport {bound}}
    ID      id-package-callReport}
signallingControlPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {sendChargingInformation {bound}}
    ID      id-package-signallingControl}
activityTestPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {activityTest}
    ID      id-package-activityTest}
 cancelPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {cancel {bound}}
    ID      id-package-cancel}
cphResponsePackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {
            continueWithArgument {bound}|
            disconnectLeg {bound}|
            mergeCallSegments {bound}|
            moveCallSegments {bound}|
            moveLeg {bound}|
            createCallSegmentAssociation {bound} |
            splitLeg {bound}
            }
    ID      id-package-cphResponse}
exceptionInformPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {entityReleased}
    ID      id-package-entityReleased}

triggerManagementPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {manageTriggerData {bound}}
    ID      id-package-triggerManagement}

uSIHandlingPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestReportUTSI | sendSTUI}
    SUPPLIER INVOKES   {reportUTSI}
    ID      id-package-uSIHandling
    }
```

6.2.1.2 Abstract syntax

This version of the INAP requires the support of two abstract syntaxes:

a) the abstract syntax of Transaction Capabilities (TC) dialogue control PDUs, **dialogue-abstract-syntax**, which is needed to establish the dialogues between FEs and specified in ETS 300 287-1 [7];

b) the abstract syntax for conveying the PDUs for invoking the operations involved in the operation packages specified in subclause 5.2.2 and reporting their outcome.

The ASN.1 type from which the values of the last abstract syntax are derived is specified using the parameterized type **TCMessage {}** defined in ETS 300 287-1 [7].

All these abstract syntaxes shall (as a minimum) be encoded according to the Basic ASN.1 encoding rules with the restrictions listed in ETS 300 287-1 [7].

The SSF-SCF INAP Packages that realize the operation packages specified as above share the following abstract syntaxes. These are specified as information objects of the class ABSTRACT-SYNTAX.

```
ssf-scfGenericAbstractSyntax ABSTRACT-SYNTAX:: = {
    GenericSSF-SCF-PDUs
    IDENTIFIED BY   id-as-ssf-scfGenericAS}

GenericSSF-SCF-PDUs:: = TCMessage {{SsfToScfGenericInvokable},
            {SsfToScfGenericReturnable}}
SsfScfGenericInvokable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet} |
            activityTest |
            applyCharging {networkSpecificBoundSet} |
            applyChargingReport {networkSpecificBoundSet} |
            callInformationReport {networkSpecificBoundSet} |
            callInformationRequest {networkSpecificBoundSet} |
            cancel {networkSpecificBoundSet} |
            collectInformation {networkSpecificBoundSet} |
            connect {networkSpecificBoundSet} |
            connectToResource {networkSpecificBoundSet} |
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet} |
            entityReleased {networkSpecificBoundSet} |
            establishTemporaryConnection {networkSpecificBoundSet} |
            eventNotificationCharging {networkSpecificBoundSet} |
            eventReportBCSM {networkSpecificBoundSet} |
            furnishChargingInformation {networkSpecificBoundSet} |
            initialDP {networkSpecificBoundSet} |
            mergeCallSegments {networkSpecificBoundSet} |
            moveCallSegments {networkSpecificBoundSet} |
            moveLeg {networkSpecificBoundSet} |
            createCallSegmentAssociation {networkSpecificBoundSet} |
            releaseCall {networkSpecificBoundSet} |
            reportUTSI {networkSpecificBoundSet} |
            requestNotificationChargingEvent {networkSpecificBoundSet} |
            requestReportBCSMEvent {networkSpecificBoundSet} |
            requestReportUTSI {networkSpecificBoundSet} |
            resetTimer {networkSpecificBoundSet} |
            sendChargingInformation {networkSpecificBoundSet} |
            sendSTUI {networkSpecificBoundSet} |
            serviceFilteringResponse {networkSpecificBoundSet} |
            splitLeg {networkSpecificBoundSet} |
            playAnnouncement {networkSpecificBoundSet} |
            promptAndCollectUserInformation {networkSpecificBoundSet} |
            scriptClose {networkSpecificBoundSet} |
            scriptEvent {networkSpecificBoundSet} |
            scriptInformation {networkSpecificBoundSet} |
            scriptRun {networkSpecificBoundSet} |
            specializedResourceReport |
            promptAndReceiveMessage {networkSpecificBoundSet}
            }
SsfScfGenericReturnable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet} |
            activityTest |
            applyCharging {networkSpecificBoundSet} |
            applyChargingReport {networkSpecificBoundSet} |
            callGap {networkSpecificBoundSet} |
            callInformationRequest {networkSpecificBoundSet} |
            cancel {networkSpecificBoundSet} |
            collectInformation {networkSpecificBoundSet} |
            connect {networkSpecificBoundSet} |
            connectToResource {networkSpecificBoundSet} |
            continue |
            continueWithArgument {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet}|
            establishTemporaryConnection {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            initialDP {networkSpecificBoundSet}|
            mergeCallSegments {networkSpecificBoundSet}|
            moveCallSegments {networkSpecificBoundSet}|
            moveLeg {networkSpecificBoundSet}|
            createCallSegmentAssociation {networkSpecificBoundSet}|
```

```
            releaseCall {networkSpecificBoundSet}|
            requestNotificationChargingEvent {networkSpecificBoundSet}|
            requestReportBCSMEvent {networkSpecificBoundSet}|
            requestReportUTSI {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            sendChargingInformation {networkSpecificBoundSet}|
            sendSTUI {networkSpecificBoundSet}|
            splitLeg {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            promptAndReceiveMessage {networkSpecificBoundSet}
            }

assistHandoff-ssf-scfAbstractSyntax ABSTRACT-SYNTAX:: = {
    AssistHandofor further studySF-SCF-PDUs
    IDENTIFIED BY  id-as-assistHandoff-ssf-scfAS}
AssistHandofor further studySF-SCF-PDUs:: = TCMessage {{AssistHandofor further
studysfToScfInvokable},
            {AssistHandofor further studysfToScfReturnable}}
AssistHandofor further studysfToScfInvokable OPERATION:: = {
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            assistRequestInstructions {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptEvent {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            specializedResourceReport |
            promptAndReceiveMessage {networkSpecificBoundSet}
            }
AssistHandofor further studysfToScfReturnable OPERATION:: = {
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            assistRequestInstructions {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            promptAndReceiveMessage {networkSpecificBoundSet}
            }

scf-ssfGenericAbstractSyntax ABSTRACT-SYNTAX:: = {
    GenericSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfGenericAS}
GenericSCF-SSF-PDUs:: = TCMessage {{ScfToSsfGenericInvokable}, {ScfToSsfGenericReturnable}}
ScfSsfGenericInvokable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet}|
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            callInformationRequest {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            collectInformation {networkSpecificBoundSet}|
            connect {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            continue |
            continueWithArgument{networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
```

```
            disconnectLeg {networkSpecificBoundSet}|
            establishTemporaryConnection {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            initiateCallAttempt {networkSpecificBoundSet}|
            mergeCallSegments {networkSpecificBoundSet}|
            moveCallSegments {networkSpecificBoundSet}|
            moveLeg {networkSpecificBoundSet}|
            createCallSegmentAssociation {networkSpecificBoundSet}|
            releaseCall {networkSpecificBoundSet}|
            requestNotificationChargingEvent {networkSpecificBoundSet}|
            requestReportBCSMEvent {networkSpecificBoundSet}|
            requestReportUTSI {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            sendChargingInformation {networkSpecificBoundSet}|
            sendSTUI {networkSpecificBoundSet}|
            splitLeg {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            promptAndReceiveMessage {networkSpecificBoundSet}
            }
ScfSsfGenericReturnable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet}|
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            callInformationReport {networkSpecificBoundSet}|
            callInformationRequest {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            collectInformation {networkSpecificBoundSet}|
            connect {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet}|
            entityReleased {networkSpecificBoundSet}|
            establishTemporaryConnection {networkSpecificBoundSet}|
            eventNotificationCharging {networkSpecificBoundSet} |
            resetTimer {networkSpecificBoundSet}|
            eventReportBCSM {networkSpecificBoundSet}|
            eventReportFacility {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            initiateCallAttempt {networkSpecificBoundSet}|
            mergeCallSegments {networkSpecificBoundSet}|
            moveCallSegments {networkSpecificBoundSet}|
            moveLeg {networkSpecificBoundSet}|
            createCallSegmentAssociation {networkSpecificBoundSet}|
            reportUTSI {networkSpecificBoundSet}|
            requestNotificationChargingEvent {networkSpecificBoundSet}|
            requestReportBCSMEvent {networkSpecificBoundSet}|
            requestReportUTSI {networkSpecificBoundSet} |
            sendChargingInformation {networkSpecificBoundSet}|
            sendSTUI {networkSpecificBoundSet}|
            serviceFilteringResponse {networkSpecificBoundSet}|
            splitLeg {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptEvent {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            specializedResourceReport |
            promptAndReceiveMessage {networkSpecificBoundSet}
            }



scf-ssfTrafficManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    TrafficManagementSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfTrafficManagementAS}
TrafficManagementSCF-SSF-PDUs:: = TCMessage {{ScfToSsfTrafficManagementInvokable}}
ScfToSsfTrafficManagementInvokable OPERATION:: = {
    callGap {networkSpecificBoundSet}
    }
```

```
scf-ssfServiceManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    ServiceManagementSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfServiceManagementAS}
ServiceManagementSCF-SSF-PDUs:: = TCMessage {{ScfToSsfServiceManagementInvokable},
           {ScfToSsfServiceManagementReturnable}}
ScfToSsfServiceManagementInvokable OPERATION:: = {
    activateServiceFiltering {networkSpecificBoundSet}
    }
ScfToSsfServiceManagementReturnable OPERATION:: = {
    activateServiceFiltering {networkSpecificBoundSet}
    }

ssf-scfServiceManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    ServiceManagementSSF-SCF-PDUs
    IDENTIFIED BY  id-as-ssf-scfServiceManagementAS}
ServiceManagementSSF-SCF-PDUs:: = TCMessage {{SsfToScfServiceManagementInvokable}}
SsfToScfServiceManagementInvokable OPERATION:: = {
    serviceFilteringResponse {networkSpecificBoundSet}
    }



scf-ssfTriggerManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    TriggerManagementSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfTriggerManagementAS}
TriggerManagementSCF-SSF-PDUs:: = TCMessage {{ScfToSsfTriggerManagementInvokable},
           {ScfToSsfTriggerManagementReturnable}}
ScfToSsfTriggerManagementInvokable OPERATION:: = {
    manageTriggerData
    }
ScfToSsfTriggerManagementReturnable OPERATION:: = {
    manageTriggerData
    }
```

6.2.1.3 SSF-SCF ACs

The **SSF to SCF Contracts** are realized by three ACs, **cs2ssf-scfGenericAC**, **cs2ssf-scfAssistHandoffAC** and **cs2ssf-scfServiceManagementAC**. These ACs are specified as information objects of the class APPLICATION-CONTEXT.

```
cs2ssf-scfGenericAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2SsfToScfGeneric
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
           ssf-scfGenericAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-ssf-scfGenericAC}

cs2ssf-scfAssistHandoffAC APPLICATION-CONTEXT:: =     {
    CONTRACT    inCs2AssistHandofor further studysfToScf
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
           assistHandoff-ssf-scfAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-ssf-scfAssistHandoffAC}

cs2ssf-scfServiceManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2SsfToScfServiceManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
           ssf-scfServiceManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-ssf-scfServiceManagementAC}
```

The **SCF to SSF Contracts** are realized by four ACs, **cs2scf-ssfGenericAC**, **cs2scf-ssfTrafficManagementAC**, **cs2scf-ssfServiceManagementAC and cs2scf-ssfTriggerManagementAC**. These ACs are specified as information objects of the class APPLICATION-CONTEXT.

```
cs2scf-ssfGenericAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfGeneric
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
           ssf-scfGenericAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfGenericAC}

cs2scf-ssfTrafficManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfTrafficManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
           scf-ssfTrafficManagementAbstractSyntax}
```

```
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfTrafficManagementAC}

cs2scf-ssfServiceManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfServiceManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES   {dialogue-abstract-syntax |
            scf-ssfServiceManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfServiceManagementAC}

cs2scf-ssfTriggerManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfTriggerManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES   {dialogue-abstract-syntax |
            scf-ssfTriggerManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfTriggerManagementAC}
```

6.2.2 SSF/SCF ASN.1 module

```
IN-CS2-SSF-SCF-pkgs-contracts-acs { ccitt(0) identified-organization(4) etsi(0) inDomain(1)
in-network(1) CS2(20) modules(0) in-cs2-ssf-scf-pkgs-contracts-acs (6) version1(0)}
DEFINITIONS:: =
BEGIN

-- This module describes the operation-packages, contracts and application-contexts used
-- over the SSF-SCF interface.

IMPORTS


    PARAMETERS-BOUND,
    networkSpecificBoundSet
FROM IN-CS2-classes classes


    ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, OPERATION
FROM Remote-Operations-Information-Objects ros-InformationObjects

    TCMessage {}
FROM TCAPMessages tc-Messages

    APPLICATION-CONTEXT, dialogue-abstract-syntax
FROM TC-Notation-Extensions tc-NotationExtensions

        activateServiceFiltering {},
        activityTest,
        applyCharging {},
        applyChargingReport {},
        assistRequestInstructions {},
        callGap {},
        callInformationReport {},
        callInformationRequest {},
        cancel {},
        collectInformation {},
        connect {},
        connectToResource {},
        continue,
        continueWithArgument {},
        createCallSegmentAssociation {},
        disconnectForwardConnection,
        disconnectForwardConnectionWithArgument {},
        disconnectLeg {},
        entityReleased {},
        establishTemporaryConnection {},
        eventNotificationCharging {},
        eventReportBCSM {},
        furnishChargingInformation {},
        initialDP {},
        initiateCallAttempt {},
        manageTriggerData {},
        mergeCallSegments {},
        moveCallSegments {},
        moveLeg {},
        releaseCall {},
        reportUTSI {},
        requestNotificationChargingEvent {},
        requestReportBCSMEvent {},
        requestReportUTSI {},
```

```
        resetTimer {},
        sendChargingInformation {},
        sendSTUI {},
        serviceFilteringResponse {},
        splitLeg {}
FROM IN-CS2-SSF-SCF-ops-args ssf-scf-Operations

        playAnnouncement {},
        promptAndCollectUserInformation {},
        promptAndReceiveMessage {},
        scriptClose {},
        scriptEvent {},
        scriptInformation {},
        scriptRun {},
        specializedResourceReport
FROM IN-CS2-SCF-SRF-ops-args scf-srf-Operations

    specializedResourceControlPackage {},
    scriptControlPackage {},
    messageControlPackage {}
FROM IN-CS2-SCF-SRF-pkgs-contracts-acs scf-srf-Protocol

    id-ac-cs2-ssf-scfGenericAC,
    id-ac-cs2-ssf-scfAssistHandoffAC,
    id-ac-cs2-ssf-scfServiceManagementAC,
    id-ac-cs2-scf-ssfGenericAC,
    id-ac-cs2-scf-ssfTrafficManagementAC,
    id-ac-cs2-scf-ssfServiceManagementAC,
    id-ac-cs2-scf-ssfTriggerManagementAC,
    id-inCs2SsfToScfGeneric,
    id-inCs2AssistHandofor further studysfToScf,
    id-inCs2ScfToSsfGeneric,
    id-inCs2ScfToSsfTrafficManagement,
    id-inCs2ScfToSsfServiceManagement,
    id-inCs2SsfToScfServiceManagement,
    id-inCs2ScfToSsfTriggerManagement,
    id-as-ssf-scfGenericAS,
    id-as-assistHandoff-ssf-scfAS,
    id-as-scf-ssfGenericAS,
    id-as-scf-ssfTrafficManagementAS,
    id-as-scf-ssfServiceManagementAS,
    id-as-ssf-scfServiceManagementAS,
    id-as-scf-ssfTriggerManagementAS,
    id-package-scfActivation,
    id-package-srf-scfActivationOfAssist,
    id-package-assistConnectionEstablishment,
    id-package-genericDisconnectResource,
    id-package-nonAssistedConnectionEstablishment,
    id-package-connect,
    id-package-callHandling,
    id-package-bcsmEventHandling,
    id-package-chargingEventHandling,
    id-package-ssfCallProcessing,
    id-package-scfCallInitiation,
    id-package-timer,
    id-package-billing,
    id-package-charging,
    id-package-trafficManagement,
    id-package-serviceManagementActivate,
    id-package-serviceManagementResponse,
    id-package-callReport,
    id-package-signallingControl,
    id-package-activityTest,
    id-package-cancel,
    id-package-cphResponse,
    id-package-entityReleased,
    id-package-triggerManagement,
    id-package-uSIHandling,
    classes, ros-InformationObjects, tc-Messages, tc-NotationExtensions,
    ssf-scf-Operations, scf-srf-Operations, scf-srf-Protocol
FROM IN-CS2-object-identifiers { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) CS2(20) modules(0) in-cs2-object-identifiers (17) version1(0)}

;
-- Application Contexts

cs2ssf-scfGenericAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2SsfToScfGeneric
    DIALOGUE MODE   structured
```

```
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            ssf-scfGenericAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-ssf-scfGenericAC}

cs2ssf-scfAssistHandoffAC APPLICATION-CONTEXT:: =     {
    CONTRACT    inCs2AssistHandofor further studysfToScf
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            assistHandoff-ssf-scfAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-ssf-scfAssistHandoffAC}

cs2ssf-scfServiceManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2SsfToScfServiceManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            ssf-scfServiceManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-ssf-scfServiceManagementAC}
cs2scf-ssfGenericAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfGeneric
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            ssf-scfGenericAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfGenericAC}

cs2scf-ssfTrafficManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfTrafficManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            scf-ssfTrafficManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfTrafficManagementAC}

cs2scf-ssfServiceManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfServiceManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            scf-ssfServiceManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfServiceManagementAC}

cs2scf-ssfTriggerManagementAC APPLICATION-CONTEXT:: = {
    CONTRACT    inCs2ScfToSsfTriggerManagement
    DIALOGUE MODE   structured
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            scf-ssfTriggerManagementAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-cs2-scf-ssfTriggerManagementAC}


-- Contracts
inCs2SsfToScfGeneric CONTRACT:: = {
-- dialogue initiated by SSF with InitialDP Operation
    INITIATOR CONSUMER OF  {exceptionInformPackage {networkSpecificBoundSet} |
            scfActivationPackage {networkSpecificBoundSet} }
    RESPONDER CONSUMER OF  {activityTestPackage|
            assistConnectionEstablishmentPackage {networkSpecificBoundSet} |
            bcsmEventHandlingPackage {networkSpecificBoundSet} |
            billingPackage {networkSpecificBoundSet} |
            callHandlingPackage {networkSpecificBoundSet} |
            callReportPackage {networkSpecificBoundSet} |
            cancelPackage {networkSpecificBoundSet} |
            chargingEventHandlingPackage {networkSpecificBoundSet} |
            chargingPackage {networkSpecificBoundSet} |
            connectPackage {networkSpecificBoundSet} |
            cphResponsePackage {networkSpecificBoundSet} |
            genericDisconnectResourcePackage {networkSpecificBoundSet} |
            nonAssistedConnectionEstablishmentPackage {networkSpecificBoundSet} |
            scfCallInitiationPackage {networkSpecificBoundSet} |
            signallingControlPackage {networkSpecificBoundSet} |
            specializedResourceControlPackage {networkSpecificBoundSet} |
            scriptControlPackage {networkSpecificBoundSet} |
            messageControlPackage {networkSpecificBoundSet} |
            ssfCallProcessingPackage {networkSpecificBoundSet} |
            timerPackage {networkSpecificBoundSet} |
            trafficManagementPackage {networkSpecificBoundSet} |
            uSIHandlingPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2SsfToScfGeneric
    }

inCs2AssistHandofor further studysfToScf CONTRACT:: = {
-- dialogue initiated by SSF with AssistRequestInstructions
```

```
    INITIATOR CONSUMER OF  {srf-scfActivationOfAssistPackage {networkSpecificBoundSet} }
    RESPONDER CONSUMER OF  {activityTestPackage|
            billingPackage {networkSpecificBoundSet} |
            callHandlingPackage {networkSpecificBoundSet} |
            cancelPackage {networkSpecificBoundSet} |
            chargingPackage {networkSpecificBoundSet} |
            genericDisconnectResourcePackage {networkSpecificBoundSet} |
            nonAssistedConnectionEstablishmentPackage {networkSpecificBoundSet} |
            specializedResourceControlPackage {networkSpecificBoundSet} |
            scriptControlPackage {networkSpecificBoundSet} |
            messageControlPackage {networkSpecificBoundSet} |
            timerPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2AssistHandofor further studysfToScf
    }

inCs2ScfToSsfGeneric CONTRACT:: = {
-- dialogue initiated by SCF with InitiateCallAttempt, Generic Case
    INITIATOR CONSUMER OF  {activityTestPackage|
            assistConnectionEstablishmentPackage {networkSpecificBoundSet} |
            bcsmEventHandlingPackage {networkSpecificBoundSet} |
            billingPackage {networkSpecificBoundSet} |
            callHandlingPackage {networkSpecificBoundSet} |
            callReportPackage {networkSpecificBoundSet} |
            cancelPackage {networkSpecificBoundSet} |
            chargingPackage {networkSpecificBoundSet} |
            connectPackage {networkSpecificBoundSet} |
            cphResponsePackage {networkSpecificBoundSet} |
            genericDisconnectResourcePackage {networkSpecificBoundSet} |
            nonAssistedConnectionEstablishmentPackage {networkSpecificBoundSet} |
            scfCallInitiationPackage {networkSpecificBoundSet} |
            signallingControlPackage {networkSpecificBoundSet} |
            specializedResourceControlPackage {networkSpecificBoundSet} |
            scriptControlPackage {networkSpecificBoundSet} |
            messageControlPackage {networkSpecificBoundSet} |
            ssfCallProcessingPackage {networkSpecificBoundSet} |
            timerPackage {networkSpecificBoundSet} |
            uSIHandlingPackage {networkSpecificBoundSet}
            }
    RESPONDER CONSUMER OF  {exceptionInformPackage {networkSpecificBoundSet} }
    ID      id-inCs2ScfToSsfGeneric
    }
inCs2ScfToSsfTrafficManagement CONTRACT:: = {
-- dialogue initiated by SCF with CallGap
    INITIATOR CONSUMER OF  {trafficManagementPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2ScfToSsfTrafficManagement
    }
inCs2ScfToSsfServiceManagement CONTRACT:: = {
-- dialogue initiated by SCF with ActivateServiceFiltering
    INITIATOR CONSUMER OF  {serviceManagementActivatePackage {networkSpecificBoundSet}
            }
    ID      id-inCs2ScfToSsfServiceManagement
    }
inCs2SsfToScfServiceManagement CONTRACT:: = {
-- dialogue initiated/ended by SSF with ServiceFilteringResponse
    INITIATOR CONSUMER OF  {serviceManagementResponsePackage {networkSpecificBoundSet}
            }
    ID      id-inCs2SsfToScfServiceManagement
    }
inCs2ScfToSsfTriggerManagement CONTRACT:: = {
-- dialogue initiated by SCF with Manage Trigger Data
    INITIATOR CONSUMER OF  {triggerManagementPackage {networkSpecificBoundSet}
            }
    ID      id-inCs2ScfToSsfTriggerManagement
    }

-- Operation Packages

scfActivationPackage {PARAMETERS-BOUND: bound}  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {initialDP {bound}}
    ID      id-package-scfActivation}
srf-scfActivationOfAssistPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {assistRequestInstructions {bound}}
    ID      id-package-srf-scfActivationOfAssist}
assistConnectionEstablishmentPackage {PARAMETERS-BOUND: bound}   OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {establishTemporaryConnection {bound}}
    ID      id-package-assistConnectionEstablishment}
```

```
genericDisconnectResourcePackage {PARAMETERS-BOUND: bound}   OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {bound}}
    ID      id-package-genericDisconnectResource}
nonAssistedConnectionEstablishmentPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {connectToResource {bound}}
    ID      id-package-nonAssistedConnectionEstablishment}
connectPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {connect {bound}}
    ID      id-package-connect}
callHandlingPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {releaseCall {bound}}
    ID      id-package-callHandling}
bcsmEventHandlingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestReportBCSMEvent {bound}}
    SUPPLIER INVOKES   {eventReportBCSM {bound}}
    ID      id-package-bcsmEventHandling}
chargingEventHandlingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestNotificationChargingEvent {bound}}
    SUPPLIER INVOKES   {eventNotificationCharging {bound}}
    ID      id-package-chargingEventHandling}
ssfCallProcessingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {collectInformation {bound} | continue}
    ID      id-package-ssfCallProcessing}
scfCallInitiationPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {initiateCallAttempt {bound}}
    ID      id-package-scfCallInitiation}
timerPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {resetTimer {bound}}
    ID      id-package-timer}
billingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {furnishChargingInformation {bound}}
    ID      id-package-billing}
chargingPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {applyCharging {bound}}
    SUPPLIER INVOKES   {applyChargingReport {bound}}
    ID      id-package-charging}
trafficManagementPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {callGap {bound}}
    ID      id-package-trafficManagement}
serviceManagementActivatePackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {activateServiceFiltering {bound}}
    ID      id-package-serviceManagementActivate}
serviceManagementResponsePackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {serviceFilteringResponse {bound}}
    ID      id-package-serviceManagementResponse}
callReportPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {callInformationRequest {bound}}
    SUPPLIER INVOKES   {callInformationReport {bound}}
    ID      id-package-callReport}
signallingControlPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {sendChargingInformation {bound}}
    ID      id-package-signallingControl}
activityTestPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {activityTest}
    ID      id-package-activityTest}
cancelPackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {cancel {bound} }
    ID      id-package-cancel}
cphResponsePackage {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {
            continueWithArgument {bound}|
            disconnectLeg {bound}|
            mergeCallSegments {bound}|
            moveCallSegments {bound}|
            moveLeg {bound}|
            createCallSegmentAssociation {bound} |
            splitLeg {bound}
            }
    ID      id-package-cphResponse}
exceptionInformPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {entityReleased {bound}}
    ID      id-package-entityReleased}
triggerManagementPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {manageTriggerData {bound}}
    ID      id-package-triggerManagement}
```

```
uSIHandlingPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestReportUTSI {bound}| sendSTUI {bound}}
    SUPPLIER INVOKES   {reportUTSI {bound}}
    ID      id-package-uSIHandling
    }



-- Abstract Syntaxes

ssf-scfGenericAbstractSyntax ABSTRACT-SYNTAX:: = {
    GenericSSF-SCF-PDUs
    IDENTIFIED BY  id-as-ssf-scfGenericAS}
GenericSSF-SCF-PDUs:: = TCMessage {{SsfToScfGenericInvokable},
            {SsfToScfGenericReturnable}}
SsfScfGenericInvokable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet} |
            activityTest |
            applyCharging {networkSpecificBoundSet} |
            applyChargingReport {networkSpecificBoundSet} |
            callInformationReport {networkSpecificBoundSet} |
            callInformationRequest {networkSpecificBoundSet} |
            cancel {networkSpecificBoundSet} |
            cancelStatusReportRequest {networkSpecificBoundSet} |
            collectInformation {networkSpecificBoundSet} |
            connect {networkSpecificBoundSet} |
            connectToResource {networkSpecificBoundSet} |
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet} |
            entityReleased {networkSpecificBoundSet} |
            establishTemporaryConnection {networkSpecificBoundSet} |
            eventNotificationCharging {networkSpecificBoundSet} |
            eventReportBCSM {networkSpecificBoundSet} |
            furnishChargingInformation {networkSpecificBoundSet} |
            initialDP {networkSpecificBoundSet} |
            mergeCallSegments {networkSpecificBoundSet} |
            moveCallSegments {networkSpecificBoundSet} |
            moveLeg {networkSpecificBoundSet} |
            createCallSegmentAssociation {networkSpecificBoundSet} |
            releaseCall {networkSpecificBoundSet} |
            reportUTSI {networkSpecificBoundSet} |
            requestNotificationChargingEvent {networkSpecificBoundSet} |
            requestReportBCSMEvent {networkSpecificBoundSet} |
            requestReportFacilityEvent {networkSpecificBoundSet} |
            requestReportUTSI {networkSpecificBoundSet} |
            resetTimer {networkSpecificBoundSet} |
            sendChargingInformation {networkSpecificBoundSet} |
            sendSTUI {networkSpecificBoundSet} |
            serviceFilteringResponse {networkSpecificBoundSet} |
            splitLeg {networkSpecificBoundSet} |
            playAnnouncement {networkSpecificBoundSet} |
            promptAndCollectUserInformation {networkSpecificBoundSet} |
            scriptClose {networkSpecificBoundSet} |
            scriptEvent {networkSpecificBoundSet} |
            scriptInformation {networkSpecificBoundSet} |
            scriptRun {networkSpecificBoundSet} |
            specializedResourceReport |
            promptAndReceiveMessage {networkSpecificBoundSet}
            }
SsfScfGenericReturnable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet} |
            activityTest |
            applyCharging {networkSpecificBoundSet} |
            applyChargingReport {networkSpecificBoundSet} |
            callGap {networkSpecificBoundSet} |
            callInformationRequest {networkSpecificBoundSet} |
            cancel {networkSpecificBoundSet} |
            collectInformation {networkSpecificBoundSet} |
            connect {networkSpecificBoundSet} |
            connectToResource {networkSpecificBoundSet} |
            continue |
            continueWithArgument {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet}|
            establishTemporaryConnection {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            initialDP {networkSpecificBoundSet}|
```

```
            mergeCallSegments {networkSpecificBoundSet}|
            moveCallSegments {networkSpecificBoundSet}|
            moveLeg {networkSpecificBoundSet}|
            createCallSegmentAssociation {networkSpecificBoundSet}|
            releaseCall {networkSpecificBoundSet}|
            requestNotificationChargingEvent {networkSpecificBoundSet}|
            requestReportBCSMEvent {networkSpecificBoundSet}|
            requestReportUTSI {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            sendChargingInformation {networkSpecificBoundSet}|
            sendSTUI {networkSpecificBoundSet}|
            splitLeg {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            promptAndReceiveMessage {networkSpecificBoundSet}
            }

assistHandoff-ssf-scfAbstractSyntax ABSTRACT-SYNTAX:: = {
    AssistHandofor further studySF-SCF-PDUs
    IDENTIFIED BY  id-as-assistHandoff-ssf-scfAS}
AssistHandofor further studySF-SCF-PDUs:: = TCMessage {{AssistHandofor further
studysfToScfInvokable},
            {AssistHandofor further studysfToScfReturnable}}
AssistHandofor further studysfToScfInvokable OPERATION:: = {
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            assistRequestInstructions {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            holdCallInNetwork |
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptEvent {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            specializedResourceReport |
            promptAndReceiveMessage {networkSpecificBoundSet}
            }
AssistHandofor further studysfToScfReturnable OPERATION:: = {
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            assistRequestInstructions {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            promptAndReceiveMessage {networkSpecificBoundSet}
            }


scf-ssfGenericAbstractSyntax ABSTRACT-SYNTAX:: = {
    GenericSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfGenericAS}
GenericSCF-SSF-PDUs:: = TCMessage {{ScfToSsfGenericInvokable}, {ScfToSsfGenericReturnable}}
ScfSsfGenericInvokable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet}|
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            callInformationRequest {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
```

```
            collectInformation {networkSpecificBoundSet}|
            connect {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            continue |
            continueWithArgument{networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet}|
            establishTemporaryConnection {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            initiateCallAttempt {networkSpecificBoundSet}|
            mergeCallSegments {networkSpecificBoundSet}|
            moveCallSegments {networkSpecificBoundSet}|
            moveLeg {networkSpecificBoundSet}|
            createCallSegmentAssociation {networkSpecificBoundSet}|
            releaseCall {networkSpecificBoundSet}|
            requestNotificationChargingEvent {networkSpecificBoundSet}|
            requestReportBCSMEvent {networkSpecificBoundSet}|
            requestReportUTSI {networkSpecificBoundSet}|
            resetTimer {networkSpecificBoundSet}|
            sendChargingInformation {networkSpecificBoundSet}|
            sendSTUI {networkSpecificBoundSet}|
            splitLeg {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            promptAndReceiveMessage {networkSpecificBoundSet}
            }
ScfSsfGenericReturnable OPERATION:: = {
            activateServiceFiltering {networkSpecificBoundSet}|
            activityTest |
            applyCharging {networkSpecificBoundSet}|
            applyChargingReport {networkSpecificBoundSet}|
            callInformationReport {networkSpecificBoundSet}|
            callInformationRequest {networkSpecificBoundSet}|
            cancel {networkSpecificBoundSet}|
            collectInformation {networkSpecificBoundSet}|
            connect {networkSpecificBoundSet}|
            connectToResource {networkSpecificBoundSet}|
            disconnectForwardConnection |
            disconnectForwardConnectionWithArgument {networkSpecificBoundSet}|
            disconnectLeg {networkSpecificBoundSet}|
            entityReleased {networkSpecificBoundSet}|
            establishTemporaryConnection {networkSpecificBoundSet}|
            eventNotificationCharging {networkSpecificBoundSet} | resetTimer
{networkSpecificBoundSet}|
            eventReportBCSM {networkSpecificBoundSet}|
            furnishChargingInformation {networkSpecificBoundSet}|
            holdCallInNetwork |
            initiateCallAttempt {networkSpecificBoundSet}|
            mergeCallSegments {networkSpecificBoundSet}|
            moveCallSegments {networkSpecificBoundSet}|
            moveLeg {networkSpecificBoundSet}|
            createCallSegmentAssociation {networkSpecificBoundSet}|
            reportUTSI {networkSpecificBoundSet}|
            requestNotificationChargingEvent {networkSpecificBoundSet}|
            requestReportBCSMEvent {networkSpecificBoundSet}|
            requestReportUTSI {networkSpecificBoundSet} |
            sendChargingInformation {networkSpecificBoundSet}|
            sendSTUI {networkSpecificBoundSet}|
            serviceFilteringResponse {networkSpecificBoundSet}|
            splitLeg {networkSpecificBoundSet}|
            statusReport {networkSpecificBoundSet}|
            playAnnouncement {networkSpecificBoundSet}|
            promptAndCollectUserInformation {networkSpecificBoundSet}|
            scriptClose {networkSpecificBoundSet}|
            scriptEvent {networkSpecificBoundSet}|
            scriptInformation {networkSpecificBoundSet}|
            scriptRun {networkSpecificBoundSet}|
            specializedResourceReport |
            promptAndReceiveMessage {networkSpecificBoundSet}
            }


scf-ssfTrafficManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    TrafficManagementSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfTrafficManagementAS}
```

```
TrafficManagementSCF-SSF-PDUs:: = TCMessage {{ScfToSsfTrafficManagementInvokable}}
ScfToSsfTrafficManagementInvokable OPERATION:: = {
    callGap {networkSpecificBoundSet}
    }


scf-ssfServiceManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    ServiceManagementSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfServiceManagementAS}
ServiceManagementSCF-SSF-PDUs:: = TCMessage {{ScfToSsfServiceManagementInvokable},
           {ScfToSsfServiceManagementReturnable}}
ScfToSsfServiceManagementInvokable OPERATION:: = {
    activateServiceFiltering {networkSpecificBoundSet}
    }
ScfToSsfServiceManagementReturnable OPERATION:: = {
    activateServiceFiltering {networkSpecificBoundSet}
    }

ssf-scfServiceManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    ServiceManagementSSF-SCF-PDUs
    IDENTIFIED BY  id-as-ssf-scfServiceManagementAS}
ServiceManagementSSF-SCF-PDUs:: = TCMessage {{SsfToScfServiceManagementInvokable}}
SsfToScfServiceManagementInvokable OPERATION:: = {
    serviceFilteringResponse {networkSpecificBoundSet}
    }


scf-ssfTriggerManagementAbstractSyntax ABSTRACT-SYNTAX:: = {
    TriggerManagementSCF-SSF-PDUs
    IDENTIFIED BY  id-as-scf-ssfTriggerManagementAS}
TriggerManagementSCF-SSF-PDUs:: = TCMessage {{ScfToSsfTriggerManagementInvokable},
           {ScfToSsfTriggerManagementReturnable}}
ScfToSsfTriggerManagementInvokable OPERATION:: = {
    manageTriggerData {networkSpecificBoundSet}
    }
ScfToSsfTriggerManagementReturnable OPERATION:: = {
    manageTriggerData {networkSpecificBoundSet}
    }

END
```

7 SCF/SRF interface

7.1 SCF/SRF operations and arguments

```
IN-CS2-SCF-SRF-ops-args {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-srf-ops-args
(7) version1(0)}

DEFINITIONS IMPLICIT TAGS:: =

BEGIN

IMPORTS

    OPERATION
FROM Remote-Operations-Information-Objects ros-InformationObjects

    opcode-eraseMessage,
    opcode-playAnnouncement,
    opcode-promptAndCollectUserInformation,
    opcode-promptAndReceiveMessage,
    opcode-scriptClose,
    opcode-scriptEvent,
    opcode-scriptInformation,
    opcode-scriptRun,
    opcode-specializedResourceReport

FROM IN-CS2-operationcodes operationcodes

    CallSegmentID {},
    CollectedInfo,
    Digits {},
    ExtensionField {},
```

```
    InformationToRecord {},
    InformationToSend {},
    LegID,
    MailBoxID {},
    Media,
    GenericNumber {},
    ReceivedStatus,
    RecordedMessageID

FROM IN-CS2-datatypes datatypes

    canceled,
    improperCallerResponse,
    missingParameter,
    parameterOutOfRange,
    systemFailure,
    taskRefused,
    unavailableResource,
    unexpectedComponentSequence,
    unexpectedDataValue,
    unexpectedParameter,
    unknownRecordedMessageID,
    unknownSubscriber

FROM IN-CS2-errortypes errortypes

    UISCRIPT,
    SupportedUIScripts {},
    PARAMETERS-BOUND

FROM IN-CS2-classes classes

    ros-InformationObjects, operationcodes, datatypes, errortypes, classes

FROM IN-CS2-object-identifiers
{ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-identifiers(17) version1(0)}
;

eraseMessage {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    EraseMessageArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unavailableResource |
            unexpectedComponentSequence |
            unexpectedDataValue|
            unknownRecordedMessageID
            }
    CODE    opcode-eraseMessage
    }
-- Direction: SCF ->SRF, Timer: Tem

EraseMessageArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    disconnectFromIPForbidden  [0] BOOLEAN DEFAULT TRUE,
    recordedMessageIDsToErase  [1] SEQUENCE
                 SIZE(1..bound.&numOfRecordedMessageIDs) OF
                RecordedMessageID,
    callSegmentID   [2] CallSegmentID {bound}  OPTIONAL,
    extensions  [3] SEQUENCE SIZE(0..bound.&numOfExtensions)  OF
                 ExtensionField {bound}    OPTIONAL,
    ...
    }

playAnnouncement {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    PlayAnnouncementArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {canceled |
            missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter |
            unavailableResource
            }
    LINKED  {specializedResourceReport}
```

```
    CODE    opcode-playAnnouncement
    }
-- Direction: SCF -> SRF, Timer: Tpa
-- This operation is to be used after Establish Temporary Connection (assist procedure with a
second SSP)
-- or a Connect to Resource (no assist) operation. It may be used for inband interaction with an
analog user,
-- or for interaction with an ISDN user. In the former case, the SRF is usually collocated with
the SSF for
-- standard tones (congestion tone...) or standard announcements. In the latter case, the SRF is
always
-- collocated with the SSF in the switch. Any error is returned to the SCF. The timer associated
with this
-- operation must be of a sufficient duration to allow its linked operation to be correctly
correlated.

PlayAnnouncementArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    informationToSend  [0] InformationToSend {bound},
    disconnectFromIPForbidden  [1] BOOLEAN DEFAULT TRUE,
    requestAnnouncementComplete    [2] BOOLEAN DEFAULT TRUE,
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField  {bound}   OPTIONAL,
    connectedParty  CHOICE {
            legID   [4] LegID,
            callSegmentID   [5] CallSegmentID {bound}
            }   OPTIONAL,
    ...
    }

promptAndCollectUserInformation {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    PromptAndCollectUserInformationArg { bound}
    RESULT  ReceivedInformationArg { bound}
    ERRORS  {canceled |
            improperCallerResponse |
            missingParameter |
            parameterOutOfRange |
            systemFailure |
            taskRefused |
            unexpectedComponentSequence |
            unavailableResource |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-promptAndCollectUserInformation
    }
-- Direction: SCF -> SRF, Timer: Tpc
-- This operation is used to interact with a user to collect information.

PromptAndCollectUserInformationArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    collectedInfo   [0] CollectedInfo,
    disconnectFromIPForbidden  [1] BOOLEAN DEFAULT TRUE,
    informationToSend  [2] InformationToSend {bound}  OPTIONAL,
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField  {bound}   OPTIONAL,
    callSegmentID   [4] CallSegmentID {bound}  OPTIONAL,
    ...
    }

ReceivedInformationArg {PARAMETERS-BOUND: bound}:: = CHOICE {
    digitsResponse  [0] Digits {bound},
    iA5Response [1] IA5String
    }

promptAndReceiveMessage {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    PromptAndReceiveMessageArg { bound}
    RESULT  MessageReceivedArg { bound}
    ERRORS  {canceled |
            improperCallerResponse |
            missingParameter |
            parameterOutOfRange |
            taskRefused |
            systemFailure |
            unavailableResource |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter |
            unknownSubscriber
            }
```

```
    CODE    opcode-promptAndReceiveMessage
    }
--  Direction: SCF ->SRF, Timer: Tprm
--  Used to prompt a user to store a message

PromptAndReceiveMessageArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    disconnectFromIPForbidden  [0] BOOLEAN DEFAULT TRUE,
    informationToSend  [1] InformationToSend {bound}  OPTIONAL,
    extensions  [3] SEQUENCE SIZE(0..bound.&numOfExtensions)  OF
                ExtensionField {bound}    OPTIONAL,
    subscriberID    [4] GenericNumber {bound}  OPTIONAL,
    mailBoxID   [5] MailBoxID {bound}  OPTIONAL,
    informationToRecord[6] InformationToRecord {bound},
    media   [7] Media   DEFAULT voiceMail,
    callSegmentID   [8] CallSegmentID {bound}  OPTIONAL,
    ...
    }

MessageReceivedArg {PARAMETERS-BOUND: bound}  :: = SEQUENCE {
    receivedStatus  [0] ReceivedStatus,
    recordedMessageID  [1] RecordedMessageID  OPTIONAL,
    recordedMessageUnits   [2] INTEGER(1..bound.&maxRecordedMessageUnits)    OPTIONAL,
    extensions  [3] SEQUENCE SIZE(1..bound.&numOfExtensions)  OF
                ExtensionField {bound}    OPTIONAL,
    ...
    }

scriptClose {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ScriptCloseArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {
            systemFailure |
            missingParameter |
            taskRefused |
            unavailableResource |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-scriptClose
    }
-- Direction: SCF-> SRF. Timer: T_Cl
-- This operation is issued by the SCF to deallocate the resources used to perform the
-- instance of the "User Interaction" script: the context is released.

ScriptCloseArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    uIScriptId  UISCRIPT.&id({SupportedUIScripts { bound}}),
    uIScriptSpecificInfo   [0] UISCRIPT.&SpecificInfo({SupportedUIScripts {bound}}{@uIScriptId})
                OPTIONAL,
    extensions  [1] SEQUENCE SIZE (1..bound.&numOfExtensions)
                OF ExtensionField {bound} OPTIONAL,
    callSegmentID   [2] CallSegmentID {bound}  OPTIONAL,
    ...
    }

scriptEvent {PARAMETERS-BOUND: bound} OPERATION:: = {
    ARGUMENT    ScriptEventArg { bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-scriptEvent
    }
-- Direction: SRF-> SCF. Timer: T_Re
-- This operation is issued by the SRF to return information to the SCF on the results of the
-- execution of the instance of User Interaction script.

ScriptEventArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    uIScriptId  UISCRIPT.&id({SupportedUIScripts { bound}}),
    uIScriptResult  [0] UISCRIPT.&Result({SupportedUIScripts {bound}}{@uIScriptId})
                OPTIONAL,
    extensions  [1] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
                ExtensionField {bound}    OPTIONAL,
    callSegmentID   [2] CallSegmentID {bound}  OPTIONAL,
    lastEventIndicator [3] BOOLEAN    DEFAULT  FALSE,
    ...
    }

scriptInformation {PARAMETERS-BOUND: bound} OPERATION:: ={
    ARGUMENT    ScriptInformationArg { bound}
    RETURN RESULT   FALSE
```

```
    ERRORS  {
            systemFailure |
            missingParameter |
            taskRefused |
            unavailableResource |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-scriptInformation
    }
-- Direction: SCF-> SRF. Timer: Tinf

ScriptInformationArg {PARAMETERS-BOUND: bound }:: = SEQUENCE {
    uIScriptId  UISCRIPT.&id({SupportedUIScripts { bound}}),
    uIScriptSpecificInfo   [0] UISCRIPT.&SpecificInfo({SupportedUIScripts {
bound}}{@uIScriptId})
                OPTIONAL,
    extensions  [1] SEQUENCE SIZE(0..bound.&numOfExtensions)  OF
                 ExtensionField {bound}    OPTIONAL,
    callSegmentID   [2] CallSegmentID {bound}  OPTIONAL,
    ...
    }

scriptRun {PARAMETERS-BOUND: bound} OPERATION:: ={
    ARGUMENT    ScriptRunArg { bound}
    RETURN RESULT   FALSE
    ERRORS  {

            systemFailure |
            missingParameter |
            taskRefused |
            unavailableResource |
            unexpectedComponentSequence |
            unexpectedDataValue |
            unexpectedParameter
            }
    CODE    opcode-scriptRun
    }
-- Direction: SCF-> SRF. Timer: Tru
-- This operation is issued by the SCF to allocate the necessary resources to perform the
-- instance of the "User Interaction" script and then to activate this "User Interaction" script
-- instance. A context is partially defined for it if necessary.

ScriptRunArg {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    uIScriptId  UISCRIPT.&id({SupportedUIScripts { bound}}),
    uIScriptSpecificInfo   [0] UISCRIPT.&SpecificInfo({SupportedUIScripts {
bound}}{@uIScriptId})
                OPTIONAL,
    extensions  [1] SEQUENCE SIZE (1..bound.&numOfExtensions) OF
                 ExtensionField {bound}    OPTIONAL,
    disconnectFromIPForbidden  [2] BOOLEAN DEFAULT TRUE,
    callSegmentID   [3] CallSegmentID {bound}  OPTIONAL,
    ...
    }

specializedResourceReport OPERATION:: = {
    ARGUMENT    SpecializedResourceReportArg
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-specializedResourceReport
    }
-- Direction: SRF ->  SCF, Timer: Tsrr
-- This operation is used as  the response to a PlayAnnouncement operation when the announcement
completed
-- report indication is set.

SpecializedResourceReportArg:: =   NULL

END
```

The following value ranges do apply for operation specific timers in INAP:

short:	1 - 10 s
medium:	1 - 60 s
long:	1 s - 30 minutes
for further study:	For Further Study

Table 6-1 lists all operation timers and the value range for each timer. The definitive value for each operation timer may be network specific and has to be defined by the network operator.

Table 6-1: Operation timers and their value range

Operation Name	Timer	value range
ScriptClose	T_{cl}	short
EraseMessage	T_{em}	short
ScriptInformation	T_{inf}	short
PlayAnnouncement	T_{pa}	long
PromptAndCollectUserInformation	T_{pc}	long
PromptAndReceiveMessage	T_{prm}	long
ScriptEvent	T_{re}	short
ScriptRun	T_{ru}	long
SpecializedResourceReport	T_{srr}	short

7.2 SRF/SCF contracts, packages and ACs

7.2.1 Protocol overview

The **srf-scfContract** expresses the form of the service in which the SRF, a ROS-object of class **srf-scf**, initiates the contract. A ROS-object of class **scf-srf** responds in this contract.

The **srf-scfContract** is composed of a connection package, **emptyConnectionPackage** and operation packages**: specializedResourceControlPackage, srf-scfCancelPackage,srf-scfActivationOfAssistPackage scriptControlPackage, and messageControlPackage**. The connection package, **emptyConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE in subclause 4.5.

When an SCF and an SRF are located in different IN PE, these association contracts shall be realized as an SS7 application layer protocol. The definition of this protocol in terms of an SS7 AC is provided in subclause 6.2.2 of this ITU-T Recommendation.

The operation package **specializedResourceControlPackage** is defined as an information object of class OPERATION-PACKAGE. The operations of this package is defined in subclause 6.1.

The operation package srf-scfrCancelPackage is defined as information object of class OPERATION-PACKAGE. The operation of this package is defined in subclause 5.1.

The operation package scriptControlPackage is defined as information object of class OPERATION-PACKAGE. The operations of this package are defined in subclause 6.1.

The operation package messageControlPackage is defined as an information object of class OPERATION-PACKAGE. The operations of this package are defined in section 6.1

```
messageControlPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {eraseMessage | promptAndReceiveMessage}
    ID      id-package-messageControl
    }
```

Abstract Syntax

This version of the INAP requires the support of two types of abstract syntaxes:

a) the abstract syntax of TC dialogue control PDUs, **dialogue-abstract-syntax,** which is needed to establish the dialogue between FEs and specified in ETS 300 287-1 [7];

b) the abstract syntax for conveying the PDUs for invoking the operations involved in the operation packages specified as above and reporting their outcome.

The ASN.1 type from which the values of the last abstract syntax are derived is specified using the parameterized types **TCMessages{}** defined in ETS 300 287-1 [7].

All these abstract syntaxes shall (as a minimum) be encoded according to the Basic ASN.1 encoding rules with the restrictions listed in ETS 300 287-1 [7].

The SRF-SCF INAP ASEs that realize the operation packages specified as above and the empty connection package specified in subclause 4.5 share a single abstract syntax, srf-scf-abstract-syntax. This is specified as an information object of the class ABSTRACT-SYNTAX.

SCF-SRF ACs

The srf-scfContract is realized by an ACs, srf-scf-ac. These ACs are specified as information objects of the class APPLICATION-CONTEXT.

7.2.2 SRF/SCF ASN.1 modules

```
IN-CS2-SCF-SRF-pkgs-contracts-acs {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-srf-
pkgs-contracts-acs(8) version1(0)}

DEFINITIONS:: =

BEGIN

-- This module describes the operation-packages, contracts and application-contexts used
--  over the SCF-SRF interface.

IMPORTS

    PARAMETERS-BOUND,
    networkSpecificBoundSet,
    emptyConnectionPackage

FROM IN-CS2-classes classes


    ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, OPERATION

FROM Remote-Operations-Information-Objects ros-InformationObjects

    TCMessage {}
        FROM TCAPMessages tc-Messages

    APPLICATION-CONTEXT, dialogue-abstract-syntax
        FROM TC-Notation-Extensions tc-NotationExtensions


    eraseMessage {},
    playAnnouncement {},
    promptAndReceiveMessage {},
    promptAndCollectUserInformation {},
    scriptClose {},
    scriptEvent {},
    scriptInformation {},
    scriptRun {},
    specializedResourceReport
FROM IN-CS2-SCF-SRF-ops-args scf-srf-Operations

    cancel {},
    assistRequestInstructions {}
FROM IN-CS2-SSF-SCF-ops-args ssf-scf-Operations

    srf-scfActivationOfAssistPackage {}

FROM IN-CS2-SSF-SCF-pkgs-contracts-acs ssf-scf-Protocol

    id-package-specializedResourceControl,
    id-ac-srf-scf,
    id-contract-srf-scf,
    id-package-srf-scfCancel,
    id-package-scriptControl,
    id-package-messageControl,
    id-as-basic-srf-scf,
    classes, ros-InformationObjects, tc-Messages, tc-NotationExtensions,
    scf-srf-Operations, ssf-scf-Operations, ssf-scf-Protocol

FROM IN-CS2-object-identifiers {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-
identifiers (17) version1(0)}
```

```
;

-- Application Contexts --

srf-scf-ac APPLICATION-CONTEXT:: = {
    CONTRACT    srf-scf-contract
    DIALOGUE MODE  structured
    TERMINATION basic
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
            srf-scf-abstract-syntax }
    APPLICATION CONTEXT NAME   id-ac-srf-scf }


-- Contracts --

srf-scf-contract CONTRACT:: = {
    CONNECTION  emptyConnectionPackage
    INITIATOR CONSUMER OF  {srf-scfActivationOfAssistPackage {networkSpecificBoundSet}  }
    RESPONDER CONSUMER OF  {specializedResourceControlPackage {networkSpecificBoundSet}|
            srf-scfCancelPackage {networkSpecificBoundSet}|
            scriptControlPackage {networkSpecificBoundSet}|
            messageControlPackage {networkSpecificBoundSet}}
    ID      id-contract-srf-scf }


-- specializedResourceControl package --

specializedResourceControlPackage  {PARAMETERS-BOUND: bound}  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {playAnnouncement {bound} |
            promptAndCollectUserInformation {bound}
            }
    SUPPLIER INVOKES   {specializedResourceReport}
    ID      id-package-specializedResourceControl}

-- srf-scfCancel package --

srf-scfCancelPackage  {PARAMETERS-BOUND: bound}   OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {cancel {bound}}
    ID      id-package-srf-scfCancel}

-- scriptControl package --

scriptControlPackage  {PARAMETERS-BOUND: bound}  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   { scriptClose {bound}| scriptRun {bound} |
            scriptInformation {bound}}
    SUPPLIER INVOKES   { scriptEvent {bound}}
    ID      id-package-scriptControl}

-- messageControl package

messageControlPackage  {PARAMETERS-BOUND: bound} OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {eraseMessage {bound}| promptAndReceiveMessage {bound}}
    ID      id-package-messageControl
    }

-- Abstract Syntaxes --


srf-scf-abstract-syntax ABSTRACT-SYNTAX:: = {
    BASIC-SRF-SCF-PDUs
    IDENTIFIED BY  id-as-basic-srf-scf}

BASIC-SRF-SCF-PDUs:: = TCMessage {{SRF-SCF-Invokable},{SRF-SCF-Returnable} }

SRF-SCF-Invokable OPERATION:: = {
        assistRequestInstructions  {networkSpecificBoundSet}|
        cancel  {networkSpecificBoundSet}|
        playAnnouncement  {networkSpecificBoundSet}|
        promptAndCollectUserInformation  {networkSpecificBoundSet}|
        scriptClose  {networkSpecificBoundSet}|
        scriptEvent  {networkSpecificBoundSet}|
        scriptInformation  {networkSpecificBoundSet}|
        scriptRun  {networkSpecificBoundSet}|
        specializedResourceReport |
        promptAndReceiveMessage  {networkSpecificBoundSet}|
        eraseMessage {networkSpecificBoundSet}
        }
```

```
SRF-SCF-Returnable OPERATION:: = {
        assistRequestInstructions  {networkSpecificBoundSet}|
        cancel  {networkSpecificBoundSet}|
        playAnnouncement  {networkSpecificBoundSet}|
        promptAndCollectUserInformation  {networkSpecificBoundSet}|
        scriptClose  {networkSpecificBoundSet}|
        scriptInformation  {networkSpecificBoundSet}|
        scriptRun  {networkSpecificBoundSet}|
        promptAndReceiveMessage  {networkSpecificBoundSet}|
        eraseMessage  {networkSpecificBoundSet}
        }

END
```

8 SCF/SDF interface

8.1 Introduction to the reuse of ITU-T Recommendation X.500 [36] for SDF Interfaces

8.1.1 Alignment between the ITU-T Recommendation X.500 [36] concepts and the IN

The ITU-T Recommendation X.500 [36] are used to specify the SCF-SDF interface and the contents of the SDF. Most of the concepts of the ITU-T Recommendation X.500 [36] series are directly used in the IN environment, however some alignments need to be done at the terminology level to ensure that the concepts introduced in the Directory are correctly understood. The purpose of this part is to provide this alignment. It therefore only concentrates on the terms that are ambiguous in the IN environment.

When looking at the structure of the SCF, the Service Data Management is the part of the SCF responsible for the interactions with the SDF. It can be mapped onto the concept of Directory User Agent (DUA). When an SCF on behalf of a user wants to set-up an association with an SDF, an instance of a DUA is created in the Service Logic Program Instance (SLPI). It is killed when the association is ended.

The SDF is the entity responsible for answering the database requests. This Functional Entity (FE) can be mapped onto the Directory System Agent (DSA). When an association is set-up between an SCF and an SDF, an instance of a DSA is created for the length of the association.

The Directory is a collection of DSAs/SDFs. This set can be used for a specific service or for a variety of services. The notion of Directory is equivalent to the concept of database systems in IN.

The Directory can also be seen as a repository of data. IN services provides various kinds of data access to users. The information is organized into entries. An entry is a collection of information that can be identified (or named). When it represents an object (i.e. contains primary information about an object), it is called an object entry.

Objects are anything which are identifiable (can be named) and which are of interest to hold information on in the database. A typical example of an object is a user. Objects can be described by several entries. Each individual item of information that is used to describe an object is an attribute. They are associated with entries.

In the IN environment, the service provider is responsible for the management and the administration of the data contained in a DSA. Therefore the service provider plays the administrator role. It is the administrative authority in ITU-T Recommendation X.500 [36] terminology. The service provider enforces the security procedure (authentication and access control).

8.1.2 Use of a limited subset of ITU-T Recommendation X.500 [36]

The primary purpose of the ITU-T Recommendation X.500 [36] is to provide a directory service and not the description of the SCF-SDF interface as SG 11 wants to use them. The ITU-T Recommendation X.500 [36] functionality covers more than that needed for IN CS2. This clause indicates which aspects of the Directory Abstract Service should be considered and supported by implementors within the scope of CS2. It also mentions the attitude to adopt when an unsupported parameter is received. Profiling is used as a means of presenting the status of the different parameters.

It is important to note that the number of parameters carried in a message should be minimized, because each of them is associated with a load in the signalling traffic and with some processing time. This is the reason why the parameters are removed unless they are absolutely necessary when they are sent. On reception, removed parameters should not be treated but should be understood by the receiving entity. This allows the extensions of the profile in the future according to its actual description in the third edition of the Directory.

For convenience and clarity this profile is defined using ASN.1 subtyping facilities; however these definitions do not form a protocol specification. This simply indicates which parameters an implementation should not send. It does not change the behaviour of the receiving entity which shall still be capable of decoding values which conform to the original definition of the Directory Abstract Service. Nevertheless elements that are excluded by subtyping should be ignored.

8.1.3 Working assumptions

Several assumptions were used to design the Directory Abstract Service profile for IN CS2. References to the assumptions used are made in subclause 7.3 They are as follows:

Assumption 1: The version of the Directory Abstract Service used for CS2 is the third edition. The parameters only used for the 1988 version will be ignored. Functionalities that might be needed in future Capabilities Sets should be at least considered if not supported.

Assumption 2: The alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed.

Assumption 3: An SCF-SDF operation cannot be abandoned. If an operation takes too much time, its timer expires and there is no need to abandon it.

8.2 The SDF information model

ITU-T Recommendation X.501 [37] provides a generic information model that is needed to support the service provided by the Directory. In the context of IN, the generic information model should conform to the subclause 7.1 of ITU-T Recommendation X.501 [37] However certain aspects of this ITU-T Recommendation need not be supported. This includes the DIT content rules whose use is a local matter.

Some other points are outside the scope of this ITU-T Recommendation. This concerns the items associated with capabilities not covered by IN CS2. Therefore the following parts of ITU-T Recommendation X.501 [37] are not applicable:

- paragraphs f), h) and i) in subclause 16.2.3;
- paragraph a) in subclause 16.2.4. The compare operation is not used, the search operation is used instead. Therefore the FilterMatch permission replaces the Compare permission.

8.2.1 Information Framework

IN defines a number of extensions to the ITU-T Recommendation X.501 [37] information framework in order to meet IN service requirements. Only the enhancements are defined in these clauses. Unless stated the definition of other elements is the same as for ITU-T Recommendation X.501 [37] version 3.

8.2.1.1 METHOD

Each method represents a sequence of Directory Access Protocol (DAP) operations which are performed under the control of the DSA. The DUA is responsible for providing all necessary information in order for the DSA to complete the method. The DSA is responsible for collecting all information to be returned to the DUA.

For documentation purposes, it is suggested to add a description field to the class definition.

The **&InputAttributes** field identifies the attributes which may be submitted as input to the method execution.

The **&OutputAttributes** field identifies the attributes which may be returned as output of the method execution.

The **&SpecificInput** field provides that syntax of additional information which may be used as input to the method execution.

The **&SpecificOutput** field provides that syntax of additional information which may be used as output to the method execution.

The **&id** field uniquely identifies the method.

```
METHOD:: = CLASS {
    &InputAttributes       ATTRIBUTE OPTIONAL,
    &SpecificInput         OPTIONAL,
    &OutputAttributes      ATTRIBUTE OPTIONAL,
    &SpecificOutput        OPTIONAL,
    &description           PrintableString OPTIONAL,
    &id                    OBJECT IDENTIFIER UNIQUE}
WITH SYNTAX {
    [INPUT ATTRIBUTES      &InputAttributes]
    [SPECIFIC-INPUT        &SpecificInput]
    [OUTPUT ATTRIBUTES     &OutputAttributes]
    [SPECIFIC-OUTPUT       &SpecificOutput]
    [BEHAVIOUR         &description]
    ID                 &id}
```

8.2.1.2 DIT METHOD Use

8.2.1.2.1 Overview

A DIT METHOD Use is a specification provided by the subschema administrative authority to specify the METHOD types that may be used on entries of a particular object-class.

A DIT METHOD Use definition includes:

a) an indication of the object-class type to which it applies;

b) an indication of the METHOD types that shall be associated with the object-class whenever entries of that object-class are stored.

The DIT Method Use definition for a particular object-class also applies to any subclass which may be subsequently defined.

8.2.1.2.2 DIT METHOD Use specification

The abstract syntax of a DIT METHOD Use is expressed by the following ASN.1 type:

```
DITMethodUse    :: =    SEQUENCE {
    objectClass         OBJECT-CLASS.&id,
    methods     [1] SET OF METHOD.&id }
```

The correspondence between the parts of the definition, as listed in subclause 7.2.1.2.1, and the various components of the ASN.1 type defined above, is as follows:

a) the **objectClass** component identifies the object-class to which the DIT METHOD Use applies;

b) the **methods** component specifies types that shall be associated with the object-class whenever entries of that object-class are stored.

The **DITMethodUse** definition for a particular object-class also applies to any subclass which may be subsequently defined.

The METHOD-USE-RULE information object class is provided to facilitate the documentation of the DIT METHOD Use rules:

```
METHOD-USE-RULE:: = CLASS {
    &objectClassType           OBJECT-CLASS.&id   UNIQUE,
    &Mandatory         METHOD }
WITH SYNTAX {
    OBJECT-CLASS TYPE      &objectClassType
    METHODS            &Mandatory }
```

The METHOD-USE-RULE definition for a particular object-class also applies to any subclass which may be subsequently defined.

8.2.2 Basic Access Control

The following enhancements to the third edition ITU-T Recommendation X.500 [36] specification of Access Control Information (ACI) are required to support IN CS2 requirements on the SCF-SDF interface. Only the enhancements are described here. The remaining elements apply as described in the third edition ITU-T Recommendation X.500 [36].

8.2.2.1 ProtectedItems

The definitions of ProtectedItems is extended as follows:

```
ProtectedItems :: =    SEQUENCE {
    entry                   [0] NULL OPTIONAL,
    allUserAttributeTypes       [1] NULL OPTIONAL,
    attributeType           [2] SET OF AttributeType OPTIONAL,
    allAttributeValues          [3] SET OF AttributeType OPTIONAL, allUserAttributeTypesAndValues
    [4] NULL OPTIONAL,
    attributeValue          [5] SET OF AttributeTypeAndValue OPTIONAL,
    selfValue               [6] SET OF AttributeType OPTIONAL,
    rangeOfValues           [7] Filter OPTIONAL,
    maxValueCount           [8] SET OF MaxValueCount OPTIONAL,
    maxImmSub               [9] INTEGER OPTIONAL,
    restrictedBy                [10]    SET OF RestrictedValue OPTIONAL,
    contexts                [11]    SET OF ContextAssertion OPTIONAL,
    entryMethods            [30]    SET OF MethodIDs OPTIONAL}
```

entryMethods identifies the specified Methods for which the level of protection is to be applied.

```
MethodIDs:: =  METHOD.&id
```

8.2.2.2 GrantsAndDenials

The definitions of GrantsAndDenials is extended as follows:

```
GrantsAndDenials            :: =    BIT STRING {
    -- permissions that may be used in conjunction with
    -- with any component of ProtectedItems
    grantAdd            (0),
    denyAdd         (1),
    grantDiscloseOnError    (2),
    denyDiscloseOnError(3),
    grantRead           (4),
    denyRead            (5),
    grantRemove     (6),
    denyRemove          (7),
    -- permissions that may be used only in conjunction
    -- with the entry component
    grantBrowse         (8),
    denyBrowse          (9),
    grantExport         (10),
    denyExport          (11),
    grantImport         (12),
    denyImport          (13),
    grantModify         (14),
    denyModify          (15),
    grantRename     (16),
    denyRename          (17),
    grantReturnDN       (18),
    denyReturnDN        (19),
    -- permissions that may be used in conjunction
    -- with any component, except entry, of ProtectedItems
    grantCompare        (20),
    denyCompare     (21),
    grantFilterMatch        (22),
    denyFilterMatch     (23),
    -- permissions that may be used in conjunction
    -- with entryMethod component of ProtectedItems
    grantExecuteMethod      (30),
    denyExecuteMethod       (31) }
```

grantExecuteMethod means that the user can perform the specific Methods for the Entry.

NOTE: It is a matter for network operators as to whether the grantExecuteMethod permission bypasses the normal access control mechanisms for Entries and Attributes.

denyExecuteMethod means that the user can not perform the specific Methods for the Entry

8.2.3 Attribute contexts

8.2.3.1 Basic Service context

This Basic Service context associates an attribute value with a basic service for which the attribute value is semantically valid. For example, the Basic Service context will be associated with an Integrated Services Digital Network (ISDN) address to indicate the type of basic service that could be used with it. In the UPT case, this context allows the definition of registration addresses for different basic services.

```
basicServiceContext CONTEXT:: = {
    WITH SYNTAX BasicService
    ID          id-avc-basicService}

BasicService:: = INTEGER {
    telephony (1),
    faxGroup2-3 (2),
    faxGroup4 (3),
    teletexBasicAndMixed (4),
    teletexBazicAndProcessable (5),
    teletexBasic (6),
    syntaxBasedVideotex (7),
    internationalVideotex (8),
    telex (9),
    messageHandlingSystems (10),
    osiApplication (11),
    audioVisual (12)}
```

A presented value is considered to match a stored value if the context value (i.e. a basic service value) in the presented value is identical to that in the stored value.

8.2.3.2 Line Identity context

The line identity context associates an attribute value with the identity of a line for which the attribute value is semantically valid. For example, this Line Identity context will be associated with a routeing number to provide calling-line dependent routeing.

```
lineIdentityContext CONTEXT:: = {
    WITH SYNTAX IsdnAddress
    ID          id-avc-lineIdentity}
IsdnAddress:: = AddressString {ub-international-isdn-number}
```

8.2.3.3 Assignment context

The assignment context associates an attribute value with a Distinguished name (e.g. customer's number or customer's name) for with the attribute value is assigned. For example, assuming that a set of available resources is modelled as a multivalued attribute and customer has been designated by a distinguished name, this Assignment context will be associated with the used resource to provide the state of the ressource (reserved) and the name of the current customer using it.

```
assignmentContext  CONTEXT:: = {
    WITH SYNTAX DistinguishedName
    ID          id-avc-assignment }
```

8.2.4 Attribute Definitions

8.2.4.1 DIT Method Use operational attribute

The **methodUse** operational attribute is used to indicate the methods which shall be used with an object-class and all of its subclasses:

```
methodUse ATTRIBUTE:: = {
    WITH SYNTAX            MethodUseDescription
    EQUALITY MATCHING RULE objectIdentifierFirstComponentMatch
    USAGE                  directoryOperation
    ID                     id-soa-methodRuleUse }

MethodUseDescription   :: =    SEQUENCE {
    identifier         OBJECT-CLASS.&id,
    name               SET OF DirectoryString { ub-schema } OPTIONAL,
    description        DirectoryString { ub-schema } OPTIONAL,
    obsolete           BOOLEAN DEFAULT FALSE,
    information    [0] SET OF METHOD.&id }
```

The ID component of a value of the **methodUse** operational attribute is the object ID of the object-class type to which it applies. The value **id-oa-allObject-classTypes** indicates that it applies to all object-class types.

The **information** component of a value identifies the method types associated with the object-class identified by **ID.**

Every entry in the DIT is governed by at most one methodUse operational attribute. In addition the entry is also governed by all the methodUse operation attribute defined for the superclasses of its structural object class.

NOTE: This means that before processing an execute operation the SDF shall check the methodUse attributes associated with the structural object classes which belong to the inheritance chain of the entry's structural object class.

As a methodRule attribute is associated with a structural object class, it follows that all of the entries on the same structural object class will have the same Method Use Rule regardless of the DIT structure rule governing their location in the DIT and of the DIT content rule governing their contents.

8.3 The SCF-SDF Interface Protocol

8.3.1 Information types and common procedures

8.3.1.1 CommonArguments

```
IN-CommonArguments:: = CommonArguments (
    WITH COMPONENTS {
        ...,
        serviceControls    (IN-ServiceControls),
        aliasedRDNs        ABSENT })
```

The **serviceControls** component is described in subclause 7.3.1.2.

The **aliasedRDNs** component is present in the third edition only for compatibility reasons. It should always be omitted in the third edition implementations of the Directory (assumption 1).

8.3.1.2 ServiceControls

```
IN-ServiceControls:: = ServiceControls
    (WITH COMPONENTS {
        ...,
        timeLimit      ABSENT,
        sizeLimit      ABSENT,
        scopeOfReferralABSENT,
        attributeSizeLimit ABSENT})
```

The **timeLimit** component indicates the maximum elapsed time to fulfil a request. It is redundant with the operation timers of TCAP and therefore is not needed.

The **sizeLimit** and the **attributeSizeLimit** sets some size limits on the results either in terms of objects or in terms of attributes. This is useful when requests are expected to be general (the requestor does not know the structure of the DSA), but in the case of IN, this type of limitation does not seem applicable.

8.3.1.3 Entry Information Selection

```
IN-EntryInformationSelection:: = EntryInformationSelection
    (WITH COMPONENTS {
        ...,
        infoTypes      (attributeTypesAndValues) })
```

The **attributes** component specifies the attributes that should be returned in a retrieval service. The **allUserAttributes** option is kept even though it is advised to service specifiers to avoid its use which generates more traffic than needed. Instead the **select** option which precisely names the requested attributes should be used.

The **infoTypes** component specifies whether the attribute types and values should be returned or only the types. IN services are mainly interested in the attribute values that are relevant to the processing of the service. This component should be absent given its default value.

8.3.1.4 EntryInformation

```
IN-EntryInformation:: = EntryInformation
    (WITH COMPONENTS {
        ...,
        fromEntry      (TRUE),
        information    (WITH COMPONENTS {
                ...,
                attributeType      ABSENT}) OPTIONAL})
```

The **fromEntry** component indicates if a copy or the entry itself is returned. Since IN CS1 does not use copy mechanisms (assumption 3), only the default value of this component should be used.

The **information** parameter contains the relevant information which is returned. Given the choice made for the **infoTypes** component (see subclause 7.3.1.3), only the **attribute** option should be used.

8.3.1.5 Simple Public Key GSS-API Mechanism (SPKM) Token profile

The ITU-T Recommendation X.511 [39] **Bind** operation allows the use of SPKM security procedures to be specified. This subclause profiles the SPKM token which can be used for IN CS2 operations.

```
IN-Context-Data:: = Context-Data
    (WITH COMPONENTS {
        ...,
        channelId      ABSENT,
        seq-number     ABSENT})
```

Context-Data specifies options and the confidentiality, integrity, and one-way authentication function algorithm **ID**s.

The channel identifier (**channelId**) is not required for IN CS2, as only one channel is used.

The **seq-number** parameter indicates the sequence number of the token. This is not required for IN CS2 because the sequencing of messages is assumed via the lower protocol layers.

```
IN-Mic-Header:: = Mic-Header
    (WITH COMPONENTS {
        ...,
        snd-seq ABSENT})
```

The **snd-seq** parameter indicates the sequence number of the token. This is not required for IN CS2 because the sequencing of messages is assumed via the lower protocol layers.

Mic-Header contains the token ID, the context ID, and the integrity algorithm ID.

```
IN-Wrap-Header:: = Wrap-Header
    (WITH COMPONENTS {
        ...,
        snd-seq ABSENT}.)
```

Wrap-Header specifies header information containing the token ID, the context ID, the integrity algorithm ID, and the confidentiality algorithm ID.

```
IN-Del-Header:: = Del-Header
    (WITH COMPONENTS {
        ...,
        snd-seq ABSENT})
```

Del-Header contains the token ID, the context ID, and the integrity algorithm ID.

8.3.2 Operations

8.3.2.1 Bind operation

```
in-DirectoryBind  OPERATION:: = {
    ARGUMENT    DirectoryBindArgument
    RESULT      DirectoryBindResult
    ERRORS      {in-DirectoryBindError}
    CODE        in-opcode-in-bind}
```

8.3.2.2 Search operation

```
in-Search  OPERATION:: = {
    ARGUMENT    IN-SearchArgument
    RESULT      IN-SearchResult
    ERRORS      {nameError | in-ServiceError | securityError | attributeError | referral}
    CODE        id-opcode-in-search}
```

The **search** operation is used to search a portion of the DIT for entries of interest.

```
IN-SearchArgument:: = SearchArgument(
    WITH COMPONENTS {
        ...,
        searchAliases       (TRUE),
        selection           (IN-EntryInformationSelection),
        pagedResults        ABSENT,
        extendedFilter      ABSENT,
        COMPONENTS OF   IN-CommonArguments }) })
```

The **filter** parameter is used to eliminate entries from the search space. However the **extendedFilter** parameter was added in the 1993 version of the Directory for compatibility reasons and should therefore not be sent. Only the **filter** parameter should be sent.

The **searchAliases** parameter indicates whether the aliases encountered in the search space (except the base object) should be considered. Since in IN aliases are always dereferenced when searching, this parameter should be used only with its default value.

The **selection** parameter indicates what information from the entries, e.g. types and values, is requested (see subclause 7.3.1.3).

The **pagedResults** parameter is used to request a page by page result. The **pagedResults** parameter is used to present the results of a search operation in a page format. This type of information is not needed in IN CS2 since the SCF treats the results.

The **abandoned** error is not supported.

```
IN-SearchResult:: = SearchResult
    (WITH COMPONENTS {
        ...,
        searchInfo      (WITH COMPONENTS {
            ...,
            entries (WITH COMPONENT (IN-EntryInformation)),
            partialOutcomeQualifier     (PartialOutcomeQualifier
                    (WITH COMPONENTS {
                    ...,
                    queryReference      ABSENT}))OPTIONAL})})
```

The **entries** parameter contains the entries that satisfy the filter.

The **partialOutcomeQualifier** parameter is present when the search operation was not fully completed. It contains information on the reasons why the search operation was not finished and on where the operation was stopped.

The **queryReference** parameter is used when paged results were requested and therefore is not needed.

8.3.2.3 AddEntry operation

```
in-AddEntry  OPERATION:: = {
    ARGUMENT    AddEntryArgument
    RESULT      AddEntryResult
    ERRORS      {nameError | in-ServiceError | securityError | attributeError | updateError |
                 referral}
    CODE        id-opcode-in-addEntry}
```

8.3.2.4 RemoveEntry operation

```
in-RemoveEntry  OPERATION:: = {
    ARGUMENT    RemoveEntryArgument
    RESULT      RemoveEntryResult
    ERRORS      {nameError | in-ServiceError | securityError | updateError | referral}
    CODE        id-opcode-in-removeEntry}
```

8.3.2.5 ModifyEntry operation

```
in-ModifyEntry  OPERATION:: = {
    ARGUMENT    IN-ModifyEntryArgument
    RESULT      IN-ModifyEntryResult
    ERRORS      {nameError | in-ServiceError | securityError | attributeError | updateError |
                referral}
    CODE        id-opcode-in-modifyEntry}
IN-ModifyEntryArgument:: = ModifyEntryArgument
    (WITH COMPONENTS {
        ...,
        selection   (IN-EntryInformationSelection)})
```

The **selection** parameter specifies some attributes and values to be returned (See subclause 7.3.1.3).

```
IN-ModifyEntryResult:: = ModifyEntryResult
    (WITH COMPONENTS {
        ...,
        null        ,
        information Information
            (WITH COMPONENTS {
                ...,
                entry   (IN-EntryInformation)})}
```

If no information was to be retrieved with the modifyEntry operation, the **null** result is returned. Otherwise the information is to be returned in the **entry** component of the **information** result. For IN CS2 this component is specified in subclause 7.3.1.4.

8.3.2.6 Execute operation

The execute operation performs a sequence of execution steps, according to a pre-defined method, using input information and returns result information. Each step is either a DAP operation (that could be an execute operation), the execution of an algorithm or a decision test.

The parameters of the individual DAP operations are taken from the input parameters and the results of previous operations and/or the output of the algorithms associated with the method. The output parameters are taken from the results of the individual operations. The execute operation is considered to be an atomic operation.

```
execute OPERATION:: = {
    ARGUMENT    ExecuteArgument
    RESULT      ExecuteResult
    ERRORS      { attributeError | nameError |
                  serviceError | referral |
                 securityError |
                  updateError | executionError }
    CODE        id-opcode-execute }
```

```
ExecuteArgument:: =  OPTIONALLY-PROTECTED {
    SET {
        object          [0] Name,
        method-id       [1] METHOD.&id({SupportedMethods}),
        input-assertions   [2] SEQUENCE OF SEQUENCE {
                    type METHOD.&InputAttributes.&id({SupportedMethods}{@method-id}),
                    values SET OF METHOD.&InputAttributes.&id({SupportedMethods}{@method-id})
OPTIONAL,
                    valuesWithContext [0] SET OF SEQUENCE {
                            value   [0] METHOD.&InputAttributes.&id({SupportedMethods}{@method-
id})
OPTIONAL,
                            contextList [1] SET OF Context
                            } OPTIONAL
                    } OPTIONAL,
        specific-input  [3] METHOD.&SpecificInput({SupportedMethods}{@method-id}) OPTIONAL,
        COMPONENTS OF  CommonArguments },
    DIRQOP.&dapModifyEntryArg-QOP{@qop} }
```

The **object** field identifies the entry in the DIT from/on which the method is to be executed.

The **execute-id** field identifies the method which is to be executed within the SDF.

The **input-assertions** field provides a set of attribute values which are used as an input to the method execution.

The **specific-input** field identifies the additional information which is required by the SDF in order to perform the method.

```
ExecuteResult:: = OPTIONALLY-PROTECTED {
    SET {
        method-id       [1] METHOD.&id({SupportedMethods}),
        output-assertions  [2] SEQUENCE OF SEQUENCE {
                    type METHOD.&OutputAttributes.&id({SupportedMethods}{@method-id}),
                    values  SET OF METHOD.&OutputAttributes.&Type({SupportedMethods}{@method-
id,@.type}) OPTIONAL,
                    valuesWithContext [0] SET OF SEQUENCE {
                        value   [0]     METHOD.&OutputAttributes.&Type({SupportedMethods}{@method-
id,@.type}) OPTIONAL,
                        contextList [1] SET OF Context
                        } OPTIONAL
                    } OPTIONAL,
        specific-output [3] METHOD.&SpecificOutput({SupportedMethods}{@method-id})
                                    OPTIONAL,
        COMPONENTS OF CommonResults },
    DIRQOP.&dapModifyEntryRes-QOP{@qop} }
```

The **specific-output** field contains information returned as a result of the method execution.

The **output-assertions** contains attributes values returned as a result of the method execution.

```
SupportedMethods METHOD:: = {  ... }
```

The SupportedMethods set contains all of the defined methods for the interface. Its exact contents are a matter for local determination as it will depend on the service and network provider agreements being supported.

8.3.2.7 in-directoryUnbind operation

The **in-directoryUnbind** operation replaces the ITU-T Recommendation X.511 [39] **directoryUnbind** operation to provide class 4 operation behaviour for unbind procedures.

```
in-directoryUnbind  OPERATION:: = inEmptyUnbind
```

8.3.3 Errors

The precedence rule defined in the Directory should apply.

The **abandoned** and **abandonFailed** errors are not considered because not supported by CS2 (assumption 1 and assumption 4).

8.3.3.1 Bind error

```
IN-DirectoryBindError:: = DirectoryBindError
    (WITH COMPONENTS {
        ...,
        error   (WITH COMPONENTS {
                    securityError (SecurityProblem (1|2|7|10)),
                    serviceError (ServiceProblem (2))})})
```

SecurityProblem 10 indicates that the supplied SPKM token was found to be invalid.

In reception, all the possible errors should be supported to understand a Bind error.

8.3.3.2 Service error

```
in-ServiceError ERROR:: = {
    PARAMETER   IN-ServiceErrorParameter
    CODE        id-errcode-in-serviceError}

IN-ServiceErrorParameter:: =ServiceErrorParameter
    (WITH COMPONENTS {
        problem (ServiceProblem (1 | 2 | 3 | 4 |5 | 6 | 8 | 9 | 10 | 11 | 12))})
```

invalidQueryReference should not be sent because it is linked to the use of paged results.

8.3.3.4 execution Error

The executionError is returned by an Execute operation in the case of the operation not completing.

```
executionError ERROR:: = {
    PARAMETER   OPTIONALLY-PROTECTED {
                SET {
                    problem     [0]     ExecutionProblem,
                    COMPONENTS OF CommonResults },
                DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE            id-errcode-executionError }
ExecutionProblem :: =  INTEGER {
    missingInputValues (1),
    executionFailure(2) }
```

The executeProblem identifies the cause of the execute operation failure:

- **missingInputValues** is returned in the input-values field contains the wrong input information for the method being executed;

- **executionFailure** is returned when the method fails to complete correctly. This is caused by the failure of one of the DAP operations contained within the method.

8.4 Protocol overview

8.4.1 Remote operations

ITU-T Recommendation X.880 | ISO/IEC 9072-1 [47] defines several information object classes that are useful in the specification of ROS-based application protocols such as the various Directory protocols defined in this Directory Specification. A number of these classes are used in subsequent clauses. The specification techniques provided in ITU-T Recommendation X.880 | ISO/IEC 9072-1 [47] are used to define a generic protocol between objects. When realized as an SS7 application layer protocol, the concepts of ITU-T Recommendation X.880 | ISO/IEC 9072-1 [47] are mapped to SS7 concepts in ETS 300 287-1 [7].

8.4.2 Directory ROS-objects and contracts

ITU-T Recommendation X.519 | ISO/IEC 9594-5 [41] defines the abstract service between a DUA and the Directory which provides an access point to support a user accessing Directory services. Subclause 7.5 defines the sub-set of this abstract service used in the context of INs.

The **dua** class of ROS-object describes a DUA, being an instance of this class, as the initiator of the contract **dapContract** or the **dapExecuteContract**. These contracts are referred to in these Directory Specifications as the Directory Abstract Service. It is specified as a ROS-based information object in subclause 7.4.2.

```
dua ROS-OBJECT-CLASS:: = {
    INITIATES   {dapContract| dapExecuteContract}
    ID      id-rosObject-dua}
```

The **directory** class of ROS-object describes the provider of the Directory Abstract Service. This provider is the responder of the **dapContract/dapExecuteContract.**

```
directory ROS-OBJECT-CLASS:: = {
    RESPONDS    {dapContract| dapExecuteContract}
    ID      id-rosObject-directory}
```

The Directory is further modelled as being represented to a DUA by a DSA which supports the particular access point concerned. In the context of INs, each DSA is potentially an access point to the Directory.

The **directory** object is manifested as a set of DSAs *(each of which resides in an SDF)*. Each DSA comprising the **directory** is an instance of the **dap-dsa** class. A **dap-dsa** object assumes the role of responder in the **dapContract/ dapExecuteContract**.

```
dap-dsa ROS-OBJECT-CLASS:: = {
    RESPONDS    {dapContract| dapExecuteContract}
    ID      id-rosObject-dapDSA}
```

Future versions of this ITU-T Recommendation will enable DSAs to interact with one another to achieve various objectives.

8.4.3 DAP contract and packages

The **dapContract** is defined as an information object of class CONTRACT.

```
dapContract CONTRACT:: = {
    CONNECTION          dapConnectionPackage
    INITIATOR CONSUMER OF  {searchPackage | modifyPackage}
    ID                  id-contract-dap}
```

When a DUA and a DSA are located in different IN PE, this association contract shall be realized as an SS7 application layer protocol, referred to as the IN DAP. The definition of this protocol in terms of an SS7 AC is provided in 7.5.2.1 of this ITU-T Recommendation.

The dapContract is composed of a connection package, dapConnectionPackage, and two operation packages, searchPackage and modifyPackage.

The dapExecuteContract is defined as an information object of class CONTRACT.

```
dapExecuteContract CONTRACT:: = {
    CONNECTION          dapConnectionPackage
    INITIATOR CONSUMER OF  {searchPackage | modifyPackage | executePackage }
    ID                  id-contract-dapExecute}
```

The **dapExecuteContract** is composed of a connection package, **dapConnectionPackage**, and three operation packages, **searchPackage**, **modifyPackage**, and **executePackage**.

The **connection package**, **dapConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE. The bind operation of this connection package, directoryBind, is defined in ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39]. The unbind operation of this connection package, in-directoryUnbind is defined in clause 7.3.2.8.

```
dapConnectionPackage CONNECTION-PACKAGE:: = {
    BIND           in-DirectoryBind
    UNBIND      in-directoryUnbind
    ID          id-package-dapConnection}
```

The operation packages, **searchPackage** and **modifyPackage**, are defined as information objects of class OPERATION-PACKAGE. The operations of these operation packages are defined in ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39]. ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39] defines additional operations for supporting access to the Directory. Such operations are not used in the context of INs.

```
searchPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES       {search}
    ID                  id-package-search}
modifyPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES       {addEntry | removeEntry | modifyEntry}
    ID                  id-package-modify}
```

NOTE: These packages, when realized as ASEs, are used for the construction of ACs defined in this Specification. They are not intended to allow for claims of conformance to individual, or other combinations of, ASEs.

The operation package, **executePackage**, is defined as an information object of class OPERATION-PACKAGE. The operation of this operation package is defined in subclause 7.3.2.6.

```
executePackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {execute}
    ID          id-package-execute}
```

Since the DUA is the initiator of the **dapContract**/**dapExecuteContract**, it assumes the role of consumer of the operation packages of the contract. This means that only the DUA can invoke operations in these contracts and their SS7 realizations.

8.5 Directory protocol abstract syntax

8.5.1 Abstract syntaxes

This version of the DAP requires the support of three abstract syntaxes:

a) the abstract-syntax of TC dialogue-control PDUs, **dialogue-abstract-syntax**, which is needed to establish the dialogues between the SCFs and the SDFs and is specified in ETS 300 287-1 [7];

b) the abstract-syntax for conveying the PDUs for invoking **directoryBind** and **directioryUnbind** operations and reporting their outcome;

c) the abstract-syntax for conveying the PDUs for invoking the operations involved in the operation packages specified in 7.3.3 and reporting their outcome.

The ASN.1 type from which the values of the second abstract syntax are derived is specified using the parameterized types, **Bind {}** and **Unbind {}** which are defined in ITU-T Recommendation X.880 [47].

The ASN.1 type from which the values of the last abstract syntax are derived is specified using the parameterized types **TCMessage {}** defined in ETS 300 287-1 [7].

All these abstract syntaxes shall (as a minimum) be encoded according to the Basic ASN.1 encoding rules with the restrictions listed in ETS 300 287-1 [7].

8.5.1.1 DAP abstract syntax

The Directory ASEs that realize the operation packages specified in subclause 7.4.3, excluding the **executePackage**, share a single abstract syntax, **directoryOperationsAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectoryOperationsAbstractSyntax ABSTRACT-SYNTAX:: = {
    BasicDAP-PDUs
    IDENTIFIED BY  id-as-indirectoryOperationsAS}
BasicDAP-PDUs:: = TCMessage {{DAP-Invokable},{DAP-Returnable}}
DAP-Invokable  OPERATION:: = {search | addEntry | removeEntry | modifyEntry}
DAP-Returnable OPERATION:: = {search | addEntry | removeEntry | modifyEntry}
```

8.5.1.2 Extended DAP abstract syntax

The Directory ASEs that realize the operation packages specified in 7.4.3, including the **executePackage**, share a single abstract syntax, **inExtendedDirectoryOperationsAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inExtendedDirectoryOperationsAbstractSyntax ABSTRACT-SYNTAX:: = {
    Extended-BasicDAP-PDUs
    IDENTIFIED BY  id-as-inExtendedDirectoryOperationsAS}
Extended-BasicDAP-PDUs:: = TCMessage {{Extended-DAP-Invokable},{Extended-DAP-Returnable}}
Extended-DAP-Invokable  OPERATION:: = {search | addEntry | removeEntry | modifyEntry | execute}
Extended-DAP-Returnable OPERATION:: = {search | addEntry | removeEntry | modifyEntry | execute}
```

8.5.1.3 DAP binding abstract syntax

The realization of the connection package specified in 7.4.3 uses a separate abstract syntax, **directoryBindingAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX

```
inDirectoryBindingAbstractSyntax   ABSTRACT-SYNTAX:: = {
    DAPBinding-PDUs
    IDENTIFIED BY  id-as-indirectoryBindingAS}
DAPBinding-PDUs:: = CHOICE {
    bind    Bind {directoryBind},
    unbind  Unbind {in-directoryUnbind}}
```

8.5.1.4 SESE abstract syntax

An additional abstract syntax, **inSESEAbstractSyntax,** is used in the iNdirectoryAccessWith3seAC defined in subclause 7.4.2.3**.** This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inSESEAbstractSyntax   ABSTRACT-SYNTAX:: = {
    SESEapdus {{spkmThreeWay},NoInvocationId}
    IDENTIFIED BY  {id-as-inSESEAS}}
```

SESEapdus is imported from ITU-T Recommendation X.832 [46] and spkmThreeWay is imported from ITU-T Recommendation X.519 [41].

8.5.2 Directory ACs

8.5.2.1 Directory access AC

The **dapContract** is realized as the **iNdirectoryAccessAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
iNdirectoryAccessAC APPLICATION-CONTEXT:: =  {
    CONTRACT               dapContract
    DIALOGUE MODE          structured
    TERMINATION                basic
    ABSTRACT SYNTAXES          {dialogue-abstract-syntax |
                               inDirectoryOperationsAbstractSyntax |
                               inDirectoryBindingAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-indirectoryAccessAC}
```

If three-way authentication is required then the **dapContract** is realized as the **iNdirectoryAccessWith3seAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
iNdirectoryAccessWith3seAC APPLICATION-CONTEXT:: =  {
    CONTRACT           dapContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                       inDirectoryOperationsAbstractSyntax |
                       inDirectoryBindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-indirectoryAccessWith3seAC}
```

8.5.2.2 Extended Directory Access AC

The **dapExecuteContract** is realized as the **inExtendedDirectoryAccessAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inExtendedDirectoryAccessAC APPLICATION-CONTEXT:: =  {
    CONTRACT              dapExecuteContract
    DIALOGUE MODE         structured
    TERMINATION               basic
    ABSTRACT SYNTAXES         {dialogue-abstract-syntax |
                              inExtendedDirectoryOperationsAbstractSyntax |
                              inDirectoryBindingAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-inExtendedDirectoryAccessAC}
```

If three-way authentication is required then the **dapExecuteContract** is realized as the **inExtendedDirectoryAccessAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inExtendedDirectoryAccessWith3seAC APPLICATION-CONTEXT:: =  {
    CONTRACT           dapExecuteContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                       inExtendedDirectoryOperationsAbstractSyntax |
                       inExtendedDirectoryBindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-inExtendedDirectoryAccessWith3seAC}
```

8.5.3 Operation codes

The operations involved in the packages defined in this ITU-T Recommendation are specified in ITU-T Recommendation X.519 [41] where the assigned operation codes are imported from ITU-T Recommendation X.519 [41].

8.5.4 Error codes

The errors involved in the packages defined in this ITU-T Recommendation are specified in ITU-T Recommendation X.519 [41] where the assigned error codes are imported from ITU-T Recommendation X.519 [41].

8.5.5 Versions and the rules for extensibility

The Directory may be distributed and more than two Directory Application Entities may interoperate to service a request. The Directory AEs may be implemented conforming to different editions of the Directory specification of the Directory service which may or may not be represented by different protocol version numbers. The version number is negotiated to the highest common version number between two directly binding Directory AEs.

8.5.5.1 Version negotiation

When accepting an association, .e., binding, utilising the DAP, the version negotiated shall only affect the point to point aspects of the protocol exchanged between the DUA and the DSA to which it is connected. Subsequent requests or responses on the dialogue shall be constrained by the version negotiated.

NOTE: There are no point to point aspects of the DAP that are currently indicated by different protocol versions.

8.5.5.2 DUA side

8.5.5.2.1 Request and response processing at the DUA side

The DUA may initiate requests using the highest edition of the specification of that request it supports. If one or more elements of the request are critical, it shall indicate the extension number(s) in the critical Extensions parameter.

NOTE: If the information the extension replaced in a CHOICE, ENUMERATED or INTEGER (used as ENUMERATED) type would be essential for proper operation in a DSA implemented according to an earlier edition of the Specification, it is recommended that the extension be marked critical.

When processing a response, a DUA shall:

a) ignore all unknown bit name assignments within a bit string; and

b) ignore all unknown named numbers in an ENUMERATED type or INTEGER type that is being used in the enumerated style, provided the number occurs as an optional element of a SET or SEQUENCE; and

c) ignore all unknown elements in SETs, at the end of SEQUENCEs, or in CHOICEs where the CHOICE is itself an optional element of a SET or SEQUENCE;

NOTE: Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

d) not consider the receipt of unknown attribute types and attribute values as a protocol violation; and

e) optionally report the unknown attribute types and attribute values to the user.

8.5.5.2.2 Extensibility rules for error handling at the DUA side

When processing a known error type with unknown indicated problems and parameters, a DUA

a) shall not consider the receipt of unknown indicated problems and parameters as a protocol violation (i.e. it shall not issue a TC-U-REJECT or abort the dialogue); and

b) may optionally report the additional error information to the user.

When processing an unknown error type, a DUA

a) shall not consider the receipt of unknown error type as a protocol violation (i.e., it shall not issue a TC-U-REJECT or abort the application association); and

b) may optionally report the error to the user.

8.5.5.3 Request processing at the DSA side

If any DSA performing an operation detects an element **criticalExtensions** whose semantic is unkown, it shall return an **unavailableCriticalExtension** indication as a **serviceError**.

NOTE: If a **criticalExtensions** string with one or more zero values is received, this indicates either that the extensions corresponding to the values are not present or are not critical. The presence of a zero value in a **criticalExtensions** string shall not be inferred as either the presence or absence of the corresponding extension in the APDU.

Otherwise, when processing a request from a DUA, a DSA shall:

a) ignore all unknown bit name assignments within a bit string; and

b) ignore all unknown named numbers in an ENUMERATED type or INTEGER type that is being used in the enumerated style, provided the number occurs as an optional element of a SET or SEQUENCE; and

c) ignore all unknown elements in SETs, at the end of SEQUENCEs, or in CHOICEs where the CHOICE is itself an optional element of a SET or SEQUENCE.

NOTE: Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is not specified in IN CS2.

8.6 Conformance

This clause defines the requirements for conformance to this specification.

8.6.1 Conformance by SCFs

An SCF implementation claiming conformance to this specification shall satisfy the requirements specified in subclauses 7.6.1.1 to 7.6.1.3.

8.6.1.1 Statement requirements

The following shall be stated:

a) the operations of the **iNdirectoryAccessAC** application-context that the SCF is capable of invoking for which conformance is claimed;

b) the security-level(s) for which conformance is claimed (none, simple, strong);

c) the extensions listed in the table of 7.3.1 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39], that the SCF is capable of initiating for which conformance is claimed.

8.6.1.2 Static requirements

An SCF shall:

a) have the capability of supporting the **iNdirectoryAccessAC** application-context as defined by its abstract syntax in subclause 7.5.2.1;

b) conform to the extensions for which conformance was claimed in subclause 7.6.1.1 point c).

8.6.1.3 Dynamic requirements

An SCF shall:

a) conform to the mapping onto used services defined in subclause 18.1.6;

b) shall conform to the rules of extensibility procedures defined in subclause 7.5.5.2.

8.6.2 Conformance by SDFs

An SDF implementation claiming conformance to this specification shall satisfy the requirements specified in subclauses 7.6.2.1 to 7.6.2.3.

8.6.2.1 Statement requirements

The following shall be stated:

a) the application-context for which conformance is claimed. The present version of this ITU-T Recommendation only requires conformance to the **iNdirectoryAccessAC** application-context;

NOTE: An AC shall not be divided except as stated herein; in particular, conformance shall not be claimed to particular operations.

b) the security-level(s) for which conformance is claimed (none, simple, strong);

c) the attribute types for which conformance is claimed and whether for attributes based on the syntax **DirectoryString,** conformance is claimed for the **UNIVERSAL STRING** choice;

d) the object classes, for which conformance is claimed;

e) the extensions listed in the table of 7.3.1 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39], that the SDF is capable of responding to for which conformance is claimed;

f) whether conformance is claimed for collective attributes as defined in subclause 8.8 of ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37] and 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39];

g) whether conformance is claimed for hierarchical attributes as defined in subclauses 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39];

h) the operational attribute types defined in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37] and any other operational attribute types for which conformance is claimed;

i) whether conformance is claimed for return of alias names as described in subclause 7.7.1 of ITU-T Recommendation X.511 |ISO/IEC 9594-3 [39];

j) whether conformance is claimed for indicating that returned entry information is complete, as described in subclause 7.7.6 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39];

k) whether conformance is claimed for modifying the object class attribute to add and/or remove values identifying auxiliary object classes, as described in 11.3.2 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39];

l) whether conformance is claimed to Basic Access Control;

m) whether conformance is claimed to Simplified Access Control;

n) the name bindings for which conformance is claimed;

o) whether the SDF is capable of administering collective attributes, as defined in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37];

p) whether conformance is claimed for contexts.

8.6.2.2 Static requirements

An SDF shall:

a) have the capability of supporting the application-contexts for which conformance is claimed as defined by their abstract syntax in subclause 7.5.2.1;

b) have the capability of supporting the information framework defined by its abstract syntax in ITU-T Recommendation X.501 | ISO/IEC9594-2 [37];

c) have the capability of supporting the attribute types for which conformance is claimed; as defined by their abstract syntaxes;

d) have the capability of supporting the object classes for which conformance is claimed, as defined by their abstract syntaxes;

e) conform to the extensions for which conformance was claimed in subclause 7.6.2.1;

f) if conformance is claimed for collective attributes, have the capability of performing the related procedures defined in subclauses 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39];

g) if conformance is claimed for hierarchical attributes, have the capability of performing the related procedures defined in subclauses 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39];

h) have the capability of supporting the operational attribute types for which conformance is claimed;

i) if conformance is claimed to Basic Access Control, have the capability of holding ACI items that conform to the definitions of Basic Access Control;

j) if conformance is claimed to Simplified Access Control, have the capability of holding ACI items that conform to the definitions of Simplified Access Control.

8.6.2.3 Dynamic requirements

An SDF shall:

a) conform to the mapping onto used services defined in subclause 18.1.6.

b) conform to the rules of extensibility procedures defined in subclause 7.5.5.3;

c) if conformance is claimed to Basic Access Control, have the capability of protecting information within the SDF in accordance with the procedures of Basic Access Control;

d) if conformance is claimed to Simplified Access Control, have the capability of protecting information within the SDF in accordance with the procedures of Simplified Access Control.

8.7 ASN.1 Modules for the SCF-SDF interface

The following set of ASN.1 modules define the SCF-SDF interface for IN CS2. They contain all the modifications to the Directory specifications as required for the support of INs.

The modules also contain the definitions which are impacted by these modifications because they make use of a modified type.

8.7.1 IN-CS2-SDF-InformationFramework module

This module contains the enhancements made to ITU-T Recommendations X.501 [37] (InformationFramework module) to meet the IN CS2 needs.

```
IN-CS2-SDF-InformationFramework
        { ccitt ITU-T Recommendation Q.1228 module(0) sdfInformationFramework(9) version1(0) }
DEFINITIONS:: =
BEGIN
-- EXPORTS ALL--
```

```
-- types and values are exported for use in the ASN.1 modules which define the IN profile of the
Directory
-- Abstract Service,  the Directory Access Protocol and the Directory Information Shadowing
Protocol.
-- The types and values defined in this module are exported for use in the other ASN.1 modules
contained
-- within the Directory Specifications, and for the use of other applications which will use
them to access
-- Directory services. Other applications may use them for their own purposes, but this will not
constrain
-- extensions and modifications needed to maintain or improve the Directory service.
IMPORTS
informationFramework, upperBounds, selectedAttributeTypes
    FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 3}
ATTRIBUTE, OBJECT-CLASS, objectClass, aliasedEntryName
    FROM InformationFramework  informationFramework
DirectoryString{}, objectIdentifierFirstComponentMatch
    FROM SelectedAttributeTypes    selectedAttributeTypes
ub-schema
    FROM UpperBounds           upperBounds
id-soa-methodRuleUse
    FROM IN-CS2-object-identifiers
        { ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-identifiers(17) version1(0)
}
;
-- attribute data types --
--  Definition of the following information object set is deferred, perhaps to standardized
--  profiles or to protocol implementation conformance statements. The set is required to
--  specify a table constraint on the values component of Attribute, the value component
--  of AttributeTypeAndValue, and the assertion component of AttributeValueAssertion.
SupportedAttributes ATTRIBUTE:: = { objectClass | aliasedEntryName , ...}

-- METHOD information object class specification --
METHOD:: = CLASS {
    &InputAttributes       ATTRIBUTE OPTIONAL,
    &SpecificInput     OPTIONAL,
    &OutputAttributes  ATTRIBUTE OPTIONAL,
    &SpecificOutput    OPTIONAL,
    &description           PrintableString OPTIONAL,
    &id                OBJECT IDENTIFIER UNIQUE}
WITH SYNTAX {
    [INPUT ATTRIBUTES      &InputAttributes]
    [SPECIFIC-INPUT        &SpecificInput]
    [OUTPUT ATTRIBUTES     &OutputAttributes]
    [SPECIFIC-OUTPUT       &SpecificOutput]
    [BEHAVIOUR         &description]
    ID                 &id}

DITMethodUse   :: =    SEQUENCE {
    objectClass    OBJECT-CLASS.&id,
    methods     [1] SET OF METHOD.&id }
METHOD-USE-RULE:: = CLASS {
    &objectClassType           OBJECT-CLASS.&id   UNIQUE,
    &Mandatory         METHOD }
WITH SYNTAX {
    OBJECT-CLASS TYPE      &objectClassType
    METHODS            &Mandatory }
-- attributes --
methodUse ATTRIBUTE:: = {
    WITH SYNTAX            MethodUseDescription
    EQUALITY MATCHING RULE objectIdentifierFirstComponentMatch
    USAGE                  directoryOperation
    ID                     id-soa-methodRuleUse }
MethodUseDescription   :: =    SEQUENCE {
    identifier         OBJECT-CLASS.&id,
    name               SET OF DirectoryString { ub-schema } OPTIONAL,
    description        DirectoryString { ub-schema } OPTIONAL,
    obsolete           BOOLEAN DEFAULT FALSE,
    information    [0] SET OF METHOD.&id }
END
```

8.7.2 IN-CS2-SDF-BasicAccessControl module

This module contains the enhancements made to the ITU-T Recommendation X.501 [37] (InformationFramework module)to meet the IN needs.

```
IN-CS2-SDF-BasicAccessControl
        { ccitt ITU-T Recommendation Q.1228 module(0) sdfBasicAccessControl(10) version1(0) }
DEFINITIONS:: =
BEGIN
-- EXPORTS All --
-- The types and values defined in this module are exported for use in the other ASN.1 modules
contained
-- within the Directory Specifications, and for the use of other applications which will use
them to access
-- Directory services. Other applications may use them for their own purposes, but this will not
constrain
-- extensions and modifications needed to maintain or improve the Directory service.
IMPORTS
informationFramework, upperBounds, selectedAttributeTypes, basicAccessControl,
directoryAbstractService
    FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 3}
ATTRIBUTE, AttributeType, AttributeTypeAndValue, SubtreeSpecification, ContextAssertion
    FROM InformationFramework informationFramework
id-aca-prescriptiveACI, id-aca-entryACI, id-aca-subentryACI,
sdf-InformationFramework
    FROM IN-CS2-object-identifiers
        { ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-identifiers(17) version1(0)
}
ub-tag
    FROM UpperBounds upperBounds
METHOD
    FROM IN-CS2-SDF-InformationFramework
sdf-InformationFramework
Filter
    FROM DirectoryAbstractService  directoryAbstractService
NameAndOptionalUID, directoryStringFirstComponentMatch, DirectoryString{}
    FROM SelectedAttributeTypes selectedAttributeTypes
MaxValueCount, RestrictedValue, AuthenticationLevel, Precedence
    FROM BasicAccessControl basicAccessControl
;

-- types --
ACIItem     :: =        SEQUENCE {
    identificationTag       DirectoryString { ub-tag },
    precedence          Precedence,
    authenticationLevelAuthenticationLevel,
    itemOrUserFirst     CHOICE {
        itemFirst       [0] SEQUENCE {
            protectedItems  ProtectedItems,
            itemPermissions SET OF ItemPermission },
        userFirst       [1] SEQUENCE {
            userClasses     UserClasses,
            userPermissions SET OF UserPermission }}}
ProtectedItems  :: =    SEQUENCE {
    entry                       [0] NULL OPTIONAL,
    allUserAttributeTypes           [1] NULL OPTIONAL,
    attributeType               [2] SET OF AttributeType OPTIONAL,
    allAttributeValues              [3] SET OF AttributeType OPTIONAL,
    allUserAttributeTypesAndValues [4] NULL OPTIONAL,
    attributeValue              [5] SET OF AttributeTypeAndValue OPTIONAL,
    selfValue                   [6] SET OF AttributeType OPTIONAL,
    rangeOfValues               [7] Filter OPTIONAL,
    maxValueCount               [8] SET OF MaxValueCount OPTIONAL,
    maxImmSub                   [9] INTEGER OPTIONAL,
    restrictedBy                    [10]    SET OF RestrictedValue OPTIONAL,
    contexts                    [11]    SET OF ContextAssertion OPTIONAL,
    entryMethods                [30]    SET OF MethodIDs OPTIONAL}
MethodIDs:: =   METHOD.&id
UserClasses :: =    SEQUENCE {
    allUsers    [0] NULL OPTIONAL,
    thisEntry   [1] NULL OPTIONAL,
    name        [2] SET OF NameAndOptionalUID OPTIONAL,
    userGroup   [3] SET OF NameAndOptionalUID OPTIONAL,
                    -- dn component must be the name of an
                    -- entry of GroupOfUniqueNames
    subtree [4] SET OF SubtreeSpecification OPTIONAL}
```

```
ItemPermission          :: =    SEQUENCE {
    precedence      Precedence OPTIONAL,
                    -- defaults to precedence in ACIItem --
    userClasses     UserClasses,
    grantsAndDenials    GrantsAndDenials }
UserPermission          :: =    SEQUENCE {
    precedence      Precedence OPTIONAL,
                    -- defaults to precedence in ACIItem
    protectedItems  ProtectedItems,
    grantsAndDenials    GrantsAndDenials }
GrantsAndDenials            :: =    BIT STRING {
    -- permissions that may be used in conjunction with
    -- with any component of ProtectedItems
    grantAdd            (0),
    denyAdd         (1),
    grantDiscloseOnError    (2),
    denyDiscloseOnError(3),
    grantRead           (4),
    denyRead            (5),
    grantRemove     (6),
    denyRemove          (7),
    -- permissions that may be used only in conjunction
    -- with the entry component
    grantBrowse         (8),
    denyBrowse          (9),
    grantExport         (10),
    denyExport          (11),
    grantImport         (12),
    denyImport          (13),
    grantModify         (14),
    denyModify          (15),
    grantRename      (16),
    denyRename          (17),
    grantReturnDN       (18),
    denyReturnDN        (19),
    -- permissions that may be used in conjunction
    -- with any component, except entry, of ProtectedItems
    grantCompare        (20),
    denyCompare      (21),
    grantFilterMatch        (22),
    denyFilterMatch     (23),
    -- permissions that may be used in conjunction
    -- with entryMethod component of ProtectedItems
    grantExecuteMethod (30),
    denyExecuteMethod  (31) }
-- attributes --
prescriptiveACI     ATTRIBUTE:: =  {
    WITH SYNTAX             ACIItem
    EQUALITY MATCHING RULE directoryStringFirstComponentMatch
    USAGE                   directoryOperation
    ID                      id-aca-prescriptiveACI }
entryACI            ATTRIBUTE:: =  {
    WITH SYNTAX             ACIItem
    EQUALITY MATCHING RULE directoryStringFirstComponentMatch
    USAGE                   directoryOperation
    ID                      id-aca-entryACI }
subentryACI         ATTRIBUTE:: =  {
    WITH SYNTAX             ACIItem
    EQUALITY MATCHING RULE directoryStringFirstComponentMatch
    USAGE                   directoryOperation
    ID                      id-aca-subentryACI }
END
```

8.7.3 IN-CS2-SCF-SDF-Operations module

```
IN-CS2-SCF-SDF-Operations
           {ccitt ITU-T Recommendation Q.1228 module(0) scf-sdf-operations(11) version1(0) }
DEFINITIONS:: =
BEGIN
-- EXPORTS All --
-- The types and values defined in this module are exported for use in the other ASN.1 modules
contained
-- within the IN Directory Specifications, and for the use of other applications which will use
them to access
-- IN Directory services. Other applications may use them for their own purposes, but this will
not constrain
-- extensions and modifications needed to maintain or improve the Directory service.
IMPORTS
informationFramework, distributedOperations, authenticationFramework, upperBounds,
directoryAbstractService, enhancedSecurity
    FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 3}
CONTEXT, Context, DistinguishedName, Name
    FROM InformationFramework informationFramework
OperationProgress, ReferenceType, Exclusions, AccessPoint, ContinuationReference
    FROM  DistributedOperations  distributedOperations
CertificationPath, SIGNED {}, SIGNATURE {}, AlgorithmIdentifier
    FROM  AuthenticationFramework authenticationFramework
id-avc-assignment,
contexts, ros-InformationObjects, sdf-InformationFramework
    FROM IN-CS2-object-identifiers
        { ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-identifiers(17) version1(0)
}
basicServiceContext, lineIdentityContext
    FROM IN-Contexts    contexts
Code, OPERATION, ERROR
    FROM Remote-Operations-Information-Objects ros-InformationObjects
inEmptyUnbind
    FROM IN-CS2-classes {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-classes(4)
version1(0)}
METHOD
    FROM  IN-CS2-SDF-InformationFramework
sdf-InformationFramework
OPTIONALLY-PROTECTED{}, DIRQOP
    FROM EnhancedSecurity enhancedSecurity
CommonArguments, CommonResults, attributeError, nameError, serviceError, securityError,
referral, updateError
    FROM DirectoryAbstractService   directoryAbstractService
;
execute OPERATION:: = {
    ARGUMENT    ExecuteArgument
    RESULT      ExecuteResult
    ERRORS      { attributeError | nameError |
                  serviceError | referral |
                 securityError |
                  updateError | executionError }
    CODE        id-opcode-execute }
ExecuteArgument:: =  OPTIONALLY-PROTECTED {
    SET {
        object      [0] Name,
        method-id       [1] METHOD.&id({SupportedMethods}),
        input-assertions    [2] SEQUENCE OF SEQUENCE {
                            type    METHOD.&InputAttributes.&id({SupportedMethods}{@method-id}),
                            values  SET OF
METHOD.&InputAttributes.&Type({SupportedMethods}{@method-id,@.type}) OPTIONAL,
                            valuesWithContext [0] SET OF SEQUENCE {
                                value   [0]
    METHOD.&InputAttributes.&Type({SupportedMethods}{@method-id,@.type}) OPTIONAL,
                                contextList [1]    SET OF Context
                                } OPTIONAL
                            } OPTIONAL,
        specific-input [3] METHOD.&SpecificInput({SupportedMethods}{@method-id}) OPTIONAL,
        COMPONENTS OF CommonArguments },
    DIRQOP.&dapModifyEntryArg-QOP{@qop} }
```

```
ExecuteResult:: = OPTIONALLY-PROTECTED {
    SET {
        method-id      [1] METHOD.&id({SupportedMethods}),
        output-assertions  [2] SEQUENCE OF SEQUENCE {
                           type    METHOD.&OutputAttributes.&id({SupportedMethods}{@method-id}),
                           values  SET OF
METHOD.&OutputAttributes.&Type({SupportedMethods}{@method-id,@.type})OPTIONAL,
                           valuesWithContext [0] SET OF SEQUENCE {
                               value   [0]
    METHOD.&OutputAttributes.&Type({SupportedMethods}{@method-id,@.type})OPTIONAL,
                               contextList [1] SET OF Context
                               } OPTIONAL
                           } OPTIONAL,
        specific-output [3] METHOD.&SpecificOutput({SupportedMethods}{@method-id}) OPTIONAL,
        COMPONENTS OF CommonResults },
    DIRQOP.&dapModifyEntryRes-QOP{@qop} }
SupportedMethods METHOD:: = { ... }
in-directoryUnbind  OPERATION:: = inEmptyUnbind
assignmentContext  CONTEXT:: = {
    WITH SYNTAX DistinguishedName
    ID          id-avc-assignment }

executionError ERROR:: = {
    PARAMETER   OPTIONALLY-PROTECTED {
                    SET {
                        problem     [0]     ExecutionProblem ,
                        COMPONENTS OF CommonResults },
                    DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE            id-errcode-executionError }
ExecutionProblem :: =  INTEGER {
    missingInputValues (1),
    executionFailure(2) }
-- object identifier assignment
-- error codes
id-errcode-executionError      Code:: = local: 10
-- operation codes
id-opcode-execute          Code:: =local: 10

END
```

8.7.4 IN-CS2-SCF-SDF-Protocol module

This subclause includes all of the ASN.1 type and value definitions contained in this Directory Specification, in the form of the ASN.1 module, "IN-CS2-SCF-SDF-Protocol ".

```
IN-CS2-SCF-SDF-Protocol {ccitt ITU-T Recommendation q 1218 modules(0) in-scf-sdf-protocol(12)
version1(0)}
DEFINITIONS:: =
 BEGIN
-- EXPORTS All --
 -- The types and values defined in this module are exported for use in the other ASN.1 modules
contained
-- within the Directory Specifications, and for the use of other applications which will use
them to access
-- Directory services. Other applications may use them for their own purposes, but this will not
constrain
-- extensions and modifications needed to maintain or improve the Directory service.
IMPORTS
directoryAbstractService , directorySecurityExchanges, protocolObjectIdentifiers
    FROM  UsefulDefinitions  ds-UsefulDefinitions
ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, CONNECTION-PACKAGE,
 OPERATION
    FROM Remote-Operations-Information-Objects ros-InformationObjects

Bind{}, Unbind{}
    FROM Remote-Operations-Generic-ROS-PDUs ros-genericPDUs

TCMessage {}
    FROM  TCAPMessages tc-Messages

APPLICATION-CONTEXT, dialogue-abstract-syntax
    FROM TC-Notation-Extensions tc-NotationExtensions
```

```
id-ac-indirectoryAccessAC, id-ac-inExtendedDirectoryAccessAC, id-rosObject-dua, id-
rosObject-directory,
id-rosObject-dapDSA,
id-contract-dap, id-contract-dapExecute, id-package-dapConnection, id-package-search,
id-package-modify,
id-package-execute,
id-as-indirectoryOperationsAS, id-as-inExtendedDirectoryOperationsAS, id-as-
indirectoryBindingAS,
id-as-inSESEAS,
id-ac-inExtendedDirectoryAccessWith3seAC, id-ac-indirectoryAccessWith3seAC,
ros-InformationObjects, ros-genericPDUs, tc-Messages, tc-NotationExtensions, sese-
APDUs,
ds-UsefulDefinitions, scf-sdf-Operations
    FROM IN-CS2-object-identifiers
       {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-identifiers (17)
version1 (0)}
directoryBind, search, addEntry, removeEntry, modifyEntry
    FROM DirectoryAbstractService directoryAbstractService
SESEapdus{}, NoInvocationId
    FROM SeseAPDUs sese-APDUs
spkmThreeWay
    FROM DirectorySecurityExchanges directorySecurityExchanges
id-se-threewayse
    FROM ProtocolObjectIdentifiers protocolObjectIdentifiers
execute, in-directoryUnbind
    FROM IN-CS2-SCF-SDF-Operations
scf-sdf-Operations
;
-- application contexts --
iNdirectoryAccessAC APPLICATION-CONTEXT:: =  {
    CONTRACT           dapContract
    DIALOGUE MODE      structured
    TERMINATION            basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectoryOperationsAbstractSyntax |
                           inDirectoryBindingAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-indirectoryAccessAC}
iNdirectoryAccessWith3seAC APPLICATION-CONTEXT:: =  {
    CONTRACT           dapContract
    DIALOGUE MODE      structured
    TERMINATION            basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES          {dialogue-abstract-syntax |
                       inDirectoryOperationsAbstractSyntax |
                       inDirectoryBindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-indirectoryAccessWith3seAC}
inExtendedDirectoryAccessAC APPLICATION-CONTEXT:: =  {
    CONTRACT           dapExecuteContract
    DIALOGUE MODE      structured
    TERMINATION            basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inExtendedDirectoryOperationsAbstractSyntax |
                           inDirectoryBindingAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-inExtendedDirectoryAccessAC}
inExtendedDirectoryAccessWith3seAC APPLICATION-CONTEXT:: =  {
    CONTRACT           dapExecuteContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                       inExtendedDirectoryOperationsAbstractSyntax |
                       inDirectoryBindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-inExtendedDirectoryAccessWith3seAC}
-- ROS-objects --
dua ROS-OBJECT-CLASS:: = {
    INITIATES   {dapContract| dapExecuteContract}
    ID      id-rosObject-dua}
directory ROS-OBJECT-CLASS:: = {
    RESPONDS    {dapContract| dapExecuteContract}
    ID      id-rosObject-directory}
dap-dsa ROS-OBJECT-CLASS:: = {
    RESPONDS    {dapContract| dapExecuteContract}
    ID      id-rosObject-dapDSA}
-- contracts --
```

```
dapContract CONTRACT:: = {
    CONNECTION          dapConnectionPackage
    INITIATOR CONSUMER OF  {searchPackage | modifyPackage}
    ID              id-contract-dap}
dapExecuteContract CONTRACT:: = {
    CONNECTION          dapConnectionPackage
    INITIATOR CONSUMER OF  {searchPackage | modifyPackage | executePackage}
    ID              id-contract-dapExecute}
-- connection package --
dapConnectionPackage CONNECTION-PACKAGE:: = {
    BIND        directoryBind
    UNBIND  in-directoryUnbind
    ID      id-package-dapConnection}
--  search and modify packages
searchPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES        {search}
    ID              id-package-search}
modifyPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES        {addEntry | removeEntry | modifyEntry}
    ID              id-package-modify}
executePackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES    {execute}
    ID          id-package-execute}
-- abstract-syntaxes --
inDirectoryOperationsAbstractSyntax ABSTRACT-SYNTAX:: = {
    BasicDAP-PDUs
    IDENTIFIED BY  id-as-indirectoryOperationsAS}
BasicDAP-PDUs:: = TCMessage {{DAP-Invokable},{DAP-Returnable}}
DAP-Invokable  OPERATION:: = {search | addEntry | removeEntry | modifyEntry}
DAP-Returnable OPERATION:: = {search | addEntry | removeEntry | modifyEntry}
inExtendedDirectoryOperationsAbstractSyntax ABSTRACT-SYNTAX:: = {
    Extended-BasicDAP-PDUs
    IDENTIFIED BY  id-as-inExtendedDirectoryOperationsAS}
Extended-BasicDAP-PDUs:: = TCMessage {{Extended-DAP-Invokable},{Extended-DAP-Returnable}}
Extended-DAP-Invokable  OPERATION:: = {search | addEntry | removeEntry | modifyEntry | execute}
Extended-DAP-Returnable OPERATION:: = {search | addEntry | removeEntry | modifyEntry | execute}
inDirectoryBindingAbstractSyntax   ABSTRACT-SYNTAX:: = {
    DAPBinding-PDUs
    IDENTIFIED BY  id-as-indirectoryBindingAS}
DAPBinding-PDUs:: = CHOICE {
    bind    Bind {directoryBind},
    unbind  Unbind {in-directoryUnbind}}
inSESEAbstractSyntax   ABSTRACT-SYNTAX:: = {
    SESEapdus {{spkmThreeWay},NoInvocationId}
    IDENTIFIED BY  {id-as-inSESEAS}}
END
```

9 SDF/SDF interface

9.1 Introduction to the IN ITU-T Recommendation X.500 DSP and Directory Information Shadowing Protocol (DISP) subset

The purpose of the SDF-SDF interface is to allow the transfer of copies of service profiles from one SDF to another and to manage the copies within the database network. The ITU-T Recommendation X.500 [36] functionalities cover more than the functionalities needed to fulfil the CS2 requirements. This subclause tries to indicate which aspects of the DSP and DISP should be considered and supported and which should be left out or ignored. Profiling is used as a means to present the status of the different parameters.

It is important to mention that the number of parameters carried in a message should be minimized, to reduce the load on the signalling traffic and processing time. This is the reason why the parameters are removed unless they are absolutely necessary when they are sent. On reception removed parameters should not be treated but should be understood by the receiving entity. This allows the extension of the profile in the future according to its actual description in the 1993 edition of the Directory.

For convenience and clarity this profile is defined using ASN.1 subtyping facilities however these definitions do not form a protocol specification. This simply indicates which parameters an implementation should not send. It does not change the behaviour of the receiving entity which shall still be capable of decoding values which conform to the original definition of the DSP and DISP. Nevertheless elements that are excluded by subtyping should be understood but not treated.

9.2 Working assumptions

Several assumptions were used to design the DSP and DISP for IN CS2. They are as follows:

Assumption 1: The agreements between network operators concerning the transfer of data are defined off-line (e.g. management operations). The establishOperationalBinding operation is only used to activate an agreement.

Assumption 2: The agreements cannot be modified by an online operation.

Assumption 3: The terminateOperationalBinding operation is used to end an agreement between two network operators. This means that the copy held by the shadow-consumer is no longer maintained. It should not be used and should be deleted. However the agreement could be required for future associations between the two networks, therefore this information should be retained.

Assumption 4: The shadow updates are initiated by the shadow supplier who holds the master copy. Therefore modifications of the copies are not performed on the shadowed copies but only on the master copy. The modification requests are passed to the master copy by using a chained operation. Copies are updated on changes.

Assumption 5: Only direct references are used in DSAs. Operations can only be chained once. If the operation cannot be fulfilled after one chaining, a referral should be sent back.

Assumption 6: It is not possible to make a copy of a copy. One should refer to the master copy to get a copy.

Assumption 7: The shadowing mechanism is initiated by a specific DAP operation or by a management operation. The management operation is for further study.

Assumption 8: The time when a shadowing agreement is terminated depends on the type of service. In most cases it will be based on the number of copies. Once the maximum number of copies is reached for a part of a DIT, then the oldest copy has to be deleted and its agreement de-activated. The maximum number of copies can be equal to one.

Assumption 9: An SDF-SDF opeyration cannot be abandoned. If an operation takes too much time, its timer expires and there is no need to abandon it.

9.3 The IN ITU-T Recommendation X.500 DISP subset

9.3.1 Shadowing Agreement specification

The Shadowing agreement is specified as:

```
IN-ShadowingAgreementInfo:: = ShadowingAgreementInfo (
    WITH COMPONENTS {
        ...,
        master          ABSENT,
        secondaryShadows        ABSENT})
```

shadowSubject specifies the subtree, entries and attributes to shadow. The components of **UnitOfReplication** are defined in subclause 9.2 of ITU-T Recommendation X.525 [42].

updateMode specifies when updates of a shadowed area are scheduled to occur. The components of **updateMode** are defined in subclause 9.3 of ITU-T Recommendation X.252 [35].

master contains the access point of the DSA containing the mastered area. "As this information is already known by the DSA it is not required for IN."

secondaryShadows permits secondary shadow information to be subsequently supplied to the shadow supplier. The secondary shadows are ignored in the IN context (assumption 5), then this component should not be included.

9.3.2 DSA Shadow Bind

A **dSAShadowBind** operation is used at the beginning of a period of providing shadows.

```
in-dSAShadowBind OPERATION:: = in-DirectoryBind
```

IN CS2 uses the in-DirectoryBind operation as specified in subclause 7.3.2.1 of ITU-T Recommendation Q.1228 [28].

9.3.3 IN-DSA Shadow Unbind

The **in-DSAShadowUnbind** operation replaces the ITU-T Recommendation X.525 [42] **dSAShadowUnbind** operation to provide class 4 operation behaviour for unbind procedures.

```
in-DSAShadowUnbind  OPERATION:: = inEmptyUnbind
```

9.3.4 Coordinate Shadow Update

The **inCoordinateShadowUpdate** operation is used by the shadow supplier to indicate the shadowing agreement for which it intends to send updates.

```
inCoordinateShadowUpdate  OPERATION:: = {
    ARGUMENT    IN-CoordinateShadowUpdateArgument
    RESULT      IN-CoordinateShadowUpdateResult
    ERRORS      {shadowError}
    CODE        id-opcode-coordinateShadowUpdate}
IN-CoordinateShadowUpdateArgument:: = CoordinateShadowUpdateArgument (
    WITH COMPONENTS {
        ...,
        updateStrategy (standard: {total | incremental})})
IN-CoordinateShadowUpdateResult    :: =    CoordinateShadowUpdateResult(
    WITH COMPONENTS {
        ...,
        null        PRESENT})
```

The various parameters have the meanings defined below:

a) The **agreementID** argument identifies the shadowing agreement.

b) The **lastUpdate** argument indicates the shadow supplier's understanding of the time at which the last update for this agreement was sent and is the time as provided by the shadow supplier DSA. This argument may only be omitted in the first instance of either a **inCoordinateShadowUpdate** or **inRequestShadowUpdate** operation for a particular shadowing agreement

c) The **updateStrategy** argument identifies the update strategy the shadow supplier intends to use for this update. For IN CS2, a total or incremental replacement strategy should be used. The "NoChanges" option will not be used.

d) The **securityParameters** argument is defined in subclause 7.10 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39].

9.3.5 Update Shadow

An **inUpdateShadow** operation is invoked by the shadow supplier to send updates to the shadow consumer for a unit of replication. Prior to this operation being initiated, a **inCoordinateShadowUpdate** or **inRequestShadowUpdate** operation must have been successfully completed for the identified shadowing agreement.

```
inUpdateShadow  OPERATION:: = {
    ARGUMENT    IN-UpdateShadowArgument
    RESULT      IN-UpdateShadowResult
    ERRORS      {shadowError}
    CODE        id-opcode-updateShadow}
IN-UpdateShadowArgument:: = UpdateShadowArgument (
    WITH COMPONENTS {
        ...,
        updatedInfo    (IN-RefreshInformation)})
IN-UpdateShadowResult  :: =    UpdateShadowUpdateResult(
    WITH COMPONENTS {
        ...,
        null        PRESENT})
```

The various parameters have the meanings as defined below:

a) The **agreementID** identifies the shadowing agreement that has been established.

b) The **updateTime** argument is supplied by the shadow supplier. This time is used during the next **inCoordinateShadowUpdate** or **inRequestShadowUpdate** to ensure that the shadow supplier and shadow consumer have a common view of the shadowed information.

c) The **updateWindow** argument, when present, indicates the next window during which the shadow supplier expects to send an update.

d) The **updatedInfo** argument provides the information required by the shadow consumer to update its shadowed information. The semantics of the information conveyed in this parameter shall result in the shadow consumer reflecting the changes supplied.

e) The **securityParameters** argument is defined in subclause 7.10 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39].

```
IN-RefreshInformation:: = RefreshInformation (
    WITH COMPONENTS {
        ...,
        otherStrategy      ABSENT})
```

The various parameters have the meanings as defined below:

a) **noRefresh** indicates that there have been no changes to the shadowed information from the previous instance to the present. This may be used where an **updateShadow** operation must be supplied at a certain interval defined in the shadowing agreement (**updateMode**), but no modification has actually occurred.

b) **total** provides a new instance of the shadowed information. The incremental strategy should be preferably used because it saves signalling.

c) **incremental** provides, instead of a complete replacement of the shadowed information, only the changes which have occurred to that shadowed information between **lastUpdate** in the most recent **inCoordinateShadowUpdate** (or **inRequestShadowUpdate**) request and **updateTime** in the current **inUpdateShadow** request (or **inRequestShadowUpdate** response).

d) **otherStrategy** provides the ability to send updates by mechanisms outside the scope of the Directory Specification. For IN CS2, either a total or incremental strategy should be used.

Should the request succeed, a result will be returned, although no information will be conveyed with it.

Should the request fail, a **shadowError** shall be reported. Circumstances under which the particular shadow problems will be returned are defined in ITU-T Recommendation X.525 [42] subclause 11.3.3.

9.3.6 Request Shadow Update

An **inRequestShadowUpdate** operation is used by the shadow consumer to request updates from the shadow supplier.

```
inRequestShadowUpdate  OPERATION:: = {
    ARGUMENT    IN-RequestShadowUpdateArgument
    RESULT      IN-RequestShadowUpdateResult
    ERRORS      {shadowError}
    CODE        id-opcode-RequestShadowUpdate}
IN-RequestShadowUpdateArgument:: = RequestShadowUpdateArgument (
    WITH COMPONENTS {
        ...,
        requestedStrategy  (standard: {incremental | total})})
IN-RequestShadowUpdateResult:: = RequestShadowUpdateResult(
    WITH COMPONENTS {
        ...,
        null        PRESENT})
```

The various parameters have the meanings as defined below:

a) The **agreementID** identifies the shadowing agreement.

b) The **lastUpdate** argument is the time provided by the shadow supplier in the most recent successful update. This argument may only be omitted in the first instance of either a **inCoordinateShadowUpdate** or **inRequestShadowUpdate** operation for a particular shadowing agreement.

c) The **requestedStrategy** argument identifies the type of update being requested by the shadow consumer. The shadow consumer may request either an **incremental** or a **total** update from the shadow supplier.

d) The **securityParameters** argument is defined in 7.10 of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39].

9.4 The IN ITU-T Recommendation X.500 DSP subset

9.4.1 Information types and common procedures

9.4.1.1 Chaining arguments

The **ChainingArguments** are present in each chained operation, to convey to a DSA the information needed to successfully perform its part of the overall task:

```
IN-ChainingArguments:: = ChainingArguments (
    WITH COMPONENTS {
        ...,
        aliasDereferenced      ABSENT,
        aliasedRDNs         ABSENT,
        returnCrossRefs     ABSENT,
        info                ABSENT,
        timeLimit           ABSENT,
        excludeShadows      ABSENT,
        nameResolveOnMaster ABSENT})
```

The various components have the meanings as defined below:

a) The **originator** component conveys the name of the originator of the request unless already specified in the security parameters. If **requester** is present in **CommonArguments**, this argument may be omitted.

b) The **targetObject** component conveys the name of the object whose directory entry is being routed to. The role of this object depends on the particular operation concerned: it may be the object whose entry is to be operated on, or which is to be the base object for a request or sub request involving multiple objects (e.g., **ChainedModify**). This component can be omitted only if it has the same value as the object or base object parameter in the chained operation, in which case its implied value is that value.

c) The **operationProgress** component is used to inform the DSA of the progress of the operation, and hence of the role which it is expected to play in its overall performance. Even though direct knowledge references are assumed, this parameter is deemed applicable for IN CS2 since an SDF to which an operation is chained can still respond with a continuation reference in the chained operation dsaReferral error.

d) The **traceInformation** component is used to prevent looping among DSAs when chaining is in operation. A DSA adds a new element to trace information prior to chaining an operation to another DSA. On being requested to perform an operation, a DSA checks, by examination of the trace information, that the operation has not formed a loop.

e) The **aliasDereferenced** component is a boolean value which is used to indicate whether or not one or more alias entries have so far been encountered and dereferenced during the course of distributed name resolution. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

f) The **aliasedRDNs** component indicates how many of the RDNs in the **targetObject** name have been generated from the **aliasedEntryName** attributes of one (or more) alias entries. The integer value is set whenever an alias entry is encountered and dereferenced. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

g) The **returnCrossRefs** component is a Boolean value which indicates whether or not knowledge references, used during the course of performing a distributed operation, are requested to be passed back to the initial DSA as cross references, along with a result or referral. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

h) The **referenceType** component indicates, to the DSA being asked to perform the operation, what type of knowledge was used to route the request to it. The DSA may therefore be able to detect errors in the knowledge held by the invoker. If such an error is detected it shall be indicated by a **ServiceError** with the **invalidReference** problem. **ReferenceType** is described fully in 8.4.1.3.

i) The **info** component is used to convey DMD-specific (Directory Management Domain) information among DSAs which are involved in the processing of a common request. As the management protocols are not addressed in CS2, this parameter is deemed to be not applicable.

j) The **timeLimit** component, if present, indicates the time by which the operation is to be completed. It is redundant with operation timers of TCAP and is therefore not needed.

k) The **SecurityParameters** component is specified in ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39].

l) The **entryOnly** component is set to **TRUE** if the original operation was a search, with the subset argument set to **oneLevel** and an alias entry was encountered as an immediate subordinate of the **baseObject**. The DSA which successfully performs name resolution on the **targetObject** name, shall perform object evaluation on only the named entry.

m) **AuthenticationLevel** is optionally supplied when it is required to indicate the manner in which authentication has been carried out between the SDFs. The **AuthenticationLevel** element is described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

n) **UniqueIdentifier** is optionally supplied when it is required to confirm the originator name (the originator is the SDF forwarding the request. The **UniqueIdentifier** element is described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

o) The **exclusions** component has significance only for Search operations; it indicates, if present, which subtrees of entries subordinate to the **targetObject** shall be excluded from the result of the Search operation.

p) The **excludeShadows** component has significance only for Search and List operations; it indicates that the search shall be applied to entries and not to entry copies. This optional component may be used by a DSA as one way to avoid the receipt of duplicate results. Since direct knowledge references are assumed, this parameter is deemed not applicable for CS 2.

q) The **nameResolveOnMaster** component only has significance during name resolution, and is only set if NSSRs (non-specific knowledge references) have been encountered. If set to **TRUE**, it signals that subsequent name resolution, i.e., matching the remaining RDNs from **nextRDNToBeResolved**, shall not employ entry copy information; subsequent resolution of each remaining RDN shall be done in the master DSA for the entry identified by that RDN (see subclause 20.1). Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

9.4.1.2 Chaining results

The **ChainingResults** are present in the result of each operation and provide feedback to the DSA which invoked the operation.

```
IN-ChainingResults:: = ChainingResults (
    WITH COMPONENTS {
        ...,
        info           ABSENT,
        crossReferences ABSENT })
```

The various components have the meanings as defined below:

a) The **info** component is used to convey DMD-specific information among DSAs which are involved in the processing of a common request. As the management protocols are not addressed in CS2, this parameter is deemed to be not applicable.

b) The **crossReferences** component is not present in the **ChainingResults** unless the **returnCrossRefs** component of the corresponding request had the value **TRUE**. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

c) The **SecurityParameters** component is specified in ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39]. Its absence is deemed equivalent to there being an empty set of security parameters.

d) The **alreadySearched** component, if present, indicates which subordinate RDNs immediately subordinate to the **targetObject** have been processed as a part of a chained Search operation and therefore shall be excluded in a subsequent subrequest.

9.4.1.3 Reference type

A **ReferenceType** value indicates one of the various kinds of reference defined in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

```
IN-ReferenceType:: = ReferenceType (1|2|4|5|6|7|8)
```

Value (3)(cross reference) is not applicable for IN CS2 as direct references are assumed.

9.4.1.4 Access Point information

There are three types of access points:

a) An **AccessPoint** value identifies a particular point at which access to the Directory, specifically to a DSA, can occur. The access point has a **Name**, that of the DSA concerned, and a **PresentationAddress**, to be used in SS7 signalling to that DSA.

```
IN-AccessPoint:: = AccessPoint (
    WITH COMPONENTS {
        ..,
        protocolInformation ABSENT})
```

The **address** contains the network address of the DSA in the SS7.

b) A **MasterOrShadowAccessPoint** value identifies an access point to the Directory. The **category**, either **master** or **shadow**, of the access point is dependent upon whether it points to a naming context or commonly useable replicated area.

```
IN-MasterOrShadowAccessPoint:: = MasterOrShadowAccessPoint (
    WITH COMPONENTS {
        ...,
        COMPONENTS OF IN-AccessPoint})
```

c) A **MasterAndShadowAccessPoints** value identifies a set of access points to the Directory, i.e., a set of related DSAs. These access points share the property that each refers to a DSA holding entry information from a common naming context (or a common set of naming contexts mastered in one DSA when the value is a value of the **nonSpecificKnowledge** attribute. A **MasterAndShadowAccessPoints** value indicates the **category** of each **AccessPoint** value it contains. The access point of the master DSA of the naming context need not be included in the set.

```
IN-MasterAndShadowAccessPoints :: = MasterOrShadowAccessPoint
```

An **AccessPointInformation** value identifies one or more access points to the Directory.

```
IN-AccessPointInformation:: = AccessPointInformation (
    WITH COMPONENTS {
        ...,
        COMPONENTS OF IN-MasterOrShadowAccessPoint })
```

9.4.1.5 Continuation reference

A **ContinuationReference** describes how the performance of all or part of an operation can be continued at a different DSA or DSAs. It is typically returned as a referral when the DSA involved is unable or unwilling to propagate the request itself.

```
IN-ContinuationReference   :: =    ContinuationReference (
    WITH COMPONENTS {
        ...,
        aliasedRDNs         ABSENT,
        rdnsResolved        ABSENT,
        referenceType       (IN-ReferenceType),
        accessPoints            SET OF (IN-AccessPoint))
```

The various components have the meanings as defined below:

a) The **targetObject** name indicates the name which is proposed to be used in continuing the operation. This might be different from the **targetObject** name received on the incoming request if, for example, an alias has been dereferenced, or the base object in a search has been located.

b) The **aliasedRDNs** component indicates how many (if any) of the RDNs in the target object name have been produced by dereferencing an alias. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

c) The **operationProgress** indicates the amount of name resolution which has been achieved, and which will govern the further performance of the operation by the DSAs named, should the DSA or DUA receiving the **ContinuationReference** wish to follow it up.

d) The **rdnsResolved** component value (which need only be present if some of the RDNs in the name have not been the subject of full name resolution, but have been assumed to be correct from a cross reference) indicates how many RDNs have actually been resolved, using internal references only. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

e) The **referenceType** component indicates what type of knowledge was used in generating this continuation.

f) The **accessPoints** component indicates the access points which are to be contacted to achieve this continuation. Only where non-specific subordinate references are involved can there be more than one **AccessPointInformation** item.

g) The **entryOnly** component is set to **TRUE** if the original operation was a search, with the **subset** argument set to **oneLevel**, and an alias entry was encountered as an immediate subordinate of the **baseObject**. The DSA which successfully performs name resolution on the **targetObject** name, shall perform object evaluation on only the named entry. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

h) The **exclusions** component identifies a set of subordinate naming contexts that should not be explored by the receiving DSA.

i) The **returnToDUA** element is optionally supplied when the DSA creating the continuation reference wishes to indicate that it is unwilling to return information via an intermediate DSA (e.g., for security reasons), and wishes to indicate that information may be directly available via an operation over DAP between the originating DUA and the DSA. When **returnToDUA** is set to **TRUE**, **referenceType** may be set to **self.** This element may be used in IN for support of the shadowing agreement established between network operators (e.g., SDF_V to SDF_h Modify may fail based upon access control restrictions).

j) The **nameResolveOnMaster** element is optionally supplied when the DSA creating the continuation reference has encountered NSSRs. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

9.4.2 DSA Bind

A **DSABind** operation is used to begin of a period of cooperation between two DSAs providing the Directory service.

```
dSABind OPERATION:: = in-DirectoryBind
```

IN CS2 uses the in-DirectoryBind operation as specified in subclause 7.3.2.1 of ITU-T Recommendation Q.1228 [28].

9.4.3 IN DSA Unbind

The **in-DSAUnbind** operation replaces the ITU-T Recommendation X.518 [40] **dSAUnbind** operation to provide class 4 operation behaviour for unbind procedures.

```
in-DSAUnbind  OPERATION:: = inEmptyUnbind
```

9.4.4 Chained operations

A DSA, having received an operation from a DUA, may elect to construct a chained form of that operation to propagate to another DSA. For IN CS2 a DSA, having received a chained form of an operation, must either process the operation or if the originating DSA is in another network, chain it to another DSA within the same network as the receiving DSA.

The DSA invoking a chained form of an operation may optionally sign the argument of the operation; the DSA performing the operation, if so requested, may sign the result of the operation.

The chained form of an operation is specified using the parameterized type **IN-chained {}**.

```
IN-chained { OPERATION: operation } OPERATION     :: = {
    ARGUMENT    OPTIONALLY-PROTECTED { SET {
        chainedArgument    (IN-ChainingArguments),
        argument       [0] operation.&ArgumentType },
        DIRQOP.&dspChainedOp-QOP@dirqop}
    RESULT  OPTIONALLY-PROTECTED { SET {
        IN-chainedResult       ABSENT,
        result         [0] operation.&ResultType },
        DIRQOP.&dspChainedOp-QOP@dirqop }
    ERRORS  { operation.&Errors EXCEPT (referral | dsaReferral) }
    CODE    operation.&code }
```

a) **IN-chainedArgument**. This is a value of **ChainingArguments** which contains that information, over and above the original DUA-supplied argument, which is needed in order for the performing DSA to carry out the operation.

b) **argument**. This is a value **operation.&Argument** and consists of the original DUA-supplied argument.

Should the request succeed, the result of the derived operation has the components:

a) **IN-chainedResult**. This is a value of **IN-ChainingResults** which contains that information, over and above that to be supplied to the originating DUA, which may be needed by the previous DSAs in a chain. For IN CS2, it is assumed that chains are not greater than length one, therefore the need of this parameter is not needed.

b) **result**. This is a value **operation.&Result** and consists of the result which is being returned by the performer of this operation, and which is intended to be passed back in the result to the originating DUA. This information is as specified in the appropriate clause of ITU-T Recommendation X.511 | ISO/IEC 9594-3 [39].

Should the request fail, one of the errors of the set **operation.&Errors** will be returned, except that **dsaReferral** is returned instead of **referral**.

9.4.5 Chained errors

The **dsaReferral** error is generated by a DSA when, for whatever reason, it doesn't wish to continue performing an operation by chaining the operation to another DSA. For IN CS2, DSAs may not chain operations incoming from another DSA unless the DSA is in another network.

```
IN-dsaReferral  ERROR:: = dsaReferral (
    WITH COMPONENTS {
        ...,
        reference      (IN-ContinuationReference),
        contextPrefix      ABSENT})
```

The various parameters have the meanings as described below:

a) The **IN-ContinuationReference** contains the information needed by the invoker to propagate an appropriate further request, perhaps to another DSA.

b) If the **returnCrossRefs** component of the ChainingArguments for this operation had the value **TRUE**, and the referral is being based upon a subordinate or cross-reference, then the **contextPrefix** parameter may optionally be included. The administrative authority of any DSA will decide which knowledge references, if any, can be returned in this manner (the others, for example, may be confidential to that DSA). Since direct knowledge references are assumed for IN CS2, this parameter is not applicable.

9.5 Protocol overview

9.5.1 ROS-objects and contracts

The interactions between DSAs generally required to provide the Directory Abstract Service in the presence of a distributed DIB are defined as a **indspContract**. A DSA that participates in this contract is defined as a ROS-object of class **dsp-dsa**. the contract is referred in this specifications as the DSA Abstract Service

```
dsp-dsa  ROS-OBJECT-CLASS:: = {
    BOTH    { indspContract}
    ID      id-rosObject-dspDSA}
```

The Shadow Abstract Service specifies the shadowing of information between a shadow supplier and a shadow consumer DSA. This service is manifested in two forms and therefore is defined as two distinct contracts. They are specified as a ROS-based information objects in subclause 8.5.2.

The **shadowConsumerContract** expresses the form of the service in which the shadow consumer, a ROS-object of class **initiating-consumer-dsa**, initiates the contract. A ROS-object of class **responding-supplier-dsa**, responds in this contract.

```
initiating-consumer-dsa  ROS-OBJECT-CLASS:: = {
    INITIATES   {shadowConsumerContract}
    ID      id-rosObject-initiatingConsumerDSA }
responding-supplier-dsa  ROS-OBJECT-CLASS:: = {
    RESPONDS    {shadowConsumerContract}
    ID      id-rosObject-respondingSupplierDSA }
```

The **shadowSupplierContract** expresses the form of the service in which the shadow supplier, a ROS-object of class **initiating-supplier-dsa**, initiates the contract. A ROS-object of class **responding-consumer-dsa**, responds in this contract.

```
initiating-supplier-dsa  ROS-OBJECT-CLASS:: = {
    INITIATES   {shadowSupplierContract}
    ID      id-rosObject-initiatingSupplierDSA }
responding-consumer-dsa  ROS-OBJECT-CLASS:: = {
    RESPONDS    {shadowSupplierContract}
    ID      id-rosObject-respondingConsumerDSA }
```

9.5.2 DSP contract and packages

The **indspContract** is defined as an information object of class CONTRACT

```
indspContract  CONTRACT:: = {
    CONNECTION          dspConnectionPackage
    INITIATOR CONSUMER OF  { inchainedModifyPackage | inchainedSearchPackage |
                           chainedExecutePackage }
    ID                  id-contract-indsp}
```

When a pair of DSAs from different open systems interact, this association contract is realized as an SS7 application layer protocol, referred to as the IN Directory System Protocol (DSP). The definition of this protocol in terms of an SS7 AC is provided in subclause 8.6 of the present document.

The indspContract is composed of a connection package, indspConnectionPackage and two operation packages, inchainedModifyPackage, inchainedSearchPackage and chainedExecutePackage.

The connection package, **indspConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE. It is identical to the connection package, **indapConnectionPackage**.

```
dspConnectionPackage  CONNECTION-PACKAGE:: = {
    BIND        dSABind
    UNBIND  in-DSAUnbind
    ID      id-package-dspConnection}
```

The operation packages **inchainedModifyPackage** and **inchainedSearchPackage** are defined as information objects of class OPERATION-PACKAGE. The operations of these packages are defined in ITU-T Recommendation X.518 [40].

```
inchainedModifyPackage  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {chainedAddEntry | chainedRemoveEntry | chainedModifyEntry}
    ID          id-package-inchainedModify}
inchainedSearchPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {chainedSearch}
    ID          id-package-inchainedSearch}
```

The operation packages **chainedExecutePackage** is defined an information objects of class OPERATION-PACKAGE.

```
chainedExecutePackage  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   { chainedExecute }
    ID          id-package-inchainedExecute}
```

In the **indspContract** either DSA may assume the role of initiator and invoke the operations of the contract.

9.5.3 DISP contract and packages

The **shadowConsumerContract** and **shadowSupplierContract** are defined as information objects of class CONTRACT.

```
shadowConsumerContract  CONTRACT:: = {
    CONNECTION          dispConnectionPackage
    INITIATOR CONSUMER OF  {shadowConsumerPackage}
    ID                  id-contract-shadowConsumer}
shadowSupplierContract  CONTRACT:: = {
    CONNECTION          dispConnectionPackage
    RESPONDER CONSUMER OF  {shadowSupplierPackage}
    ID                  id-contract-shadowSupplier}
```

The SS7 realization of the two forms of Shadow Abstract Service, referred to as the IN DISP are defined in terms of several SS7 ACs provided in subclause 8.6 of the present document.

The **shadowConsumerContract** and **shadowSupplierContract** are composed of a common connection package, **dispConnectionPackage** and one operation package, either **ShadowConsumerPackage** in the first case or **shadowSupplierPackage** in the second.

The connection package, **dispConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE. It is identical to the connection package, **dapConnectionPackage**.

```
dispConnectionPackage  CONNECTION-PACKAGE:: = {
    BIND        dSAShadowBind
    UNBIND  in-DSAShadowUnbind
    ID      id-package-dispConnection}
```

The operation packages **shadowConsumerPackage** and **shadowSupplierPackage** are defined as information objects of class OPERATION-PACKAGE. The operations of these packages are defined in ITU-T Recommendation X.525 [42].

```
shadowConsumerPackage  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestShadowUpdate}
    SUPPLIER INVOKES   {updateShadow}
    ID             id-package-shadowConsumer}
shadowSupplierPackage  OPERATION-PACKAGE:: = {
    SUPPLIER INVOKES   {coordinateShadowUpdate | updateShadow}
    ID             id-package-shadowSupplier}
```

Since the shadow consumer is the initiator of the **ShadowConsumerContract**, it assumes the role of consumer of the **shadowConsumerPackage**. This means that the shadow consumer invokes the **requestShadowUpdate** operation and that the shadow supplier invokes the **updateShadow** operation.

Since the shadow supplier is the initiator of the **shadowSupplierContract**, it assumes the role of supplier of the **shadowSupplierPackage**. This means that the shadow supplier invokes the operations of the contract.

9.6 Protocol abstract syntax

9.6.1 DSP abstract syntax

The Directory ASEs that realize the operation packages specified in subclause 8.5.2 share a single abstract syntax, **indirectorySystemAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectorySystemAbstractSyntax  ABSTRACT-SYNTAX:: = {
    BasicDSP-PDUs
    IDENTIFIED BY      id-as-indirectorySystemAS}
BasicDSP-PDUs:: = TCMessage {{DSP-Invokable},{DSP-Returnable}}
DSP-Invokable OPERATION:: = {chainedAddEntry | chainedRemoveEntry | chainedModifyEntry |
                      chainedSearch | chainedExecute }
DSP-Returnable OPERATION:: = {chainedAddEntry | chainedRemoveEntry | chainedModifyEntry |
                      chainedSearch | chainedExecute }
```

The realization of the connection package specified in 8.5.2 uses a separate abstract syntax, **indirectoryDSABindingAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectoryDSABindingAbstractSyntax  ABSTRACT-SYNTAX:: = {
    DSABinding-PDUs
    IDENTIFIED BY      id-as-indirectoryDSABindingAS}
DSABinding-PDUs:: = CHOICE {
    bind       Bind {dSABind},
    unbind  Unbind {in-DSAUnbind}}
```

9.6.2 DISP abstract syntax

The Directory ASEs that realize the operation packages specified in subclause 8.5.3 share the abstract syntax **inDirectoryShadowAbstractSyntax**. This abstract syntax is specified as an information object of the class **ABSTRACT-SYNTAX**.

```
inDirectoryShadowAbstractSyntax ABSTRACT-SYNTAX:: = {
    BasicDISP-PDUs
    IDENTIFIED BY      id-as-indirectoryShadowAS}
BasicDISP-PDUs:: = TCMessage {{DISP-Invokable},{DISP-Returnable}}
DISP-Invokable OPERATION:: ={requestShadowUpdate | updateShadow | coordinateShadowUpdate}
DISP-Returnable OPERATION:: ={requestShadowUpdate | updateShadow | coordinateShadowUpdate}
```

The realization of the connection package specified above uses a separate abstract syntax, **inDirectoryDSAShadowBindingAbstractSyntax**. This is specified as an information object of class ABSTRACT-SYNTAX.

```
inDirectoryDSAShadowBindingAbstractSyntax  ABSTRACT-SYNTAX:: = {
    DISPBinding-PDUs
    IDENTIFIED BY      id-as-indsaShadowBindingAS}
DISPBinding-PDUs:: = CHOICE {
    bind       Bind {dSAShadowBind},
    unbind  Unbind {in-DSAShadowUnbind}}
```

9.6.3 Directory System AC

The **indspContract** is realized as the **inDirectorySystemAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inDirectorySystemAC APPLICATION-CONTEXT:: =   {
    CONTRACT           dspContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectorySystemAbstractSyntax |
                           inDirectoryDSABindingAbstractSyntax}
    APPLICATION CONTEXT NAME id-ac-indirectorySystemAC}
```

If 3-way authentication is required then the **indspContract** is realized as the **inDirectorySystemWith3seAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inDirectorySystemWith3seAC APPLICATION-CONTEXT:: =    {
    CONTRACT           dspContract
    DIALOGUE MODE      structured
    TERMINATION            basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectorySystemAbstractSyntax |
                       inDirectoryDSABindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-indirectorySystemWith3seAC}
```

9.6.4 Directory Shadow AC

The **inshadowSupplierContract** is realized as the **inshadowSupplierInitiatedAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowSupplierInitiatedAC APPLICATION-CONTEXT:: = {
    CONTRACT           shadowSupplierContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectoryShadowAbstractSyntax |
                           inDirectoryDSAShadowBindingAbstractSyntax}
    APPLICATION CONTEXT NAME       id-ac-inShadowSupplierInitiatedAC}
```

If 3-way authentication is required then the **inshadowSupplierContract** is realized as the **inshadowSupplierInitiatedWith3seAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowSupplierInitiatedWith3seAC APPLICATION-CONTEXT:: = {
    CONTRACT                   shadowSupplierContract
    DIALOGUE MODE              structured
    TERMINATION                basic
    ADDITIONAL ASE         {id-se-threewayse}
    ABSTRACT SYNTAXES              {dialogue-abstract-syntax |
                                inDirectoryShadowAbstractSyntax |
                                inDirectoryDSAShadowBindingAbstractSyntax |
                                inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-inShadowSupplierInitiatedWith3seAC}
```

The **inshadowConsumerContract** is realized as the **inshadowConsumerInitiatedAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowConsumerInitiatedAC APPLICATION-CONTEXT:: = {
    CONTRACT           shadowConsumerContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectoryShadowAbstractSyntax |
                           inDirectoryDSAShadowBindingAbstractSyntax}
    APPLICATION CONTEXT NAME   id-ac-inShadowConsumerInitiatedAC}
```

If 3-way authentication is required then the **inshadowConsumerContract** is realized as the **inshadowConsumerInitiatedWithAC**. This AC is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowConsumerInitiatedWith3seAC APPLICATION-CONTEXT:: = {
    CONTRACT                  shadowConsumerContract
    DIALOGUE MODE             structured
    TERMINATION               basic
    ADDITIONAL ASE         {id-se-threewayse}
    ABSTRACT SYNTAXES             {dialogue-abstract-syntax |
                                inDirectoryShadowAbstractSyntax |
                                inDirectoryDSAShadowBindingAbstractSyntax |
                                inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME   id-ac-inShadowConsumerInitiatedWith3seAC}
```

9.6.5 Versions and the rules for extensibility

The Directory may be distributed and more than two Directory Application Entities may interoperate to service a request. The Directory AEs may be implemented conforming to different editions of the Directory specification of the Directory service which may or may not be represented by different protocol version numbers. The version number is negotiated to the highest common version number between two directly binding Directory AEs.

9.6.5.1 Version negotiation

When accepting an association, i.e., binding, utilizing the DSP or DISP, the version negotiated shall only affect the point to point aspects of the protocol exchanged between the initiating DSA and the responding DSA to which it is connected. Subsequent requests or responses on the dialogue shall be constrained by the version negotiated.

NOTE: There are no point to point aspects of the DSP or DISP that are currently indicated by different protocol versions.

9.6.5.2 Initiating DSA side

9.6.5.2.1 Request and response processing at the initiating DSA side

The initiating DSA may initiate requests using the highest edition of the specification of that request it supports. If one or more elements of the request are critical, it shall indicate the extension number(s) in the critical Extensions parameter.

NOTE: If the information the extension replaced in a CHOICE, ENUMERATED or INTEGER (used as ENUMERATED) type would be essential for proper operation in a responding DSA implemented according to an earlier edition of the Specification, it is recommended that the extension be marked critical.

When processing a response, a initiating DSA shall:

a) ignore all unknown bit name assignments within a bit string; and

b) ignore all unknown named numbers in an ENUMERATED type or INTEGER type that is being used in the enumerated style, provided the number occurs as an optional element of a SET or SEQUENCE; and

c) ignore all unknown elements in SETs, at the end of SEQUENCEs, or in CHOICEs where the CHOICE is itself an optional element of a SET or SEQUENCE.

NOTE: Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

d) not consider the receipt of unknown attribute types and attribute values as a protocol violation; and

e) optionally report the unknown attribute types and attribute values to the user.

9.6.5.2.2 Extensibility rules for error handling at the initiating DSA side

When processing a known error type with unknown indicated problems and parameters, a initiating DSA shall:

a) not consider the receipt of unknown indicated problems and parameters as a protocol violation (i.e., it shall not issue a TC-U-REJECT or abort the dialogue); and

b) optionally report the additional error information to the user.

When processing an unknown error type, a initiating DSA shall:

a) not consider the receipt of unknown error type as a protocol violation (i.e., it shall not issue a TC-U-REJECT or abort the application association); and

b) optionally report the error to the user.

9.6.5.3 Request processing at the responding DSA side

If any responding DSA performing an operation detects an element **criticalExtensions** whose semantic is unkown, it shall return an **unavailableCriticalExtension** indication as a **serviceError**.

NOTE 1: If a **criticalExtensions** string with one or more zero values is received, this indicates either that the extensions corresponding to the values are not present or are not critical. The presence of a zero value in a **criticalExtensions** string shall not be inferred as either the presence or absence of the corresponding extension in the APDU.

Otherwise, when processing a request from a initiating DSA, a responding DSA shall:

a) ignore all unknown bit name assignments within a bit string; and

b) ignore all unknown named numbers in an ENUMERATED type or INTEGER type that is being used in the enumerated style, provided the number occurs as an optional element of a SET or SEQUENCE; and

c) ignore all unknown elements in SETs, at the end of SEQUENCEs, or in CHOICEs where the CHOICE is itself an optional element of a SET or SEQUENCE.

NOTE 2: Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

9.7 Conformance

For the conformance of SDFs, the following statements should be added to the list of already existing statements.

9.7.1 Conformance by SDFs

9.7.1.1 Statement requirements

The following shall be stated:

a) the application-context for which conformance is claimed. The present version of the present document requires conformance to the **inDirectorySytemAC** application-context;

NOTE: An AC shall not be divided except as stated herein; in particular, conformance shall not be claimed to particular operations.

b) if conformance is claimed to the **inDirectorySystemAC** application-context, whether or not the chained mode of operation is supported, as defined in ITU-T Recommendation X.518 [40];

c) the security-level(s) for which conformance is claimed (none, simple, strong);

d) the attribute types for which conformance is claimed and whether for attributes based on the syntax **DirectoryString,** conformance is claimed for the **UNIVERSAL STRING** choice;

e) the object classes, for which conformance is claimed;

f) whether conformance is claimed for collective attributes as defined in subclause 8.8 of ITU-T Recommendation X.501 [37] and subclauses 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 [39];

g) whether conformance is claimed for hierarchical attributes as defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 [39];

h) the operational attribute types defined in ITU-T Recommendation X.501 [37] and any other operational attribute types for which conformance is claimed;

i) whether conformance is claimed for return of alias names as described in 7.7.1 of ITU T ITU-T Recommendation X.511 [39];

j) whether conformance is claimed for indicating that returned entry information is complete, as described in subclause 7.7.6 of ITU-T Recommendation X.511 [39];

k) whether conformance is claimed for modifying the object class attribute to add and/or remove values identifying auxiliary object classes, as described in 11.3.2 of ITU-T Recommendation X.511 [39];

l) whether conformance is claimed to Basic Access Control;

m) whether conformance is claimed to Simplified Access Control;

n) the name bindings for which conformance is claimed;

o) whether the SDF is capable of administering collective attributes, as defined in ITU-T Recommendation X.501 [37];

p) whether conformance is claimed for attribute contexts.

9.7.1.2 Static requirements

An SDF shall:

a) have the capability of supporting the application-contexts for which conformance is claimed as defined by their abstract syntax in 8.6;

b) have the capability of supporting the information framework defined by its abstract syntax in ITU-T Recommendation X.501 [37];

c) conform to the minimal knowledge requirements defined in ITU-T Recommendation X.518 [40];

d) have the capability of supporting the attribute types for which conformance is claimed; as defined by their abstract syntaxes;

e) have the capability of supporting the object classes for which conformance is claimed, as defined by their abstract syntaxes;

f) conform to the extensions for which conformance was claimed in the previous subclause;

g) if conformance is claimed for collective attributes, have the capability of performing the related procedures defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 [39];

h) if conformance is claimed for hierarchical attributes, have the capability of performing the related procedures defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Recommendation X.511 [39];

i) have the capability of supporting the operational attribute types for which conformance is claimed;

j) if conformance is claimed to Basic Access Control, have the capability of holding ACI items that conform to the definitions of Basic Access Control;

k) if conformance is claimed to Simplified Access Control, have the capability of holding ACI items that conform to the definitions of Simplified Access Control.

9.7.1.3 Dynamic requirements

An SDF shall:

a) conform to the mapping onto used services defined in subclause 18.1.7 of the present document;

b) conform to the procedures for distributed operations of the Directory related to referrals, as defined in ITU^^T Recommendation X.518 [40];

c) if conformance to the **directorySystemAC** application-context, conform to the referral mode of interaction as defined in ITU-T Recommendation X.518 [40];

d) if conformance is claimed for the chained mode of interaction, conformance to the chained mode of interaction as defined in ITU-T Recommendation X.518 [40];

NOTE: Only in this case it is necessary for a DSA to be capable of invoking operations of the **directorySystemAC**.

e) conform to the rules of extensibility procedures defined in subclause 7.5.5;

f) if conformance is claimed to Basic Access Control, have the capability of protecting information within the SDF in accordance with the procedures of Basic Access Control;

g) if conformance is claimed to Simplified Access Control, have the capability of protecting information within the SDF in accordance with the procedures of Simplified Access Control.

9.7.2 Conformance by a shadow supplier

A SDF implementation claiming conformance to this Directory Specification in the role of shadow supplier shall satisfy the requirements specified below.

9.7.2.1 Statement requirements

The following shall be stated:

a) the AC(s) for which conformance is claimed as a shadow supplier: **inShadowSupplierInitiatedAC** and **inShadowConsumerInitiatedAC**;

b) the security-level(s) for which conformance is claimed (none, simple, strong);

c) to which degree the **UnitOfReplication** is supported. Specifically, which (if any) of the following optional features are supported:

- Entry filtering on **ObjectClass**;
- Selection/Exclusion of attributes via **AttributeSelection**;
- The inclusion of subordinate knowledge in the replicated area;
- The inclusion of extended knowledge in addition to subordinate knowledge.

9.7.2.2 Static requirements

A SDF shall:

a) have the capability of supporting the application-context(s) for which conformance is claimed as defined in their abstract syntax above;

b) provide support for modifyTimestamp and createTimestamp operational attributes.

9.7.2.3 Dynamic requirements

A SDF shall:

a) conform to the mapping onto used services defined above;

b) conform to the procedures of ITU-T Recommendation X.525 | ISO/IEC 9594-9 [42] as they relate to the DISP.

9.7.3 Conformance by a shadow consumer

A SDF implementation claiming conformance to this Directory Specification as a shadow consumer shall satisfy the requirements specified below:

9.7.3.1 Statement requirements

The following shall be stated:

a) the AC(s) for which conformance is claimed as a shadow supplier: **inShadowSupplierInitiatedAC** and **shadowConsumerInitiatedAC**;

b) the security-level(s) for which conformance is claimed (none, simple, strong);

c) whether the SDF supports shadowing of overlapping units of replication.

9.7.3.2 Static requirements

A SDF shall:

a) have the capability of supporting the application-context(s) for which conformance is claimed as defined in their abstract syntax in subclause 8.6;

b) provide support for modifyTimestamp and createTimestamp operational attributes if overlapping units of replication is supported;

c) provide support for the copyShallDo service control.

9.7.3.3 Dynamic requirements

A SDF shall:

a) conform to the mapping onto used services defined in subclause 18.1.7;

b) conform to the procedures of ITU-T Recommendation X.525 [42] as they relate to the DISP.

9.8 ASN.1 modules for the SDF-SDF interface

The following set of ASN.1 modules define the SDF-SDF interface for IN CS2. They contain all the modifications to the Directory specifications as required for the support of INs.

The modules also contain the definitions which are impacted by these modifications because they make use of a modified type.

9.8.1 IN-CS2-SDF-SDF-Protocol module

This subclause includes all of the ASN.1 type and value definitions contained in this Directory Specification, in the form of the ASN.1 module, "IN-CS2-SDF-SDF-Protocol".

```
IN-CS2-SDF-SDF-Protocol
    {  ccitt ITU-T Recommendation Q.1228 module(0) in-cs2-sdf-sdf-Protocol(18) version1(0) }

DEFINITIONS:: =

BEGIN
-- EXPORTS All --
 -- The types and values defined in this module are exported for use in the other ASN.1 modules
contained
-- within the Directory Specifications, and for the use of other applications which will use
them to access
-- Directory services. Other applications may use them for their own purposes, but this will not
constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
distributedOperations, directoryShadowAbstractService, dsp , protocolObjectIdentifiers
    FROM  UsefulDefinitions ds-UsefulDefinitions
ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, CONNECTION-PACKAGE,
Code, OPERATION
    FROM Remote-Operations-Information-Objects ros-InformationObjects
Bind{}, Unbind{}
    FROM Remote-Operations-Generic-ROS-PDUs ros-genericPDUs
TCMessage {}
    FROM TCAPMessages tc-Messages
APPLICATION-CONTEXT, dialogue-abstract-syntax
    FROM TC-Notation-Extensions tc-NotationExtensions
dSABind,
chainedSearch, chainedAddEntry, chainedRemoveEntry, chainedModifyEntry, chained{}
    FROM DistributedOperations distributedOperations
dSAShadowBind,
coordinateShadowUpdate, updateShadow, requestShadowUpdate
    FROM DirectoryShadowAbstractService directoryShadowAbstractService
execute
    FROM IN-CS2-SCF-SDF-Operations scf-sdf-Operations
inEmptyUnbind
    FROM IN-CS2-classes {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-classes(4)
version1(0)}
id-rosObject-dspDSA, id-rosObject-initiatingConsumerDSA, id-rosObject-respondingSupplierDSA,
id-rosObject-respondingConsumerDSA, id-rosObject-initiatingSupplierDSA,
id-contract-indsp, id-contract-shadowConsumer, id-contract-shadowSupplier,
id-package-dspConnection, id-package-inchainedModify, id-package-inchainedSearch, id-package-
chainedExecute,
id-package-dispConnection, id-package-shadowConsumer, id-package-shadowSupplier,
id-as-indirectorySystemAS, id-as-indirectoryDSABindingAS, id-as-indirectoryShadowAS,
id-as-indsaShadowBindingAS,
id-ac-indirectorySystemAC, id-ac-inShadowSupplierInitiatedAC, id-ac-inShadowConsumerInitiatedAC,
id-ac-inShadowSupplierInitiatedWith3seAC,id-ac-inShadowConsumerInitiatedWith3seAC,
id-ac-indirectorySystemWith3seAC,
ds-UsefulDefinitions, ros-InformationObjects, ros-genericPDUs, tc-Messages,
tc-NotationExtensions, scf-sdf-Operations, scf-sdf-Protocol
    FROM IN-CS2-object-identifiers
        { ccitt ITU-T Recommendation Q.1228 module(0) in-cs2-object-identifiers(17) version1(0)
}
inSESEAbstractSyntax
    FROM IN-CS2-SCF-SDF-Protocol scf-sdf-Protocol
id-se-threewayse
    FROM ProtocolObjectIdentifiers protocolObjectIdentifiers

dspContract
    FROM DirectorySystemProtocol dsp
;
dsp-dsa  ROS-OBJECT-CLASS:: = {
    BOTH        {indspContract}
    ID          id-rosObject-dspDSA}

initiating-consumer-dsa  ROS-OBJECT-CLASS:: = {
    INITIATES       {shadowConsumerContract}
    ID          id-rosObject-initiatingConsumerDSA }
```

```
responding-supplier-dsa  ROS-OBJECT-CLASS:: = {
    RESPONDS        {shadowConsumerContract}
    ID          id-rosObject-respondingSupplierDSA }

initiating-supplier-dsa  ROS-OBJECT-CLASS:: = {
    INITIATES       {shadowSupplierContract}
    ID          id-rosObject-initiatingSupplierDSA }

responding-consumer-dsa  ROS-OBJECT-CLASS:: = {
    RESPONDS        {shadowSupplierContract}
    ID          id-rosObject-respondingConsumerDSA }

indspContract  CONTRACT:: = {
    CONNECTION          dspConnectionPackage
    INITIATOR CONSUMER OF  { inchainedModifyPackage | inchainedSearchPackage |
                           chainedExecutePackage }
    ID                  id-contract-indsp}

dspConnectionPackage  CONNECTION-PACKAGE:: = {
    BIND        dSABind
    UNBIND  in-DSAUnbind
    ID      id-package-dspConnection}

in-DSAUnbind    OPERATION :: = inEmptyUnbind

inchainedModifyPackage  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {chainedAddEntry | chainedRemoveEntry | chainedModifyEntry}
    ID          id-package-inchainedModify}

inchainedSearchPackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {chainedSearch}
    ID          id-package-inchainedSearch}

chainedExecutePackage OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   { chainedExecute }
    ID          id-package-chainedExecute}

chainedExecute  OPERATION :: = chained { execute }

shadowConsumerContract  CONTRACT:: = {
    CONNECTION          dispConnectionPackage
    INITIATOR CONSUMER OF  {shadowConsumerPackage}
    ID                  id-contract-shadowConsumer}

shadowSupplierContract  CONTRACT:: = {
    CONNECTION          dispConnectionPackage
    RESPONDER CONSUMER OF  {shadowSupplierPackage}
    ID                  id-contract-shadowSupplier}

dispConnectionPackage  CONNECTION-PACKAGE:: = {
    BIND        dSAShadowBind
    UNBIND  in-DSAShadowUnbind
    ID      id-package-dispConnection}

in-DSAShadowUnbind OPERATION :: = inEmptyUnbind

shadowConsumerPackage  OPERATION-PACKAGE:: = {
    CONSUMER INVOKES   {requestShadowUpdate}
    SUPPLIER INVOKES   {updateShadow}
    ID              id-package-shadowConsumer}

shadowSupplierPackage  OPERATION-PACKAGE:: = {
    SUPPLIER INVOKES   {coordinateShadowUpdate | updateShadow}
    ID              id-package-shadowSupplier}

inDirectorySystemAbstractSyntax  ABSTRACT-SYNTAX:: = {
    BasicDSP-PDUs
    IDENTIFIED BY       id-as-indirectorySystemAS}

BasicDSP-PDUs:: = TCMessage {{DSP-Invokable},{DSP-Returnable}}

DSP-Invokable OPERATION:: = {chainedAddEntry | chainedRemoveEntry | chainedModifyEntry |
                    chainedSearch | chainedExecute }

DSP-Returnable OPERATION:: = {chainedAddEntry | chainedRemoveEntry | chainedModifyEntry |
                    chainedSearch | chainedExecute }
```

```
inDirectoryDSABindingAbstractSyntax  ABSTRACT-SYNTAX:: = {
    DSABinding-PDUs
    IDENTIFIED BY      id-as-indirectoryDSABindingAS}

DSABinding-PDUs:: = CHOICE {
    bind        Bind {dSABind},
    unbind  Unbind {in-DSAUnbind}}

inDirectoryShadowAbstractSyntax ABSTRACT-SYNTAX:: = {
    BasicDISP-PDUs
    IDENTIFIED BY      id-as-indirectoryShadowAS}

BasicDISP-PDUs:: = TCMessage {{DISP-Invokable},{DISP-Returnable}}

DISP-Invokable OPERATION:: ={requestShadowUpdate | updateShadow | coordinateShadowUpdate}

DISP-Returnable OPERATION:: ={requestShadowUpdate | updateShadow | coordinateShadowUpdate}

inDirectoryDSAShadowBindingAbstractSyntax  ABSTRACT-SYNTAX:: = {
    DISPBinding-PDUs
    IDENTIFIED BY      id-as-indsaShadowBindingAS}

DISPBinding-PDUs:: = CHOICE {
    bind        Bind {dSAShadowBind},
    unbind  Unbind {in-DSAShadowUnbind}}

inDirectorySystemAC APPLICATION-CONTEXT:: =   {
    CONTRACT           indspContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectorySystemAbstractSyntax |
                           inDirectoryDSABindingAbstractSyntax}
    APPLICATION CONTEXT NAME id-ac-indirectorySystemAC}

inDirectorySystemWith3seAC APPLICATION-CONTEXT:: =    {
    CONTRACT           dspContract
    DIALOGUE MODE      structured
    TERMINATION            basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectorySystemAbstractSyntax |
                       inDirectoryDSABindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME id-ac-indirectorySystemWith3seAC}

inshadowSupplierInitiatedAC APPLICATION-CONTEXT:: = {
    CONTRACT           shadowSupplierContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectoryShadowAbstractSyntax |
                           inDirectoryDSAShadowBindingAbstractSyntax}
    APPLICATION CONTEXT NAME       id-ac-inShadowSupplierInitiatedAC}

inshadowSupplierInitiatedWith3seAC APPLICATION-CONTEXT:: = {
    CONTRACT               shadowSupplierContract
    DIALOGUE MODE          structured
    TERMINATION            basic
    ADDITIONAL ASE         {id-se-threewayse}
    ABSTRACT SYNTAXES          {dialogue-abstract-syntax |
                             inDirectoryShadowAbstractSyntax |
                             inDirectoryDSAShadowBindingAbstractSyntax |
                             inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME       id-ac-inShadowSupplierInitiatedWith3seAC}

inshadowConsumerInitiatedAC APPLICATION-CONTEXT:: = {
    CONTRACT           shadowConsumerContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectoryShadowAbstractSyntax |
                           inDirectoryDSAShadowBindingAbstractSyntax}
    APPLICATION CONTEXT NAME id-ac-inShadowConsumerInitiatedAC}
```

```
inshadowConsumerInitiatedWith3seAC APPLICATION-CONTEXT:: = {
    CONTRACT           shadowConsumerContract
    DIALOGUE MODE      structured
    TERMINATION        basic
    ADDITIONAL ASE     {id-se-threewayse}
    ABSTRACT SYNTAXES      {dialogue-abstract-syntax |
                           inDirectoryShadowAbstractSyntax |
                           inDirectoryDSAShadowBindingAbstractSyntax |
                       inSESEAbstractSyntax }
    APPLICATION CONTEXT NAME id-ac-inShadowConsumerInitiatedWith3seAC}
END
```

10 SCF/SCF interface

10.1 SCF/SCF Operations and arguments

```
IN-CS2-SCF-SCF-ops-args {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-scf-ops-args
(13) version1(0)}

-- The profiling of Directory Operations Parameters for the SCF-SCF relationship is outside the
scope of
-- IN CS2. Optional parameters received but not used in the SCF-SCF case are ignored.
-- Appropriate parameters to be used should be establish via agreement ahead of time.
DEFINITIONS IMPLICIT TAGS : : =
BEGIN
IMPORTS
    OPERATION, Code, ERROR
FROM Remote-Operations-Information-Objects ros-InformationObjects

    SecurityParameters,
    Credentials,
    SecurityProblem,
    securityError
FROM DirectoryAbstractService directoryAbstractService

    OPTIONALLY-PROTECTED{}
FROM EnhancedSecurity enhancedSecurity

    PROTECTION-MAPPING
FROM Notation guls-Notation

    AccessPointInformation
FROM DistributedOperations distributedOperations

    opcode-establishChargingRecord,
    opcode-handlingInformationRequest,
    opcode-handlingInformationResult,
    opcode-networkCapability,
    opcode-notificationProvided,
    opcode-confirmedNotificationProvided,
    opcode-provideUserInformation,
    opcode-confirmedReportChargingInformation,
    opcode-reportChargingInformation,
    opcode-requestNotification
FROM IN-CS2-operationcodes operationcodes

    EXTENSION,
    PARAMETERS-BOUND,
    SupportedExtensions {}
FROM IN-CS2-classes

    AccountNumber,
    ActivableServices,         BearerCapabilities ,
    BearerCapability {},
    CallConditions {},
    CalledPartyNumber {},
    CallingPartyNumber {},
    CallingPartysCategory,
    CallRecord {},
    Carrier,
    Cause {},
    ChargingParameters {},
    Digits {},
```

```
    DisplayInformation {},
    ErrorTreatment,
    ExtensionField {},
    HighLayerCompatibilities,
    HighLayerCompatibility,
    InfoToSend {},
    InfoType,
    Integer4,
    InteractionStrategy,
    InvokableService,
    Language,
    LocationNumber {},
    Notification,
    NotificationInformation {},
    NumberMatch {},
    OriginalCalledPartyID {},
    ReceivedInformation {},
    RedirectingPartyID {},
    RedirectionInformation,
    RequestedNotifications {},
    RequestedType,
    RoutingAddress {},
    ScfAddress {},
    ScfID {},
    SubscriberId {},
    SupplementaryServices,
    ToneId,
    TraceInformation{},
    TraceItem{},
    UnavailableNetworkResource,
    UserCredit {},
    UserInfo {},
    UserInformation {},
    UserInteractionModes

FROM IN-CS2-datatypes datatypes

    improperCallerResponse,
    missingCustomerRecord,
    missingParameter,
    parameterOutOfRange,
    systemFailure,
    unexpectedComponentSequence,
    unexpectedDataValue,
    unexpectedParameter,
    chainingRefused
FROM IN-CS2-errortypes errortypes

    errcode-scfReferral,
    errcode-scfTaskRefused
FROM IN-CS2-errorcodes errorcodes


    AuthenticationLevel
FROM BasicAccessControl basicAccessControl

    SPKM-ERROR
FROM SpkmGssTokens spkmGssTokens

    activityTest
FROM IN-CS2-SSF-SCF-ops-args ssf-scf-Operations

    ros-InformationObjects, ds-UsefulDefinitions,  operationcodes,
    classes, guls-Notation, guls-SecurityTransformations, errortypes, errorcodes,
    scf-scf-Protocol, ssf-scf-Operations, datatypes, spkmGssTokens
FROM IN-CS2-object-identifiers {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-
identifiers(17) version1(0)}

    directoryAbstractService, enhancedSecurity, distributedOperations, basicAccessControl
FROM UsefulDefinitions ds-UsefulDefinitions
;
```

```
establishChargingRecord {PARAMETERS-BOUND : bound}  OPERATION : : = {
    ARGUMENT    EstablishChargingRecordArg {bound}
    RETURN RESULT      FALSE
    ERRORS      {missingCustomerRecord |
                    missingParameter |
                    systemFailure |
                    scfTaskRefused |
                    unexpectedComponentSequence |
                    unexpectedDataValue |
                    unexpectedParameter |
                    parameterOutOfRange|
                    securityError
                    }
    CODE    opcode-establishChargingRecord
    }
```

-- Direction: supporting SCF -> controlling SCF, Timer T_{ecr}

```
-- This operation is used by the supporting SCF to give charging information to the controlling
-- SCF so that it can charge the user (on-line charging included).

EstablishChargingRecordArg  {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED { SEQUENCE {
    userCredit  [0] UserCredit {bound} OPTIONAL,
    chargingParameters [1] ChargingParameters {bound} OPTIONAL,
    reportExpected  [2] BOOLEAN     DEFAULT TRUE,
    securityParameters [3] SecurityParameters OPTIONAL,
    extensions  [4] SEQUENCE SIZE (1..bound.&numOfExtensions)
                                OF                                      ExtensionField {bound}
    OPTIONAL,
    ...
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }

handlingInformationRequest {PARAMETERS-BOUND : bound}  OPERATION : : = {
    ARGUMENT    HandlingInformationRequestArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingCustomerRecord |
                    missingParameter |
                    parameterOutOfRange |
                    systemFailure |
                    scfTaskRefused |
                    unexpectedComponentSequence |
                    unexpectedDataValue |
                    unexpectedParameter |
                    securityError |
                    scfReferral
                    }
    LINKED  {handlingInformationResult {bound}}
    CODE    opcode-handlingInformationRequest
    }
```

-- Direction: controlling SCF -> supporting SCF (or IAF), Timer T_{hi}

```
-- This operation  may be used   request the  execution of an SLP
-- in the assisting SCF and to provide to the  assisting
-- SCF the context of the call so that it can help the  controlling SCF in the processing of the
call..

HandlingInformationRequestArg {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED {SEQUENCE {
    requestedType   [0] RequestedType               OPTIONAL,
    callingPartyNumber [1] CallingPartyNumber {bound}     OPTIONAL,
    locationNumber  [2] LocationNumber {bound}     OPTIONAL,
    calledPartyNumber   [3] CalledPartyNumber {bound}      OPTIONAL,
    dialledDigits       [4] Digits {bound}              OPTIONAL,
    redirectingPartyID      [5] RedirectingPartyID {bound}     OPTIONAL,
    redirectionInformation  [6] RedirectionInformation         OPTIONAL,
    originalCalledPartyID   [7] OriginalCalledPartyID {bound}      OPTIONAL,
    numberOfCallAttempts    [8] INTEGER (1..bound.&ub-nbCall)  OPTIONAL,
    highLayerCompatibility  [9] HighLayerCompatibility         OPTIONAL,
    bearerCapability        [10] BearerCapability {bound}      OPTIONAL,
    invokedSupplementaryService     [11] InvokableService          OPTIONAL,
    activeSupplementaryServices     [12] ActivableServices             OPTIONAL,
    causeOfLastCallFailure  [13] Cause {bound}              OPTIONAL,
    userInteractionModes    [14] UserInteractionModes      OPTIONAL,
    callingPartysCategory   [15] CallingPartysCategory         OPTIONAL,
    callingPartyBusinessGroupID     [16] OCTET STRING          OPTIONAL,
    securityParameters      [17] SecurityParameters            OPTIONAL,

    extensions          [18] SEQUENCE SIZE (1..bound.&numOfExtensions)
                                OF ExtensionField {bound}      OPTIONAL,
    ...
    },
```

```
    SCFQOP.&scfArgumentQOP{@scfqop}
    }


handlingInformationResult {PARAMETERS-BOUND : bound}  OPERATION : : = {
    ARGUMENT    HandlingInformationResultArg {bound}
    RETURN RESULT          FALSE
    ERRORS      { missingParameter |
                           systemFailure |
                           parameterOutOfRange |
                           unexpectedComponentSequence |
                           unexpectedDataValue |
                           unexpectedParameter |
                           securityError
                           }
    CODE        opcode-handlingInformationResult
    }
```

-- Direction: supporting SCF(or IAF) ->controlling SCF, Timer T_{hir}

```
-- This operation is used by the assisting SCF to send information to the  controlling SCF on
how
-- to process the call and to give conditions under which it should be involved in the call
-- processing.

HandlingInformationResultArg {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED {SEQUENCE {
    routingAddress  [0] RoutingAddress {bound}     OPTIONAL,
    highLayerCompatibility [1] HighLayerCompatibility     OPTIONAL,
    supplementaryServices  [2] SupplementaryServices          OPTIONAL,
    preferredLanguage  [3] Language               OPTIONAL,
    carrier     [4] Carrier                    OPTIONAL,
    callingPartyNumber [5] CallingPartyNumber {bound}         OPTIONAL,
    originalCalledPartyID  [6] OriginalCalledPartyID  {bound}     OPTIONAL,
    redirectingPartyID [7] RedirectingPartyID  {bound}        OPTIONAL,
    redirectionInformation [8] RedirectionInformation         OPTIONAL,
    callingPartysCategory  [9] CallingPartysCategory          OPTIONAL,
    securityParameters [10] SecurityParameters           OPTIONAL,
    extensions      [11] SEQUENCE SIZE (1..bound.&numOfExtensions)
                                 OF ExtensionField {bound}       OPTIONAL,
    ...
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }


networkCapability  {PARAMETERS-BOUND : bound}  OPERATION : : = {
    ARGUMENT    NetworkCapabilityArg {bound}
    RESULT      NetworkCapabilityResultArg {bound}
    ERRORS      {missingCustomerRecord |
                    missingParameter |
                    systemFailure |
                    scfTaskRefused |
                    unexpectedComponentSequence |
                    unexpectedDataValue |
                    unexpectedParameter |
                    securityError
                    }
    CODE        opcode-networkCapability
    }
```

-- Direction: supporting SCF ->controlling SCF, Timer T_{nc}

```
-- This operation is used by the supporting SCF to request from the controlling SCF which type
of
-- service it supports.
NetworkCapabilityArg  {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED { SEQUENCE {
    bearerCapabilities [0] BearerCapabilities             OPTIONAL,
    highLayerCompatibilities   [1] HighLayerCompatibilities       OPTIONAL,
    supplementaryServices  [2] SupplementaryServices          OPTIONAL,
    securityParameters [3] SecurityParameters         OPTIONAL,
    extensions          [4] SEQUENCE SIZE (1..bound.&numOfExtensions)
                         OF ExtensionField {bound}     OPTIONAL,
    ...
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }
```

```
NetworkCapabilityResultArg  {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED { SEQUENCE {
    bearerCapabilities [0] BearerCapabilities            OPTIONAL,
    highLayerCompatibilities   [1] HighLayerCompatibilities      OPTIONAL,
    supplementaryServices  [2] SupplementaryServices         OPTIONAL,
    securityParameters [3] SecurityParameters        OPTIONAL,
    extensions         [4] SEQUENCE SIZE (1..bound.&numOfExtensions)
                               OF ExtensionField {bound}    OPTIONAL,
    ...
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }

notificationProvided  {PARAMETERS-BOUND : bound}   OPERATION : : = {
    ARGUMENT       NotificationProvidedArg {bound}
    RETURN RESULT      FALSE
    ERRORS  {missingParameter |
                       systemFailure |
                       scfTaskRefused |
                       unexpectedComponentSequence |
                       unexpectedDataValue |
                       unexpectedParameter|
                       missingCustomerRecord |
                       parameterOutOfRange |
                       securityError
                       }
        CODE       opcode-notificationProvided
    }
```

-- Direction: controlling SCF -> supporting SCF(or IAF), Timer T_{qp}

```
-- This operation is used by the controlling SCF to request assistance from the assisting SCF
-- under specific call conditions specified prior to the sending of the operation or to notify
the
-- outcome of a previous intervention of the assisting SCF.

NotificationProvidedArg {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED { SEQUENCE {
    notification       [0] Notification,
    notificationInformation    [1] NotificationInformation {bound}       OPTIONAL,
    securityParameters [2] SecurityParameters            OPTIONAL,
    extensions                 [3] SEQUENCE SIZE (1..bound.&numOfExtensions)
                                OF ExtensionField {bound}        OPTIONAL,
    ...
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }

confirmedNotificationProvided {PARAMETERS-BOUND : bound} OPERATION : : = makeConfirm {
    notificationProvided{bound},
    opcode-confirmedNotificationProvided}
```

--Direction: controlling SCF ->supporting SCF , Timer T_{hinc}

```
provideUserInformation  {PARAMETERS-BOUND : bound}   OPERATION : : = {
    ARGUMENT    ProvideUserInformationArg {bound}
    RESULT      ProvideUserInformationResultArg {bound}
    ERRORS      {missingCustomerRecord |
                            missingParameter |
                            systemFailure |
                            scfTaskRefused |
                            unexpectedComponentSequence |
                            unexpectedDataValue |
                            unexpectedParameter |
                            improperCallerResponse |
                            parameterOutOfRange |
                            securityError
                            }
    CODE        opcode-provideUserInformation
    }
```

-- Direction: supporting SCF -> controlling SCF, Timer T_{pui}

```
-- This operation is used by the supporting SCF to request information from the user that can be
-- interrogated by the controlling SCF.
```

```
ProvideUserInformationArg  {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED { SEQUENCE {
    constraints     [0] CollectedInfo,
    infoToSend      [1] InformationToSend {bound},
    errorInfo       [2] InformationToSend {bound}          OPTIONAL,
    typeOfRequestedInfo[3] InfoType           DEFAULT numericString,
    numberOfAllowedRetries     [4] INTEGER (0.. 127)  DEFAULT 0,
    actions     [5] Actions                   OPTIONAL,
    preferredLanguage  [6] Language                  OPTIONAL,
    securityParameters [7] SecurityParameters            OPTIONAL,
    extensions      [8] SEQUENCE SIZE (1.. bound.&numOfExtensions)
                     OF ExtensionField {bound}        OPTIONAL,
    ...
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }


CollectedInfo : : = CHOICE {
    collectedDigits[0] CollectedDigits,
    iA5Information  [1] BOOLEAN
}


CollectedDigits : : = SEQUENCE {
    minimumNbOfDigits  [0] INTEGER (1.. 127)          DEFAULT 1,
    maximumNbOfDigits  [1] INTEGER (1.. 127) ,
    endOfReplyDigit[2] IA5String (SIZE (1) )          OPTIONAL,
    cancelDigit     [3] IA5String (SIZE (1) )         OPTIONAL,
    startDigit      [4] IA5String (SIZE (1) )         OPTIONAL,
    firstDigitTimeOut  [5] INTEGER (1.. 127)          OPTIONAL,
    interDigitTimeOut  [6] INTEGER (1.. 127)          OPTIONAL,
    errorTreatment  [7] ErrorTreatment     DEFAULT reportErrorToScf,
    interruptableAnnInd[8] BOOLEAN            DEFAULT TRUE,
    voiceInformation   [9] BOOLEAN            DEFAULT FALSE,
    voiceBack       [10] BOOLEAN              DEFAULT FALSE
    }


InformationToSend {PARAMETERS-BOUND} : : = CHOICE {
    inbandInfo      [0] InbandInfo,
    tone            [1] Tone,
    displayInformation [2] DisplayInformation{bound}
    }


InbandInfo : : = SEQUENCE {
    messageId       [0] MessageID,
    numberOfRepetitions[1] INTEGER (1..127)          OPTIONAL,
    duration        [2] INTEGER (1..32767)        OPTIONAL,
    interval        [3] INTEGER (1..32767)           OPTIONAL
    }


Tone : : = SEQUENCE {
    toneId          [0] Integer4,
    duration        [1] Integer4                  OPTIONAL
    }


Actions : : = ENUMERATED {
    play (0) ,
    playandcollect (1)
    }



MessageID : : = OBJECT IDENTIFIER


ProvideUserInformationResultArg   {PARAMETERS-BOUND : bound}
: : = OPTIONALLY-PROTECTED { SEQUENCE {
    userInformation[0] ReceivedInformation {bound},
    securityParameters [1] SecurityParameters         OPTIONAL,
    extensions      [1] SEQUENCE SIZE (1..bound.&numOfExtensions)
                     OF ExtensionField {bound}    OPTIONAL
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }
```

```
reportChargingInformation   {PARAMETERS-BOUND : bound} OPERATION : : = {
    ARGUMENT    ReportChargingInformationArg {bound}
    RETURN RESULT              FALSE
    ERRORS      {missingCustomerRecord |
                               missingParameter |
                               systemFailure |
                               scfTaskRefused |
                               unexpectedComponentSequence |
                               unexpectedDataValue |
                               unexpectedParameter |
                               parameterOutOfRange |
                               securityError
                               }
    CODE        opcode-reportChargingInformation
    }
```

-- Direction: controlling SCF -> supporting SCF, Timer T_{rci}

```
-- This operation is used to give to the assisting network charging information collected by the
-- controlling network.


ReportChargingInformationArg   {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED { SEQUENCE
{
    callRecord      [0] CallRecord {bound}         OPTIONAL,
    remainingUserCredit[1] UserCredit {bound}          OPTIONAL,
    uniqueCallID    [2] CallIdentifier                 OPTIONAL,
    accountNumber   [3] AccountNumber          OPTIONAL,
    securityParameters [4] SecurityParameters          OPTIONAL
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }

CallIdentifier : : = Integer4


confirmedReportChargingInformation {PARAMETERS-BOUND : bound} OPERATION : : = makeConfirm {
    reportChargingInformation{bound},
    opcode-confirmedReportChargingInformation
    }
```

--Direction: controlling SCF -> supporting SCF , Timer T_{rcic}

```
requestNotification    {PARAMETERS-BOUND : bound}  OPERATION : : = {
    ARGUMENT    RequestNotificationArg {bound}
    RETURN RESULT              FALSE
    ERRORS      {missingParameter|
                    systemFailure|
                    scfTaskRefused|
                    unexpectedComponentSequence|
                    unexpectedDataValue|
                    unexpectedParameter|
                    parameterOutOfRange|
                    missingCustomerRecord|
                    securityError
                    }
    CODE        opcode-requestNotification
    }
```

-- Direction: supporting SCF (or IAF) -> controlling SCF, Timer T_{rn}

```
-- This operation is used by the assisting SCF to request notification from the controlling SCF
-- under specific call conditions specified by this operation.


RequestNotificationArg    {PARAMETERS-BOUND : bound} : : = OPTIONALLY-PROTECTED {SEQUENCE {
    requestedNotifications [0] RequestedNotifications {bound},
    securityParameters [1] SecurityParameters            OPTIONAL
    },
    SCFQOP.&scfArgumentQOP{@scfqop}
    }



scfBind {PARAMETERS-BOUND : bound} OPERATION : : = {
    ARGUMENT        SCFBindArgument{bound}
    RESULT              SCFBindResult {bound}
    ERRORS      { scfBindFailure}
        }
```

```
-- Direction: controlling SCF -> assisting SCF (or IAF) Timer Tbi
-- This operation is used to establish a relationship between two SCFs. It sent by the
controlling SCF each time it
-- needs to initiate communications with another (supporting) SCF.

SCFBindArgument  {PARAMETERS-BOUND : bound} : : = SEQUENCE {
    agreementID     [0] AgreementID,
    originatingScfAddress  [1] ScfAddress {bound}            OPTIONAL,
-- absent in a chained operation request which crosses an international internetworking boundary
    credentials     [2] Credentials               OPTIONAL
    }


SCFBindResult  {PARAMETERS-BOUND : bound} : : = SEQUENCE {
    respondingScfAddress   [0] ScfAddress {bound}               OPTIONAL,
-- absent in a chained operation request which crosses an international internetworking boundary
    returnedCredentials[1] Credentials                  OPTIONAL
    }


AgreementID : : = OBJECT IDENTIFIER




scfUnbind OPERATION : : =  {
RETURN RESULT          FALSE

ALWAYS RESPONDS        FALSE
}
-- Direction: controlling SCF -> assisting SCF (or IAF)
-- The SCF Unbind operation is used by the controlling SCF to close the relationship with the
supporting SCF.

scfChained  {OPERATION : operation, PARAMETERS-BOUND : bound}  OPERATION : : = {
    ARGUMENT    OPTIONALLY-PROTECTED {SEQUENCE {
            chainedArgument ChainingArgument {bound },
            argument           [0] operation.&ArgumentType
                                          OPTIONAL
            },
            SCFQOP.&scfArgumentQOP{@scfqop}
            }
    RESULT  OPTIONALLY-PROTECTED {SEQUENCE  {
            chainedResult   ChainingResult {bound},
            result             [0]operation.&ResultType
                                          OPTIONAL
            },
            SCFQOP.&scfArgumentQOP{@scfqop}
            }
    ERRORS      {operation.&Errors |
                    chainingRefused |
                    securityError |
                    scfReferral
                    }
    CODE        operation.&operationCode
    }

ChainingArgument    {PARAMETERS-BOUND : bound} : : = SEQUENCE  {
    originatingSCF  [0] ScfID {bound},
    target          [1] SubscriberId {bound}   OPTIONAL,
    traceInformation   [2] TraceInformation{bound},
    scfAuthenticationLevel [3] AuthenticationLevel    DEFAULT  basicLevels : {level none},
    timeLimit       [4] UTCTime        OPTIONAL,
    securityParameters [5] SecurityParameters     OPTIONAL,
    extensions                 [6] SEQUENCE SIZE (1..bound.&numOfExtensions)
                                OF ExtensionField {bound} OPTIONAL,
    ...
    }

ChainingResult    {PARAMETERS-BOUND : bound} : : = SEQUENCE  {
    ultimateResponder  [0] ScfAddress {bound}         OPTIONAL,
    traceInformation   [1] TraceInformation{bound},
    securityParameters [2] SecurityParameters     OPTIONAL,
    extensions      [3] SEQUENCE SIZE (1..bound.&numOfExtensions)
                                OF ExtensionField {bound} OPTIONAL,
    ...
    }
```

```
makeConfirm {OPERATION: operation, Code: code} OPERATION : : = {
    &ArgumentType  operation.&ArgumentType       OPTIONAL,
    &argumentTypeOptional  operation.&argumentTypeOptional OPTIONAL,
    &ResultType NULL,
    &Errors operation.&Errors          OPTIONAL,
    &alwaysReturns  BOOLEAN TRUE,
    &operationCode  code}


chainedEstablishChargingRecord   {PARAMETERS-BOUND : bound}  OPERATION : : =
scfChained{establishChargingRecord{bound},bound}

chainedHandlingInformationRequest {PARAMETERS-BOUND : bound} OPERATION : : = scfChained
{handlingInformationRequest{bound},bound}

chainedHandlingInformationResult {PARAMETERS-BOUND : bound} OPERATION : : =
scfChained{handlingInformationResult{bound},bound}

chainedNetworkCapability {PARAMETERS-BOUND : bound}   OPERATION : : = scfChained
{networkCapability{bound},bound}

chainedNotificationProvided {PARAMETERS-BOUND : bound} OPERATION : : = scfChained
{notificationProvided{bound},bound}

chainedConfirmedNotificationProvided {PARAMETERS-BOUND : bound} OPERATION : : = scfChained
{confirmedNotificationProvided {bound},bound}

chainedProvideUserInformation {PARAMETERS-BOUND : bound} OPERATION : : = scfChained
{provideUserInformation{bound},bound}

chainedReportChargingInformation {PARAMETERS-BOUND : bound} OPERATION : : = scfChained
{reportChargingInformation{bound},bound}

chainedConfirmedReportChargingInformation {PARAMETERS-BOUND : bound} OPERATION : : =
scfChained{confirmedReportChargingInformation{bound}, bound}

chainedRequestNotification {PARAMETERS-BOUND : bound} OPERATION : : =
scfChained{requestNotification{bound}, bound}


SCFQOP : : = CLASS {
    &scfqop-id      OBJECT IDENTIFIER UNIQUE,
    &scfBindErrorQOP   PROTECTION-MAPPING,
    &scfErrorsQOP   PROTECTION-MAPPING,
    &scfArgumentQOP PROTECTION-MAPPING,
    &scfResultQOP   PROTECTION-MAPPING
    }
WITH SYNTAX {
    SCFQOP-ID       &scfqop-id,
    SCFBINDERROR-QOP   &scfBindErrorQOP,
    SCFERRORS-QOP   &scfErrorsQOP,
    SCFOPARG-QOP    &scfArgumentQOP,
    SCFOPRES-QOP    &scfResultQOP
    }


scfBindFailure ERROR : : = {
    PARAMETER   FailureReason
    }


FailureReason : : = CHOICE {
    systemFailure   [0] UnavailableNetworkResource,
    scfTaskRefused  [1] ScfTaskRefusedParameter,
    securityError   [2] SET {
            problem     [0] SecurityProblem,
            spkmInfo    [1] SPKM-ERROR
            }
    }

scfTaskRefused ERROR : : = {
    PARAMETER   ScfTaskRefusedParameter
    CODE        errcode-scfTaskRefused
    }
```

```
ScfTaskRefusedParameter : : = OPTIONALLY-PROTECTED { SEQUENCE {
        reason  ENUMERATED {
            generic(0),
            unobtainable (1),
            congestion(2)
            --other values FOR FURTHER STUDY
            },
        securityParameters [1] SecurityParameters        OPTIONAL
    },
    SCFQOP.&scfErrorsQOP{@scfqop}
    }

scfReferral ERROR : : = {
    PARAMETER   ReferralParameter
    CODE        errcode-scfReferral
    }


ReferralParameter : : = OPTIONALLY-PROTECTED {
    SEQUENCE {
        tryhere [0] AccessPointInformation,
        securityParameters [1] SecurityParameters        OPTIONAL
        },
    SCFQOP.&scfErrorsQOP{@scfqop}
    }


END
```

The following value ranges do apply for operation specific timers in INAP:
short: 1 - 10 s
medium: 1 - 60 s
long: 1 s - 30 minutes
for further study: For Further Study

Table 9-1 lists all operation timers and the value range for each timer. The definitive value for each operation timer may be network specific and has to be defined by the network operator.

Table 9-1: Operation timers and their value range

Operation Name	Timer	value range
EstablishChargingRecordn	T_{ecr}	short
HandlingInformationRequest	T_{hi}	short
HandlingInformationResult	T_{hir}	short
NetworkCapability	T_{nc}	short
NotificationProvided	T_{np}	short
ConfirmedNotificationProvided	T_{cnp}	short
ProvideUserInformation	T_{pui}	long
ReportChargingInformation	T_{rci}	short
ConfirmedReportChargingInformatio	Tcrci	short
RequestNotification	Trn	short
ScfBind	Tbi	medium

10.2 SCF/SCF contracts, packages and ACs

10.2.1 Protocol overview

The **scf-scfContract** expresses the form of the service in which the SCF, a ROS-object of class **scf-scf**, initiates the contract. A ROS-object of class **scf-scf** responds in this contract.

```
scf-scfContract  CONTRACT : : = {
    CONNECTION             scf-scfConnectionPackage{networkSpecificBoundSet}
    INITIATOR CONSUMER OF  {
                                   activityTestPackage |
                                   handlingInformationPackage
{networkSpecificBoundSet}
                                   }
    RESPONDER CONSUMER OF  {
                                   activityTestPackage |
                                   chargingInformationPackage

{networkSpecificBoundSet}|                         networkCapabilityPackage

{networkSpecificBoundSet}|                  notificationPackage

{networkSpecificBoundSet}|                  userInformationPackage

{networkSpecificBoundSet}                   }
    ID                             id-contract-scf-scf
    }
```

When two SCFs are located in different IN PE, this association contract shall be realized as an SS7 application layer protocol. The definition of this protocol in terms of an SS7 AC is provided in subclause 9.2.2 of this ITU-T Recommendation.

The **scf-scfContract** is composed of a connection package, **scf-scfConnectionPackage** and **six** operation packages, **handlingInformationPackage, notificationPackage, chargingInformationPackage, activityTestPackage, userInformationPackage** and **networkCapabilityPackage**.

The dsspContract is defined as an information object of class CONTRACT.

```
dsspContract  CONTRACT : : = {
    CONNECTION             dsspConnectionPackage {networkSpecificBoundSet}
    INITIATOR CONSUMER OF      {chainedSCFOperationPackage{networkSpecificBoundSet}}
    ID                                   id-contract-dssp
    }
```

When a pair of SCFs from different open systems interact, this association contract is realized as an SS7 application layer protocol, referred to as the IN Distributed SCF System Protocol (DSSP). The definition of this protocol in terms of an SS7 AC is provided in 9.2.2 of this ITU-T Recommendation.

The **dsspContract** is composed of a connection package, **dsspConnectionPackage** and one operation package, **chainedSCFOperationPackage**.

The connection package, **scf-scfConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE defined below.

```
scf-scfConnectionPackage {PARAMETERS-BOUND : bound} CONNECTION-PACKAGE : : = {
    BIND           scfBind{bound}
    UNBIND     scfUnbind
    RESPONDER UNBIND   FALSE
    ID                                  id-package-scf-scfConnection
    }
```

The connection package, **dsspConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE.

```
dsspConnectionPackage {PARAMETERS-BOUND : bound} CONNECTION-PACKAGE : : = {
    BIND           scfBind{bound}
    UNBIND     scfUnbind
    RESPONDER UNBIND   FALSE
    ID                             id-package-dsspConnection
    }
```

The operation packages**, handlingInformationPackage, notificationPackage, chargingInformationPackage, activityTestPackage, userInformationPackage and networkCapabilityPackage, are** defined as information objects of class OPERATION-PACKAGE.

The operations of these packages are defined in subclause 9.1.

```
handlingInformationPackage  {PARAMETERS-BOUND : bound}   OPERATION-PACKAGE : : = {
    CONSUMER INVOKES    {handlingInformationRequest {bound}}
    SUPPLIER INVOKES    {handlingInformationResult {bound}}
    ID                          id-package-handlingInformation
    }

notificationPackage  {PARAMETERS-BOUND : bound}   OPERATION-PACKAGE : : = {
    CONSUMER INVOKES    { requestNotification {bound}}
    SUPPLIER INVOKES    { notificationProvided {bound}| confirmedNotificationProvided }
    ID                          id-package-notification
    }

chargingInformationPackage  {PARAMETERS-BOUND : bound}   OPERATION-PACKAGE : : = {
    CONSUMER INVOKES    { establishChargingRecord {bound} }
    SUPPLIER INVOKES    {

    confirmedReportChargingInformation{bound} |
                            reportChargingInformation  {bound}
                            }
    ID                      id-package-chargingInformation}

userInformationPackage  {PARAMETERS-BOUND : bound} OPERATION-PACKAGE : : = {
    CONSUMER INVOKES    {provideUserInformation  {bound}  }
    ID                      id-package-userInformation
    }

networkCapabilityPackage  {PARAMETERS-BOUND : bound} OPERATION-PACKAGE : : = {
    CONSUMER INVOKES    { networkCapability  {bound}}
    ID                      id-package-networkCapability
    }
```

The operation package, chainedSCFOperationPackage is defined as information objects of class OPERATION-PACKAGE. The operations of this packages are defined in subclause 9.1.

```
chainedSCFOperationPackage {PARAMETERS-BOUND : bound} OPERATION-PACKAGE : : = {
    CONSUMER INVOKES       {
                            chainedHandlingInformationRequest  {bound} |
                            chainedNotificationProvided { bound}|
                            chainedConfirmedNotificationProvided {bound}|
                            chainedReportChargingInformation { bound}|
    chainedConfirmedReportChargingInformation{bound}         }
    SUPPLIER INVOKES    {
    chainedEstablishChargingRecord { bound}|
    chainedHandlingInformationResult { bound}|
    chainedNetworkCapability { bound}|
    chainedProvideUserInformation { bound}|
    chainedRequestNotification { bound}
    }
    ID                  id-package-chainedSCFOperations
    }
```

Abstract Syntax

This version of the INAP requires the support of two abstract syntaxes:

a) the abstract syntax of TC dialogue control PDUs, dialogue-abstract-syntax, which is needed to establish the dialogues between FEs and specified in ETS 300 287-1 [7];

b) the abstract syntax for conveying the PDUs for invoking the operations involved in the operation packages specified in subclause 9.2.2 and reporting their outcome.

The ASN.1 type from which the values of the last abstract syntax are derived is specified using the parameterized types TCMessage {} defined in ETS 300 287-1 [7].

All these abstract syntaxes shall (as a minimum) be encoded according to the Basic ASN.1 encoding rules with the restrictions listed in ETS 300 287-1 [7].

The SCF-SCF INAP ASEs that realize the operation packages and the connection package specified in subclause 9.2.2 share a single abstract syntax, **scf-scfOperationsAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
scf-scfOperationsAbstractSyntax  ABSTRACT-SYNTAX : : = {
    BasicSCF-SCF-PDUs
    IDENTIFIED BY  id-as-scf-scfOperationsAS}
BasicSCF-SCF-PDUs : : = TCMessage {{SCF-SCF-Invokable}, {SCF-SCF-Returnable}}
SCF-SCF-Invokable {PARAMETERS-BOUND} OPERATION : : = {
        activityTest |
        establishChargingRecord {bound}|
        confirmedNotificationProvided {bound}|
        confirmedReportChargingInformation {bound} |
        handlingInformationRequest {bound}|
        handlingInformationResult {bound}|
        networkCapability {bound}|
        notificationProvided {bound}|
        provideUserInformation {bound}|
        reportChargingInformation {bound}|
        requestNotification {bound}
    }
SCF-SCF-Returnable {PARAMETERS-BOUND}  OPERATION : : = {
        activityTest |
        establishChargingRecord {bound}|
        confirmedNotificationProvided {bound}|
        confirmedReportChargingInformation {bound}|
        handlingInformationRequest {bound}|
        handlingInformationResult {bound}|
        networkCapability {bound}|
        provideUserInformation {bound}|
        requestNotification {bound}
    }
```

The Distributed SCF ASEs that realize the operation specified in subclause 9.1 share a single abstract syntax, **distributedSCFSystemAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
distributedSCFSystemAbstractSyntax  ABSTRACT-SYNTAX : : = {
    BasicDSSP-PDUs
    IDENTIFIED BY  id-as-distributedSCFSystemAS}
BasicDSSP-PDUs : : = TCMessage {{DSSP-Invokable}, {DSSP-Returnable}}
DSSP-Invokable {PARAMETERS-BOUND : bound} OPERATION : : = {
        chainedHandlingInformationRequest {bound}|
        chainedNotificationProvided {bound}|
        chainedConfirmedNotificationProvided{bound} |
        chainedReportChargingInformation {bound}|
        chainedConfirmedReportChargingInformation {bound}

    }
DSSP-Returnable {PARAMETERS-BOUND : bound} OPERATION : : = {
        chainedHandlingInformationRequest {bound}|
        chainedConfirmedNotificationProvided{bound} |
        chainedConfirmedReportChargingInformation {bound}
    }
```

The realization of the connection package specified in subclause 9.2.2 uses a separate abstract syntax, **distributedSCFBindingAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
distributedSCFBindingAbstractSyntax  ABSTRACT-SYNTAX  : : = {
    SCF-SCFBinding-PDUs{networkSpecificBoundSet}
    IDENTIFIED BY  id-as-scf-scfBindingAS}
SCF-SCFBinding-PDUs{PARAMETERS-BOUND: bound} : : = CHOICE {
    bind    Bind {scfBind{bound}},
    unbind  Unbind {scfUnbind}
    }
```

SCF-SCF ACs

The scf-scfContract is realized by four ACs, scf-scfOperationsAC, distributedSCFSystemAC, scf-scfOperationWith3aeAC, and distributedSCFSystemWith3aeAC. These ACs are specified as information objects of the class APPLICATION-CONTEXT.

```
scf-scfOperationsAC  APPLICATION-CONTEXT : : = {
    CONTRACT    scf-scfContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |

                              distributedSCFBindingAbstractSyntax |

                              scf-scfOperationsAbstractSyntax }

    APPLICATION CONTEXT NAME   id-ac-scf-scfOperationsAC
    }


distributedSCFSystemAC  APPLICATION-CONTEXT : : = {
    CONTRACT    dsspContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |

                                    distributedSCFSystemAbstractSyntax |

                                    distributedSCFBindingAbstractSyntax}

    APPLICATION CONTEXT NAME   id-ac-distributedSCFSystemAC
    }

scf-scfOperationsWith3seAC  APPLICATION-CONTEXT : : = {
    CONTRACT    scf-scfContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ADDITIONAL ASE  {id-se-threewayse}
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
                                    distributedSCFBindingAbstractSyntax |
                                    scf-scfOperationsAbstractSyntax  |
                                    inSESEAbstractSyntax }

    APPLICATION CONTEXT NAME   id-ac-scf-scfOperationsWith3seAC
    }

distributedSCFSystemWith3seAC  APPLICATION-CONTEXT : : = {
    CONTRACT    dsspContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ADDITIONAL ASE  {id-se-threewayse}
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
                                    distributedSCFSystemAbstractSyntax |
                                    distributedSCFBindingAbstractSyntax |

                                    inSESEAbstractSyntax }

    APPLICATION CONTEXT NAME   id-ac-distributedSCFSystemWith3seAC
    }
```

10.2.2 ASN.1 modules

```
-- This section includes all of the ASN.1 type and value definitions contained in this SCF/SCF
Specification, in the
-- form of the ASN.1 module, " IN-CS2-SCF-SCF-pkgs-contracts-acs ".
IN-CS2-SCF-SCF-pkgs-contracts-acs {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-scf-scf-
pkgs-contracts-acs (14) version1(0)}

DEFINITIONS : : =
BEGIN

-- This module describes the operation-packages, contracts and application-contexts used
--  over the SCF-SCF interface.

IMPORTS

    PARAMETERS-BOUND,
    networkSpecificBoundSet
FROM IN-CS2-classes classes
```

```
    ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, CONNECTION-PACKAGE,         OPERATION
FROM Remote-Operations-Information-Objects ros-InformationObjects


    Bind{}, Unbind{}
FROM Remote-Operations-Generic-ROS-PDUs ros-genericPDUs

    TCMessage {}
FROM  TCAPMessages tc-Messages

    APPLICATION-CONTEXT, dialogue-abstract-syntax
FROM TC-Notation-Extensions tc-NotationExtensions
    establishChargingRecord {},
    confirmedReportChargingInformation{},
    confirmedNotificationProvided {},
    handlingInformationRequest {},
    handlingInformationResult {},
    networkCapability {},
    notificationProvided {},
    provideUserInformation {},
    reportChargingInformation {},
    requestNotification {},
    chainedHandlingInformationRequest {},
    chainedNotificationProvided {},
    chainedConfirmedNotificationProvided {},
    chainedReportChargingInformation {},
    chainedConfirmedReportChargingInformation{},
    chainedEstablishChargingRecord {},
    chainedHandlingInformationResult {},
    chainedNetworkCapability {},
    chainedProvideUserInformation {},
    chainedRequestNotification {},
    scfBind{},
    scfUnbind
FROM IN-CS2-SCF-SCF-ops-args scf-scf-Operations

    id-ac,
    id-rosObject,
    id-contract,
    id-package,
    id-as,
    id-ac-scf-scfOperationsAC,
    id-ac-distributedSCFSystemAC,
    id-ac-scf-scfOperationsWith3seAC,
    id-ac-distributedSCFSystemWith3seAC,
    id-contract-scf-scf,
    id-contract-dssp,
    id-package-dsspConnection,
    id-package-scf-scfConnection,
    id-package-handlingInformation,
    id-package-notification,
    id-package-chargingInformation,
    id-package-userInformation,
    id-package-networkCapability,
    id-package-chainedSCFOperations,
    id-as-scf-scfOperationsAS,
    id-as-distributedSCFSystemAS,
    id-as-scf-scfBindingAS,
    ds-UsefulDefinitions,
    classes,
    tc-Messages, tc-NotationExtensions,
    ros-InformationObjects, ros-genericPDUs,
    scf-scf-Operations, scf-sdf-Protocol,
    ssf-scf-Operations, ssf-scf-Protocol
FROM IN-CS2-object-identifiers {ccitt ITU-T Recommendation Q.1228 modules(0) in-cs2-object-
identifiers (17) version1(0)}

    activityTest
FROM IN-CS2-SSF-SCF-ops-args ssf-scf-Operations

    activityTestPackage
FROM IN-CS2-SSF-SCF-pkgs-contracts-acs ssf-scf-Protocol

    inSESEAbstractSyntax
FROM IN-CS2-SCF-SDF-Protocol scf-sdf-Protocol

    id-se-threewayse
FROM ProtocolObjectIdentifiers protocolObjectIdentifiers
```

```
    protocolObjectIdentifiers
FROM UsefulDefinitions ds-UsefulDefinitions
;


-- Application Contexts --

scf-scfOperationsAC  APPLICATION-CONTEXT : : = {
    CONTRACT    scf-scfContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
                                distributedSCFBindingAbstractSyntax |

                                scf-scfOperationsAbstractSyntax }

    APPLICATION CONTEXT NAME   id-ac-scf-scfOperationsAC
    }

distributedSCFSystemAC  APPLICATION-CONTEXT : : = {
    CONTRACT    dsspContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
                                distributedSCFSystemAbstractSyntax |
                                distributedSCFBindingAbstractSyntax}

    APPLICATION CONTEXT NAME   id-ac-distributedSCFSystemAC
    }

scf-scfOperationsWith3seAC  APPLICATION-CONTEXT : : = {
    CONTRACT    scf-scfContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ADDITIONAL ASE  {id-se-threewayse}
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
                                distributedSCFBindingAbstractSyntax |
                                scf-scfOperationsAbstractSyntax |
                                inSESEAbstractSyntax}

    APPLICATION CONTEXT NAME   id-ac-scf-scfOperationsWith3seAC
    }

distributedSCFSystemWith3seAC  APPLICATION-CONTEXT : : = {
    CONTRACT    dsspContract
    DIALOGUE MODE  structured
    TERMINATION basic
    ADDITIONAL ASE  {id-se-threewayse}
    ABSTRACT SYNTAXES  {dialogue-abstract-syntax |
                        distributedSCFSystemAbstractSyntax |
                        distributedSCFBindingAbstractSyntax |
                        inSESEAbstractSyntax }

    APPLICATION CONTEXT NAME   id-ac-distributedSCFSystemWith3seAC
    }

-- Contracts --

scf-scfContract  CONTRACT : : = {
    CONNECTION  scf-scfConnectionPackage{networkSpecificBoundSet}
    INITIATOR CONSUMER OF      {
                                activityTestPackage |
                                handlingInformationPackage
{networkSpecificBoundSet}
                                }
    RESPONDER CONSUMER OF      {
                                activityTestPackage |
                                chargingInformationPackage
{networkSpecificBoundSet}|
                                networkCapabilityPackage
{networkSpecificBoundSet}|
                                notificationPackage
{networkSpecificBoundSet}|
                                userInformationPackage
{networkSpecificBoundSet}
                                }
    ID                          id-contract-scf-scf
    }
```

```
dsspContract  CONTRACT : : = {
    CONNECTION  dsspConnectionPackage {networkSpecificBoundSet}
    INITIATOR CONSUMER OF  {chainedSCFOperationPackage{networkSpecificBoundSet}}
    ID                  id-contract-dssp
    }

-- Connection Package --

scf-scfConnectionPackage {PARAMETERS-BOUND : bound} CONNECTION-PACKAGE : : = {
    BIND        scfBind{bound}
    UNBIND  scfUnbind
    RESPONDER UNBIND   FALSE
    ID              id-package-scf-scfConnection
    }

dsspConnectionPackage {PARAMETERS-BOUND : bound} CONNECTION-PACKAGE : : = {
    BIND        scfBind{bound}
    UNBIND  scfUnbind
    RESPONDER UNBIND   FALSE
    ID              id-package-dsspConnection
    }

--  handlingInformation package --

handlingInformationPackage  {PARAMETERS-BOUND : bound}   OPERATION-PACKAGE : : = {
    CONSUMER INVOKES   {handlingInformationRequest {bound}}
    SUPPLIER INVOKES   {handlingInformationResult {bound}}
    ID                  id-package-handlingInformation
    }

--  notification package --

notificationPackage  {PARAMETERS-BOUND : bound}   OPERATION-PACKAGE : : = {
    CONSUMER INVOKES   { requestNotification {bound}}
    SUPPLIER INVOKES   { notificationProvided {bound}| confirmedNotificationProvided }
    ID              id-package-notification
    }

--  chargingInformation package --

chargingInformationPackage  {PARAMETERS-BOUND : bound}   OPERATION-PACKAGE : : = {
    CONSUMER INVOKES   { establishChargingRecord {bound} }
    SUPPLIER INVOKES   {
    confirmedReportChargingInformation{bound} |
                    reportChargingInformation  {bound}
                    }
    ID              id-package-chargingInformation}

--  userInformation package --

userInformationPackage  {PARAMETERS-BOUND : bound} OPERATION-PACKAGE : : = {
    CONSUMER INVOKES   {provideUserInformation  {bound}  }
    ID                  id-package-userInformation
    }

--  networkCapability package --

networkCapabilityPackage  {PARAMETERS-BOUND : bound} OPERATION-PACKAGE : : = {
    CONSUMER INVOKES   { networkCapability  {bound}}
    ID              id-package-networkCapability
    }

--  chainedSCFOperation package --

chainedSCFOperationPackage {PARAMETERS-BOUND : bound} OPERATION-PACKAGE : : = {
    CONSUMER INVOKES       {
                    chainedHandlingInformationRequest  {bound} |
                    chainedNotificationProvided { bound}|
                    chainedConfirmedNotificationProvided {bound}|
                    chainedReportChargingInformation { bound}|
    chainedConfirmedReportChargingInformation{bound}             }
    SUPPLIER INVOKES   {
    chainedEstablishChargingRecord { bound}|
    chainedHandlingInformationResult { bound}|
    chainedNetworkCapability { bound}|
    chainedProvideUserInformation { bound}|
    chainedRequestNotification { bound}
    }
```

```
    ID                      id-package-chainedSCFOperations
    }


-- abstract syntaxes --

scf-scfOperationsAbstractSyntax  ABSTRACT-SYNTAX : : = {
    BasicSCF-SCF-PDUs
    IDENTIFIED BY             id-as-scf-scfOperationsAS}

BasicSCF-SCF-PDUs : : = TCMessage {{SCF-SCF-Invokable}, {SCF-SCF-Returnable}}

SCF-SCF-Invokable {PARAMETERS-BOUND} OPERATION : : = {
                          activityTest |
                          establishChargingRecord {bound}|
                          confirmedNotificationProvided {bound}|
                          confirmedReportChargingInformation {bound} |
                          handlingInformationRequest {bound}|
                          handlingInformationResult {bound}|
                          networkCapability {bound}|
                          notificationProvided {bound}|
                          provideUserInformation {bound}|
                          reportChargingInformation {bound}|
                          requestNotification {bound}
    }

SCF-SCF-Returnable {PARAMETERS-BOUND}  OPERATION : : = {
                          activityTest |
                          establishChargingRecord {bound}|
                          confirmedNotificationProvided {bound}|
                          confirmedReportChargingInformation {bound}|
                          handlingInformationRequest {bound}|
                          handlingInformationResult {bound}|
                          networkCapability {bound}|
                          provideUserInformation {bound}|
                          requestNotification {bound}
    }


distributedSCFSystemAbstractSyntax  ABSTRACT-SYNTAX : : = {
    BasicDSSP-PDUs
    IDENTIFIED BY             id-as-distributedSCFSystemAS}

BasicDSSP-PDUs : : = TCMessage {{DSSP-Invokable}, {DSSP-Returnable}}

DSSP-Invokable {PARAMETERS-BOUND : bound} OPERATION : : = {
                          chainedHandlingInformationRequest {bound}|
                          chainedNotificationProvided {bound}|
                          chainedConfirmedNotificationProvided{bound} |
                          chainedReportChargingInformation {bound}|
                          chainedConfirmedReportChargingInformation {bound}

    }

DSSP-Returnable {PARAMETERS-BOUND : bound} OPERATION : : = {
                          chainedHandlingInformationRequest {bound}|
                          chainedConfirmedNotificationProvided{bound} |
                          chainedConfirmedReportChargingInformation {bound}
    }

distributedSCFBindingAbstractSyntax  ABSTRACT-SYNTAX  : : = {
    SCF-SCFBinding-PDUs{networkSpecificBoundSet}
    IDENTIFIED BY             id-as-scf-scfBindingAS}

SCF-SCFBinding-PDUs{PARAMETERS-BOUND: bound} : : = CHOICE {
    bind                      Bind {scfBind{bound}},
    unbind                Unbind {scfUnbind}
    }

END
```

11 SCF/CUSF interface

11.1 Operations and arguments

```
IN-CS2-SCF-CUSF-ops-args { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
CS2(20) modules(0) in-cs2-scf-cusf-ops-args (15) version1(0)}

DEFINITIONS IMPLICIT TAGS:: =
BEGIN
IMPORTS
    OPERATION
FROM Remote-Operations-Information-Objects ros-InformationObjects

    EXTENSION,
        PARAMETERS-BOUND,
        SupportedExtensions { }
FROM IN-CS2-classes classes


    opcode-initiateAssociation,
    opcode-releaseAssociation,
    opcode-requestReportBCUSMEvent,
    opcode-initialAssociationDP,
    opcode-connectAssociation,
    opcode-continueAssociation,
    opcode-eventReportBCUSM

FROM IN-CS2-operationcodes operationcodes

    BCUSMEvent,
    CalledPartyNumber {},
    Cause {},
    Duration,
    ExtensionField {},
    LegID,
    Message,
    OperationCode
FROM IN-CS2-datatypes datatypes
    missingCustomerRecord,
    missingParameter,
    parameterOutOfRange,
    systemFailure,
    taskRefused,
    unexpectedComponentSequence,
    unexpectedDataValue,
    unexpectedParameter,
    unknownLegID
FROM IN-CS2-errortypes errortypes

    activityTest,
-- Direction: SCF -> CUSF, Timer: Tat
-- This operation is used to check for the continued existence of a relationship between the SCF
-- and CUSF. If the relationship is still in existence, then the CUSF will respond. If no reply
is
-- received, then the SCF will assume that the CUSF has failed in some way and will take the
-- appropriate action.

    reportUTSI,
-- Direction: CUSF -> SCF. Timer: Trru
-- This operation is issued by the CUSF in the context of the USI feature.  It is used to report
the receipt
-- of a User to Service Information (UTSI) IE to the SCF.

    requestReportUTSI,
-- Direction: SCF -> CUSF. Timer: Trru
-- This operation is issued by the SCF in the context of the USI feature to request the CUSF to
monitor for
-- a User to Service Information (UTSI) information element, which are received from a user.

    sendSTUI,
-- Direction: SCF -> CUSF. Timer: Tss
-- This operation is issued by the SCF in the context of the USI feature. It is used to request
the CUSF
-- to send a Service to User Information (STUI) information element to the indicated user.
```

```
    resetTimer
-- Direction: SCF  ->  CUSF, Timer: Trt
-- This operation is used to request the CUSF to refresh an application timer in the CUSF.

FROM IN-CS2-SSF-SCF-ops-args { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) CS2(20) modules(0) in-cs2-ssf-scf-ops-args (5)  version1(0)}

    classes, operationcodes, ros-InformationObjects, datatypes,errortypes
FROM IN-CS2-object-identifiers
{{ ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) CS2(20) modules(0) in-
cs2-object-identifiers(17) version1(0)}

;

connectAssociation  {PARAMETERS-BOUND: bound}  OPERATION  :: = {
    ARGUMENT    ConnectAssociationArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
        parameterOutOfRange |
        systemFailure |
        taskRefused |
        unexpectedComponentSequence |
        unexpectedDataValue |
        unexpectedParameter
        }
    CODE    opcode-connectAssociation
    }
-- Direction: SCF->CUSF, Timer: Tcoa
-- This operation is used to request the CUSF to proceed with processing. Additional information
-- which shall be used in further connection establishment is provided by the SCF.

ConnectAssociationArg  {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    address [0] CalledPartyNumber {bound},
    extensions  [1] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
              ExtensionField {bound}   OPTIONAL,
    ...
    }

continueAssociation  {PARAMETERS-BOUND: bound}  OPERATION :: = {
    ARGUMENT    ContinueAssociationArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
        parameterOutOfRange |
        systemFailure |
        taskRefused |
        unexpectedComponentSequence |
        unexpectedDataValue |
        unexpectedParameter
        }
    CODE    opcode-continueAssociation
    }
-- Direction: SCF->CUSF, Timer: Tcona
-- This operation is used to request the CUSF to proceed with processing. Additional information
-- which is not related to further connection establishment may be  provided by the SCF.

ContinueAssociationArg  {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    extensions  [0] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
              ExtensionField {bound}   OPTIONAL,
    ...
    }

eventReportBCUSM {PARAMETERS-BOUND: bound}  OPERATION :: = {
    ARGUMENT    EventReportBCUSMArg {bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-eventReportBCUSM
    }
-- Direction: CUSF -> SCF, Timer: Terbce
-- This operation is used to notify the SCF of a call unrelated event previously requested by
the SCF
-- in a RequestReportBCUSMEvent operation.

EventReportBCUSMArg  {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    eventTypeBCUSM  [0] EventTypeBCUSM  OPTIONAL,
    eventSpecificInformationBCUSM  [1] EventSpecificInformationBCUSM  OPTIONAL,
    miscCallInfo    [2] MiscCallInfo    DEFAULT
            {messageType request},
    cUApplicationInd    [3] CUApplicationInd    OPTIONAL,
    legID   [4] LegID   OPTIONAL,
```

```
    extensions  [5] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
              ExtensionField {bound}   OPTIONAL,
    ...
    }

initialAssociationDP  {PARAMETERS-BOUND: bound}  OPERATION    :: = {
    ARGUMENT    InitialAssociationDPArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingCustomerRecord |
        missingParameter |
        parameterOutOfRange |
        systemFailure |
        taskRefused |
        unexpectedComponentSequence |
        unexpectedDataValue |
        unexpectedParameter
        }
    CODE    opcode-initialAssociationDP
    }
-- Direction: CUSF->SCF, Timer: Tiadp
-- This operation is used after detection of a TDP to initiate a call unrelated
association with the SCF.

InitialAssociationDPArg  {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    serviceKey  [0] ServiceKey,
    cUApplicationInd    [1] CUApplicationInd    OPTIONAL,
    miscCallInfo    [2] MiscCallIinfo   OPTIONAL,
    eventtypeBCUSM  [3] EventTypeBCUSM  OPTIONAL,   calledPartyNumber  [4] CalledPartyNumber
{bound} OPTIONAL,
    callingPartyNumber [5] CallingPartyNumber {bound} OPTIONAL,
    callingPartySubAddress [6] CallingPartySubAddress OPTIONAL,
    highLayerCompatibility [7] HighLayerCompatibility OPTIONAL,
    bearerCapability    [8] BearerCapability {bound}    OPTIONAL,
    uSIServiceIndicator[9] USIServiceIndicator {bound}    OPTIONAL,
    uSIInformation  [10]    USIInformation {bound} OPTIONAL,
    extensions  [11]    SEQUENCE SIZE(1..bound.&numOfExtensions) OF
              ExtensionField {bound}   OPTIONAL,
    ...
    }

initiateAssociation  {PARAMETERS-BOUND: bound}  OPERATION :: = {
    ARGUMENT    InitiateAssociationArg {bound}
    RETURN RESULT   TRUE
    ERRORS  {missingParameter |
        parameterOutOfRange |
        systemFailure |
        taskRefused |
        unexpectedComponentSequence |
        unexpectedDataValue |
        unexpectedParameter
        }
    CODE    opcode-initiateAssociation
    }
-- Direction: SCF->CUSF, Timer: Tia
-- This operation is used for allowing the SCF to initiate a call unrelated association with the
user.
-- The subsequent operations can be sent in the same TCAP message in the following order:
--  - the RequestReportBCUSMEvent operation if an answer from the CUSF is expected

InitiateAssociationArg  {PARAMETERS-BOUND: bound}:: = SEQUENCE {
    calledPartyNumber  [0] CalledPartyNumber {bound},
    extensions  [1] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
              ExtensionField {bound}   OPTIONAL,
    uSIServiceIndicator[2] USIServiceIndicator {bound}    OPTIONAL,
    uSIInformation  [3] USIInformation {bound} OPTIONAL,
    ...
    }
```

```
releaseAssociation  {PARAMETERS-BOUND: bound}  OPERATION:: = {
    ARGUMENT    ReleaseAssociationArg {bound}
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE
    CODE    opcode-releaseAssociation
    }
--  Direction: SCF->CUSF, Timer: Trel
--  This operation is used to indicate to the CUSF to release the existing association between
the user and the
--  network, during the BCUSM suspended at a DP.
ReleaseAssociationArg  {PARAMETERS-BOUND: bound}:: = Cause {bound}

requestReportBCUSMEvent  {PARAMETERS-BOUND: bound}  OPERATION:: = {
    ARGUMENT    RequestReportBCUSMEventArg {bound}
    RETURN RESULT   FALSE
    ERRORS  {missingParameter |
        parameterOutOfRange |
        systemFailure |
        taskRefused |
        unknownLegID |
        unexpectedComponentSequence |
        unexpectedDataValue |
        unexpectedParameter
        }
    CODE    opcode-requestReportBCUSMEvent
    }
-- Direction: SCF -> CUSF, Timer: Trrbce
-- This operation is used to request the CUSF to monitor a BCUSM DP.


RequestReportBCUSMEventArg  {PARAMETERS-BOUND: bound}:: = SEQUENCE{
    bcusmEvents [0] SEQUENCE SIZE(1..bound.&numOfBCUSMEvents) OF BCUSMEvent,
    monitorDuration [3] Duration    OPTIONAL,
    extensions  [4] SEQUENCE SIZE(1..bound.&numOfExtensions) OF
             ExtensionField {bound}    OPTIONAL,
    cUDPCriteria    [5] CUApplicationInd   OPTIONAL,
    legID   [6] LegID   OPTIONAL,
    ...
    }


END
```

The following value ranges do apply for operation specific timers in INAP:

short: 1 - 10 s

medium: 1 - 60 s

long: 1 s - 30 minutes

for further study: For Further Study

The table 10 lists all operation timers and the value range for each timer. The definitive value for each operation timer may be network specific and has to be defined by the network operator.

Table 10.1

Operation Name	Timer	value range
connectAssociation	Tcoa	short
continueAssociation	Tcona	short
eventReportBCUSM	Terbce	short
initialAssociationDP	Tiadp	short
initiateAssociation	Tia	short
releaseAssociation	Trel	short
requestReportBCUSMEvent	Trrbce	short

11.2 SCF/CUSF contracts, operation packages, and ACs

11.2.1 Protocol overview

The **cusf-scf-contract** expresses the form of the service in which the CUSF, a ROS-object of class **cusf**, initiates the contract. A ROS-object of class **scf** responds in this contract.

```
cusf-scf-contract  CONTRACT:: = {
        CONNECTION      emptyConnectionPackage
INITIATOR CONSUMER OF  {basic-cusf-scf-package {networkSpecificBoundSet}}
RESPONDER CONSUMER OF  {activityTestPackage}
ID      id-contract-cusf-scf}
```

The **scf-cusf-contract** expresses the form of the service in which the SCF, a ROS-object of class **scf**, initiates the contract. A ROS-object of class **cusf** responds in this contract.

```
scf-cusf-contract  CONTRACT:: = {
        CONNECTION      emptyConnectionPackage
INITIATOR CONSUMER OF  {basic-scf-cusf-package {networkSpecificBoundSet}| activityTestPackage}
ID      id-contract-scf-cusf}
```

The **cusf-scf-contract** is composed of an operation package, **basic-cusf-scf-package**.

The **scf-cusf-contract** is composed of an operation package, **basic-scf-cusf-package**.

These operation packages are defined as information objects of class OPERATION-PACKAGE. The operations of these packages are defined in subclause 10.1.

```
basic-cusf-scf-package  OPERATION-PACKAGE:: = {
CONSUMER INVOKES   {

    initialAssociationDP | eventReportBCUSM}
SUPPLIER INVOKES   { releaseAssociation |
    requestReportBCUSMEvent | connectAssociation |
    continueAssociation}
ID      id-package-basic-cusf-scf}
basic-scf-cusf-package  OPERATION-PACKAGE:: = {
CONSUMER INVOKES   {initiateAssociation |
    releaseAssociation | requestReportBCUSMEvent |
    connectAssociation | continueAssociation}
SUPPLIER INVOKES   {
    eventReportBCUSM}
ID      id-package-basic-scf-cusf}
```

Abstract Syntax

This version of the INAP requires the support of two types of abstract syntaxes:

a) the abstract syntax of TC dialogue control PDUs, **dialogue-abstract-syntax,** which is needed to establish the dialogue between FEs and specified in ETS 300 287-1 [7];

b) the abstract syntax for conveying the PDUs for invoking the operations involved in the operation packages specified as above and reporting their outcome.

The ASN.1 type from which the values of the last abstract syntax are derived is specified using the parameterized types **TCMessages{}** defined in ETS 300 287-1 [7].

All these abstract syntaxes shall (as a minimum) be encoded according to the Basic ASN.1 encoding rules with the restrictions listed in ETS 300 287-1 [7].

The CUSF-SCF INAP ASEs that realize the operation packages specified as above and the emptyConnectioPackage specified in subclause 4.5 share the following two abstract syntaxes. They are specified as information objects of the class ABSTRACT-SYNTAX.

```
cusf-scf-abstract-syntax  ABSTRACT-SYNTAX:: = {
BASIC-CUSF-SCF-PDUs
IDENTIFIED BY  id-as-basic-cusf-scf}
BASIC-CUSF-SCF-PDUs:: = TCMessage {{CUSF-SCF-Invokable}, {CUSF-SCF-Returnable}}
CUSF-SCF-Invokable  OPERATION:: = { activityTest| releaseAssocation {networkSpecificBoundSet} |
requestReportBCUSMEvent {networkSpecificBoundSet} | connectAssociation {networkSpecificBoundSet}
|
continueAssociation {networkSpecificBoundSet} | eventReportBCUSM {networkSpecificBoundSet} |
initialAssociationDP {networkSpecificBoundSet} |
reportUTSI {networkSpecificBoundSet} |
requestReportUTSI {networkSpecificBoundSet} |
sendSTUI {networkSpecificBoundSet} }
CUSF-SCF-Returnable OPERATION:: = { activityTest| requestReportBCUSMEvent
{networkSpecificBoundSet}  connectAssociation {networkSpecificBoundSet} |
continueAssociation {networkSpecificBoundSet} |
initialAssociationDP {networkSpecificBoundSet} |
requestReportUTSI {networkSpecificBoundSet} |
sendSTUI {networkSpecificBoundSet}
}
scf-cusf-abstract-syntax  ABSTRACT-SYNTAX:: = {
        BASIC-SCF-CUSF-PDUs
        IDENTIFIED BY  id-as-basic-scf-cusf}
BASIC-SCF-CUSF-PDUs:: = TCMessage {{SCF-CUSF-Invokable}, {SCF-CUSF-Returnable}}
SCF-CUSF-Invokable  OPERATION:: = { activityTest|releaseAssociation {networkSpecificBoundSet} |
requestReportBCUSMEvent {networkSpecificBoundSet} | initiateAssociation
{networkSpecificBoundSet} |
connectAssociation {networkSpecificBoundSet} |
continueAssociation {networkSpecificBoundSet} | eventReportBCUSM {networkSpecificBoundSet} |
reportUTSI {networkSpecificBoundSet} |
requestReportUTSI {networkSpecificBoundSet} |
sendSTUI {networkSpecificBoundSet} }
SCF-CUSF-Returnable OPERATION:: = { activityTestrequestReportBCUSMEvent
{networkSpecificBoundSet} |  initiateAssociation {networkSpecificBoundSet} |
connectAssociation {networkSpecificBoundSet} |
continueAssociation {networkSpecificBoundSet} |
requestReportUTSI {networkSpecificBoundSet} |
sendSTUI {networkSpecificBoundSet}
}
```

ACs

The **cusf-scf-contract** is realized by an AC, **cusf-scf-ac**, and the **scf-cusf-contract** is realized by an AC, **scf-cusf-ac**. These ACs are specified as information objects of the class APPLICATION-CONTEXT.

```
    cusf-scf-ac  APPLICATION-CONTEXT:: = {
        CONTRACT    cusf-scf-contract
        DIALOGUE MODE  structured
        TERMINATION basic
        ABSTRACT SYNTAXES  {dialogue-abstract-syntax | cusf-scf-abstract-syntax}
        APPLICATION CONTEXT NAME  id-ac-cusf-scf}
    scf-cusf-ac  APPLICATION-CONTEXT:: = {
        CONTRACT    scf-cusf-contract
        DIALOGUE MODE  structured
        TERMINATION basic
        ABSTRACT SYNTAXES  {dialogue-abstract-syntax | scf-cusf-abstract-syntax}
        APPLICATION CONTEXT NAME  id-ac-scf-cusf}
```

11.2.2 ASN.1 module

```
IN-CS2-SCF-CUSF-pkgs-contracts-acs  { ccitt(0) identified-organization(4) etsi(0) inDomain(1)
in-network(1) CS2(20) modules(0) in-cs2-scf-cusf-pkgs-contracts-acs (16) version1(0)}

DEFINITIONS:: =
BEGIN

-- This module describes the operation-packages, contracts and application-contexts used
-- over the SCF-CUSF interface.

IMPORTS

        emptyConnectionPackage,
    PARAMETERS-BOUND,
    networkSpecificBoundSet
FROM IN-CS2-classes classes
```

```
    CONTRACT, OPERATION-PACKAGE, OPERATION
FROM Remote-Operations-Information-Objects ros-InformationObjects

    TCMessage {}
        FROM TCAPMessages tc-Messages

    APPLICATION-CONTEXT, dialogue-abstract-syntax
FROM TC-Notation-Extensions tc-NotationExtensions


    releaseAssociation {},
    requestReportBCUSMEvent {},
    initiateAssociation {},
    connectAssociation,
    continueAssociation,
    eventReportBCUSM,
    initialAssociationDP
FROM IN-CS2-SCF-CUSF-ops-args scf-cusf-Operations

    id-ac-cusf-scf,
    id-ac-scf-cusf,
    id-contract-scf-cusf,
    id-contract-cusf-scf,
    id-package-basic-cusf-scf,
    id-package-basic-scf-cusf,
    id-as-basic-cusf-scf,
    id-as-basic-scf-cusf,
    classes, ros-InformationObjects, tc-Messages, scf-cusf-Operations, tc-NotationExtensions,
    ssf-scf-Protocol, ssf-scf-Operations
FROM IN-CS2-object-identifiers { ccitt(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) CS2(20) modules(0) in-cs2-object-identifiers (17) version1(0)}

    activityTestPackage,
    uSIHandlingPackage
FROM IN-CS2-SSF-SCF-pkgs-contracts-acs ssf-scf-Protocol

    activityTest,
    reportUTSI,
    requestReportUTSI,
    sendSTUI
FROM IN-CS2-SSF-SCF-ops-args ssf-scf-Operations
;

-- application contexts --

cusf-scf-ac               APPLICATION-CONTEXT:: =    {
    CONTRACT           cusf-scf-contract
    DIALOGUE MODE   structured
    TERMINATION     basic
    ABSTRACT SYNTAXES   {dialogue-abstract-syntax |
                        cusf-scf-abstract-syntax }
    APPLICATION CONTEXT NAME    id-ac-cusf-scf }


scf-cusf-ac               APPLICATION-CONTEXT:: =    {
    CONTRACT           scf-cusf-contract
    DIALOGUE MODE   structured
    TERMINATION     basic
    ABSTRACT SYNTAXES   {dialogue-abstract-syntax |
                        scf-cusf-abstract-syntax }
    APPLICATION CONTEXT NAME    id-ac-scf-cusf}

-- contracts --

cusf-scf-contract          CONTRACT    :: =
    {CONNECTION     emptyConnectionPackage
    INITIATOR CONSUMER OF   {basic-cusf-scf-package {networkSpecificBoundSet}}
    RESPONDER CONSUMER OF   {activityTestPackage |            uSIHandlingPackage
{networkSpecificBoundSet}   ID                 id-contract-scf-cusf }


scf-cusf-contract          CONTRACT    :: =
    {CONNECTION     emptyConnectionPackage
    INITIATOR CONSUMER OF   {basic-scf-cusf-package {networkSpecificBoundSet}|
                    activityTestPackage |
                    uSIHandlingPackage {networkSpecificBoundSet}}
    ID                  id-contract-cusf-scf}
```

```
-- basic cusf-scf package --

basic-cusf-scf-package     {PARAMETERS-BOUND: bound} OPERATION-PACKAGE   :: =
    {CONSUMER INVOKES  {
                eventReportBCUSM {bound} |
                initialAssociationDP {bound}}
    SUPPLIER INVOKES   {
                        releaseAssociation {bound}|
                        requestReportBCUSMEvent {bound} |
                        connectAssociation {bound} |
                        continueAssociation {bound} }
    ID                  id-package-basic-cusf-scf }

--  basic scf-cusf package --

basic-scf-cusf-package     {PARAMETERS-BOUND: bound} OPERATION-PACKAGE   :: =
    {CONSUMER INVOKES  {initiateAssociation {bound}|

                        releaseAssociation {bound}|
                        requestReportBCUSMEvent {bound} |
                        connectAssociation {bound} |
                        continueAssociation {bound} } }
    SUPPLIER INVOKES   {       eventReportBCUSM {bound} }
    ID                  id-package-basic-scf-cusf }


-- abstract syntaxes --

cusf-scf-abstract-syntax       ABSTRACT-SYNTAX :: =    {
    BASIC-CUSF-SCF-PDUs
    IDENTIFIED BY      id-as-basic-cusf-scf}

BASIC-CUSF-SCF-PDUs :: = TCMessage {{CUSF-SCF-Invokable},{CUSF-SCF-Returnable} }

CUSF-SCF-Invokable OPERATION   :: = {
                        activityTest|
                        releaseAssociation  {networkSpecificBoundSet}|
                        requestReportBCUSMEvent {networkSpecificBoundSet}  |
                        connectAssociation {networkSpecificBoundSet} |
                        continueAssociation {networkSpecificBoundSet} |
    eventReportBCUSM {networkSpecificBoundSet} |
                        initialAssociationDP {networkSpecificBoundSet} |
                        reportUTSI {networkSpecificBoundSet} |
                        requestReportUTSI {networkSpecificBoundSet} |
                        sendSTUI {networkSpecificBoundSet}
                        }

CUSF-SCF-Returnable    OPERATION   :: = {
                         activityTest|
                        requestReportBCUSMEvent {networkSpecificBoundSet} |
                        connectAssociation {networkSpecificBoundSet} |
                        continueAssociation {networkSpecificBoundSet} |
                        initialAssociationDP {networkSpecificBoundSet} |
                        requestReportUTSI {networkSpecificBoundSet} |
                        sendSTUI {networkSpecificBoundSet}
                        }


scf-cusf-abstract-syntax       ABSTRACT-SYNTAX :: =
    {BASIC-SCF-CUSF-PDUs
    IDENTIFIED BY      id-as-basic-scf-cusf}

BASIC-SCF-CUSF-PDUs :: = TCMessage {{SCF-CUSF-Invokable},{SCF-CUSF-Returnable} }


SCF-CUSF-Invokable OPERATION   :: =                                          {
            activityTest|
                        releaseAssociation  {networkSpecificBoundSet}|
                        requestReportBCUSMEvent {networkSpecificBoundSet} |
                        initiateAssociation  {networkSpecificBoundSet}|
                        connectAssociation {networkSpecificBoundSet} |
                        continueAssociation {networkSpecificBoundSet} |
    eventReportBCUSM {networkSpecificBoundSet} |
                        reportUTSI {networkSpecificBoundSet} |
                        requestReportUTSI {networkSpecificBoundSet} |
                        sendSTUI {networkSpecificBoundSet}
                        }
```

```
SCF-CUSF-Returnable    OPERATION   :: =                                                {
            activityTest|
                        requestReportBCUSMEvent {networkSpecificBoundSet} |
                        initiateAssociation  {networkSpecificBoundSet}|
                        connectAssociation {networkSpecificBoundSet} |
                        continueAssociation {networkSpecificBoundSet} |
                        requestReportUTSI {networkSpecificBoundSet} |
                        sendSTUI {networkSpecificBoundSet}
                        }

END
```

12 SSF AE procedures

12.1 General

This subclause provides the definition of the SSF AE procedures related to the SSP - SCP interface. The procedures are based on the use of SS7; other signalling systems can be used (e.g. DSS1 layer 3).

Capabilities not explicitly covered by these procedures may be supported in an implementation dependent manner in the SSP, while remaining in line with clause 2.

The AE, following the architecture defined in the ITU-T Recommendations Q.700 [19], Q.1400 [30] and in ETS 300 287-1 [7] includes TC and one or more ASEs called TC-users. The following subclauses define the TC-user ASE which interfaces with TC using the primitives specified in ETS 300 287-1 [7]; other signalling systems, such as DSS1 layer 3, may be used.

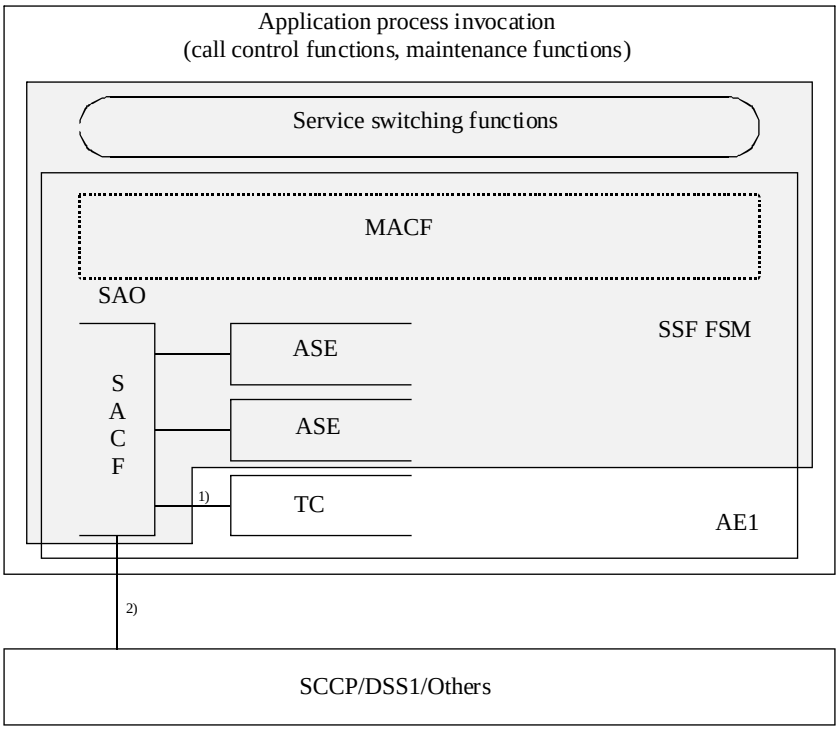
The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the AE procedures defined in the following differ from detailed procedures and the rules for using of services from TC, the statements and rules contained in the detailed clauses 17 and 18 shall be followed.

12.2 Model and interfaces

The functional model of the AE-SSF is shown in figure 11-1; the ASEs interface to TC to communicate with the SCF, and interface to the call control function (CCF) and the maintenance functions already defined for switching systems. The scope of the present document is limited to the shaded area in figure 11-1.

The interfaces shown in figure 11-1 use the TC-user ASE primitives specified in ETS 300 287-1 [7] (interface (1)) and N-Primitives specified in ETS 300 009-1 [3] (interface (2)). The operations and parameters of intelligent network application protocol (INAP) are defined in clause 17.



1) TC-Primitives.

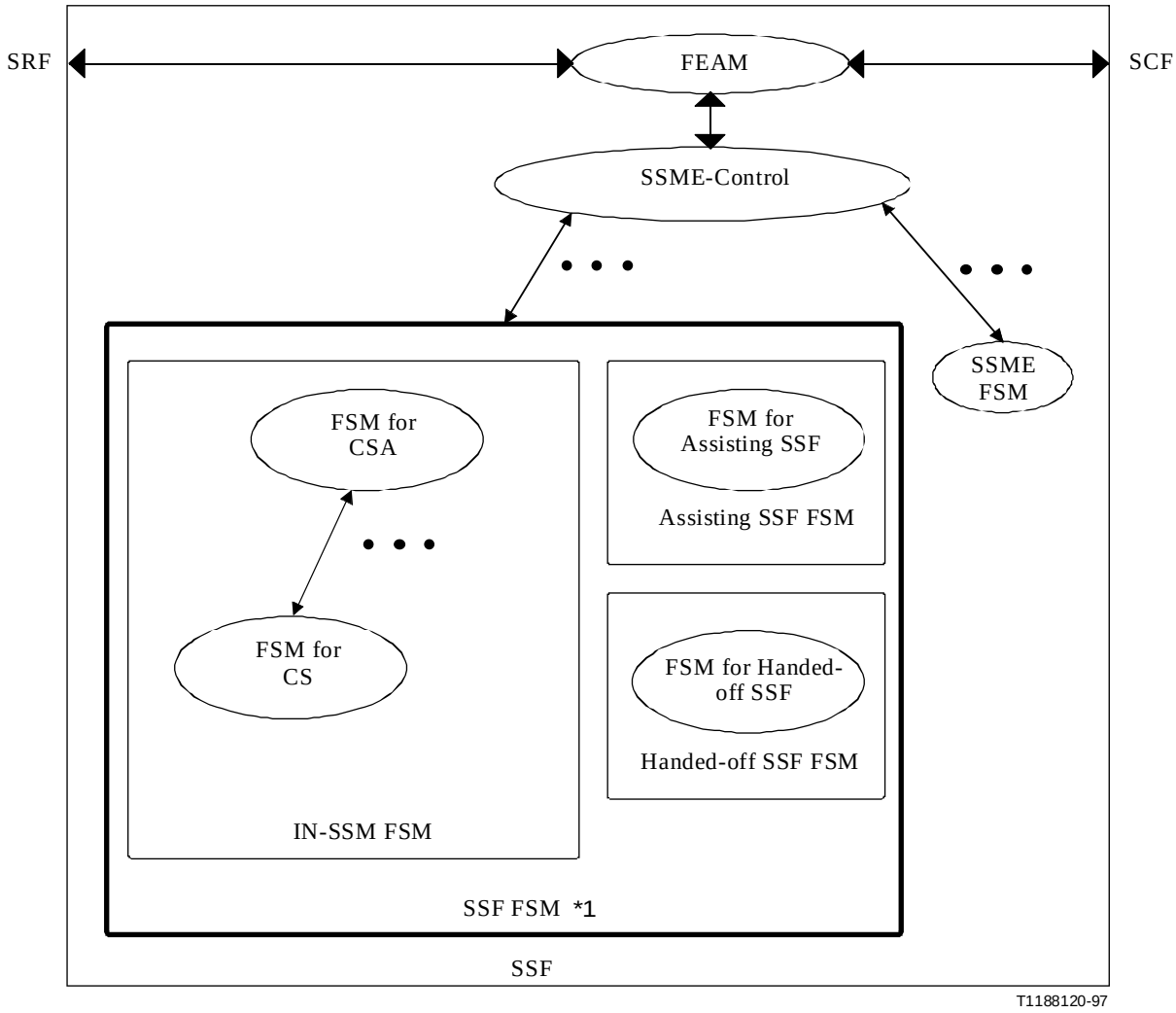
2) N-Primitives

AE1	Application entity invocation
SSF	Service switching functions
FSM	Finite state machine
MACF	Multiple assocation control function
SACF	Single association control function
SAO	Single association object

Figure 11-1: Functional model of SSF AE

12.3 Relations between SSF Finite State Model (FSM) and the CCF and maintenance functions

The primitive interface between the SSF FSM and the CCF/maintenance functions is an internal interface and is not subject to standardization in IN CS2. Nevertheless this interface should be in line with the BCSM defined in the clause 5 defining the Distributed Functional Plane (DFP) required for support of IN CS2 .



Note: One of the three possible FSM's (either IN SSM, Assisting SSF or Handed-off SSF) is selected.

CSA - Call Segment Association
CS - Call Segment

Figure 11-2: SSF Interfaces

An instance of an SSF FSM is either an IN Switching State Model (IN SSM) FSM, or an Assisting SSF FSM or an Handed-off SSF FSM (see figure 11-2).

The relationship between the BCSM and the SSF FSM is described as follows for the case of a call/attempt initiated by an end user, and the case of a call/attempt initiated by IN Service Logic (SL):

- When a call/attempt is initiated by an end user and processed at an exchange, a new instance of a BCSM is required. As the BCSM proceeds, it encounters detection points DPs. If a DP is armed as a Trigger Detection Point (TDP) an instance of an SSF FSM is required.

- If an InitiateCallAttempt is received from the SCF, an instance of a BCSM is created, as well as an instance of an SSF FSM.

The SSF logic shall:

- perform the DP processing actions including if DP criteria are met as specified in clause 5;
- check if traffic mechanisms are active;
- check for SCF accessibility;
- handle service feature interactions.

The SSF hands control back to the CCF at least in the following cases:

- if call gapping is in effect: the SSF logic instructs the CCF to terminate the call with the appropriate treatment;
- if service filtering is in effect: the call is counted (if required) and the SSF logic instructs the CCF to handle the call with the appropriate treatment;
- if a trigger (TDP) criteria match is not found (e.g., insufficient information to proceed): the SSF logic returns call control to the CCF;
- if the call is abandoned: the SSF logic returns call control to the CCF and continues processing as described in subclause 11.5;
- if the destination SCF is not accessible: the SSF logic instructs the CCF to route the call if possible (e.g. default routeing to a terminating announcement);
- if there is an existing control relationship for the call and a DP is encountered which is armed as an Trigger Detection Point - Request (TDP-R): the SSF returns call control to the CCF.

The management functions related to the execution of operations received from the SCF are executed by the SSF Management Entity (SSME). The SSME comprises a SSME-Control and several instances of SSME FSMs. The SSME-control interfaces the different SSF FSMs and SSME FSMs respectively and the Functional Entity Access Manager (FEAM). figure 11-2 shows the SSF interfaces.

The FEAM provides the low level interface maintenance functions including the following:

1 establishing and maintaining the interfaces to the SCF and SRF;

2 passing and queuing (when necessary) the messages received from the SCF and SRF to the SSME-control;

3 formatting, queuing (when necessary), and sending the messages received from the SSME-control to the SCF and SRF.

The SSME-control maintains the dialogues with the SCF, and SRF on behalf of all instances of the SSF FSM. These instances of the SSF FSM occur concurrently and asynchronously as calls occur, which explains the need for a single entity that performs the task of creation, invocation, and maintenance of the SSF FSMs. In particular the SSME-control performs the following tasks:

1 Interprets the input messages from other FEs and translates them into corresponding SSF FSM events;

2 Translates the SSF FSM outputs into corresponding messages to other FEs.

3 Captures asynchronous (with call processing) activities related to management or supervisory functions in the SSF and creates an instance of a SSME FSM. For example, the SSME provides call-unassociated treatment due to changes in Service Filtering or Call Gapping. Therefore, the SSME-control separates the SSF FSM from the Call Gapping and Service Filtering functions by creating instances of SSME FSMs for each context of management related operations.

The different contexts of the SSME FSMs may be distinguished based on the address information provided in the initiating operations. In the case of service filtering this address information is given by filteringCriteria, i.e. all ActivateServiceFiltering operations using the same address, address the same SSME-FSM handling this specific service filtering instance. For example ActivateServiceFiltering operations providing different filtering Criteria cause the invocation of new SSME-FSMs.

The SSF FSM passes call handling instructions to the related instances of the BCSM as needed. DPs may be dynamically armed as Event DPs, requiring the SSF FSM to remain active. At some point, further interaction with the SCF is not needed, and the SSF FSM may be terminated while the BCSM continues to handle the call as needed. A later TDP in the BCSM may result in a new instance of the SSF FSM for the same call.

Consistent with the single-ended control characteristic of IN service features for IN CS1, the SSF FSM only applies to a functionally separate call portion (e.g. the Originating BCSM or the Terminating BCSM (T-BCSM) in a two-party call, but not both).

12.4 SSF management (SSME) FSM

The SSME FSM state diagram is described in figure 11-3.

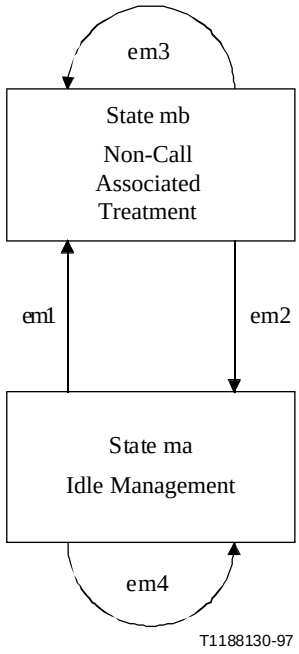


Figure 1-3: SSME FSM state diagram

The SSME FSM is independent of the individual SSF FSMs.

In the Idle Management state the following operations may be received from the SCF and processed by the SSME FSM with no resulting transition to a different state (transition em4):

ActivityTest
ManageTriggerData

The ActivityTest operation applies to call context transactions only.

The ManageTriggerData operation can only be received outside a call context transaction.

The Non-Call Associated Treatment state is entered from the Idle Management state when one of the following non-call associated operations is received (transition em1):

ActivateServiceFiltering

CallGap

The CallGap operation can be received inside as well as outside a call context transaction. The ActivateServiceFiltering operation can be received outside a call context transaction only.

During this state the following events can occur:

- given that service filtering is active, the SSF shall send a service filtering response to the SCF; the SSME FSM remains in this state (transition em3);
- given that service filtering is active, the SSF shall increment a counter; the SSME FSM remains in this state (transition em3);
- given that service filtering is active and the service filtering duration expires; the SSME shall send a **ServiceFilteringResponse** operation to the SCF; the SSME FSM moves to the Idle Management state (transition em2);
- if call gap related duration timer expires, the SSME FSM moves to the Idle Management state (transition em2);
- given that call gap/service filtering is active, another CallGap/ActivateServiceFiltering operation having the same gapping/filtering criteria can be received by the SSF: the second "filter" or "gap" replaces the first one (transition em3) unless the duration timer value is equal to zero, in which case the SSME FSM moves to the Idle Management state (transition em2) .

All other operations have no effect on the SSME-FSMs; the operations are passed by the SSME-Control to the relevant SSF FSM.

12.5 SSF switching state model (SSM) FSM

The SSM FSM consists of a FSM for a Call Segment Association (FSM for CSA). The FSM for CSA creates one or more sub FSMs for Call Segment (FSM for CS). For each created FSM for CSA instance there may be zero, one or many FSM for CS (sub FSMs) instances.

General rules and procedure principles for inclusion of Call Party Handling (CPH) capabilities into the SSF FSMs are addressed here.

- Timer treatment
  The use of a Timer to guard the SSF-SCF association (TC dialogue) or to prevent against excessive call suspension shall be done at the CS level
- The change of a Connection View (CV) is initiated either from the End User side (e.g. midcall DP) or from the SCF (SCF initiated CV change).
- The SCF may change the Connection View by sending of one of the following operations:

  Connect
  ContinueWithArgument
  DisconnectLeg
  InitiateCallAttempt
  SplitLeg
  MergeCallSegments
  MoveCallSegments
  MoveLeg
  ReleaseCall

- The SCF is informed about changes in the CV initiated by the SCF via CPH operations when ReturnResults are sent by the SSF on a successful change of the CV.
- The SCF can control a leg for which at least the Disconnect DP was armed. (Lifetime supervision of the leg.)
- The SCF shall have a connection view of the legs involved in the call. This is done by informing the SCF about the leg state changes, e.g. the disconnect of a leg. It shall not be allowed to have legs at a connection point which are not visible to the SCF (i.e. no DP armed).

- The number of operations sent by the SCF to resume call processing (i.e. when the FSM for CS is in any Waiting For Instructions state) shall be equal to the number of events that caused the suspension of the call process. Events that cause the suspension of the call processing are signalling events armed as TDP-Rs or Event Detection Point - Requests (EDP-Rs), or the processing of a CPH operation sent by the SCF. The FSM for CS shall keep track of the number of resumptions required per leg. The number of required resumptions will be incremented by 1 in case of a TDP-R or EDP-R. In case of a CPH operation the number of required resumptions will be set to 1 for not yet suspended legs associated with the involved CS for FSMs. The counters for already suspended legs (counters > 0) will, in case of a CPH operation, not be incremented. The call processing is resumed and the FSM for CS transits to a Monitoring or the Idle state only when all required resumptions for all legs within the call segment have been received.
  The processing of a ContinueWithArgument with csID or a Continue causes the number of required resumptions to be decremented by 1 for all legs within the call segment. The processing of a ContinueWithArgument with legID causes the number of resumptions required for the indicated leg to be decremented by 1.
  The processing of a Connect or CollectInformation causes the number of resumptions required for all legs within the call segment to be set to 0 and the call processing to be resumed.

CPH procedure principles for the FSM for CS:

- The import of a leg (including Event Detection Points (EDPs) or pending reports) in a CS (target CS) shall not affect the processing for other leg(s) residing in the same CS (target CS).

- The FSM for CS (one FSM per Call Segment and Connection Point) shall not know how many legs are connected to the CS. The FSM for CS exists as long as at least one report is pending or a DP is armed.

- In the states Waiting for End of User Interaction (WfI/Mon) and Waiting for End of Temporary Connection (WfI/Mon) where a SRF resource is connected to the CS, it is not allowed to change the Connection View via CPH operations.For any CS in the CSA for which user interaction applies it shall not be allowed to change the connection view via CPH operations. The following operations are not allowed:

  DisconnectLeg
  SplitLeg
  MoveLeg
  MergeCallSegments
  MoveCallSegments

- In the states Waiting for End of User Interaction (Wfi) and Waiting for End of Temporary Connection (WfI) where an SRF resource is connected to the CS and call processing is suspended , the only call processing operations allowed are:
  ContinueWithArgument
  Continue

- Per one CS only one connection is allowed to a resource (i.e. User Interaction or Temporary Connection).It implies that only one SRF resource can be connected to a CS at one time. The resource can be connected either to one leg or to the connection point in the CS.

- PromptAndCollectUserInformation and PromptAndReceiveMessage operations are restricted to be applied for a CS with only one leg

- All CPH operations received by the FSM for CS shall cause a transition to the Waiting For Instructions state.

- When the CPH operation SplitLeg is received in the Idle state for the target FSM for CS, the FSM for CS shall process the received operation and shall then transit to the Waiting for Instructions state.

- If one of the following CPH operations DisconnectLeg, SplitLeg, MoveLeg or MergeCallSegments is received in the Monitoring state the FSM for CS shall process the received CPH operation and then it shall transit to the Waiting for Instructions state (valid for SCF initiated changes of Connection view)

- The receipt of one of the above CPH operations in the state Waiting for Instructions shall not cause the change of the state Waiting for Instructions. Therefore all operation sequences shall be finalized by an operation which changes the state into Monitoring (e.g. ContinueWithArgument, Connect), except the case that a operation causes the transition to state Idle. (e.g. DisconnectLeg for the last leg)

Each FSM and the according states are discussed in the following subclauses. General rules applicable to more than one FSM/state are addressed here.

One or a sequence of components received in one or more TC messages may include a single operation or multiple operations, and is processed as follows:

- Process the operations in the order in which they are received.
- Each operation causes a state transition independent of whether or not a single operation or multiple operations are received in a message.
- The SSF examines subsequent operations in the sequence. As long as sequential execution of these operations leaves the FSM in the same state, it shall execute them (e.g. RequestReportBCSMEvent). If a subsequent operation causes a transition out of the state then the following operations shall be buffered until the current operation has been executed. In all other cases, await an event that causes a transition out of the current state (such an event is the completion of operation being executed, or the reception of an external event). An example of this is as follows:

  The SSF receives the operations FurnishChargingInformation, ConnectToResource, and PlayAnnouncement in a component sequence inside a single TC message. Upon receipt of this message, these operations are executed up to and including ConnectToResource while the SSF is in the Waiting For Instructions state. As the ConnectToResource operation is executed (and when, or after the FurnishChargingInformation operation has been completed), the SSF FSM will transit to the Waiting For End Of User Interaction state. The PlayAnnouncement operation is relayed to the SRF while the SSF is in Waiting For End Of User Interaction state.
- If there is an error in processing one of the operations in the sequence, the SSF FSM processes the error (see below) and discards all remaining operations in the sequence.
- If an operation is not understood or is out of context (i.e. violates the SACF rules defined by the SSF FSM) as described above, ABORT the interaction by sending of a TC-ABORT at the CSA level or an operation EntityReleased at the CS level.

In any state, if there is an error in a received operation, the maintenance functions are informed and the SSF FSM remains in the same state as when it received the erroneous operation; depending on the class of the operation, the error could be reported by the SSF to the SCF using the appropriate component (see ETS 300 287-1 [7]).

12.5.1 FSM for Call Segment Association (CSA)

Figure 11-4 shows the state diagram of the FSM for CSA in the SSF part of the SSP during the processing of an IN call/attempt.

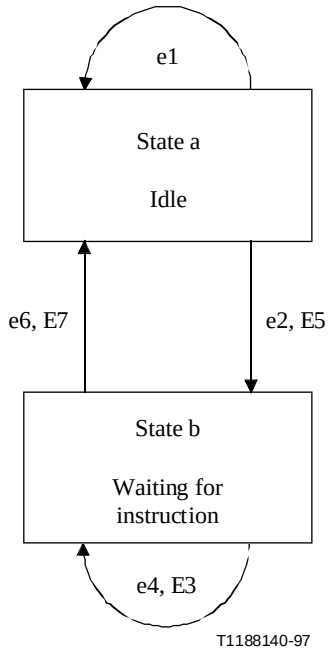


Figure 1-4: FSM for Call Segment Association

An instance of the FSM for CSA is created by SSME-Control when:

- an indication of a new call attempt is received from a user;
- a message related to a new transaction containing an InitiateCallAttempt or CreateCallSegmentAssociation operation is received from the SCF.

The FSM for CSA state diagram contains the following transitions (events):

e2 - TDP-R encountered;

E3 - any operation received from SCF that does not cause a transition to state Idle (includes InitiateCallAttempt);

e4 - EDP-R encountered;

- Event Detection Point - Notification (EDP-N) last encountered except last one from last CS;
- any response and report;

E5 - InitiateCallAttempt (received in state Idle);

- CreateCallSegmentAssociation received;

e6- - last EDP-N from last CS and no other reports pending;

E7 - any operation received from SCF which causes no remaining CS (e.g. ReleaseCall, MoveCallSegments or DisconnectLeg).

The CSA state diagram contains the following states:

State a Idle;

State b Active.

12.5.1.1 State a: Idle

The FSM for CSA enters the Idle state under a variety of conditions, as described below.

The FSM for CSA enters the Idle state when sending or receiving an TC-ABORT primitive due to abnormal conditions in the state Active.

The FSM for CSA enters the Idle state when one of the following occurs:

- when all the FSM for Call Segment instances associated with the FSM for CSA instance are released;

During this state the following call-associated events can occur:

- The following operations may be received from the FSM for CS, causing a state transition to the Active state (transition e2);
- an **InitialDP** is received indicating an encountered TDP-R. The received operation is sent to the SCF.

The rules for DP processing are described in clause 5.

- a message related to a new transaction containing an **InitiateCallAttempt** or a **CreateCallSegmentAssociation** operation is received from the SCF: in this case the FSM for CSA instance is created and moves via state Idle to the state Active (transition E5).

Any other operation received from the SCF while the FSM for CSA is in Idle state, i.e. while the SSF FSM instance does not exist, should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TC (refer to clause 18).

12.5.1.2 State b: Active

This state is entered from the Idle state by detecting a TDP-R (transition e2), from the Idle state on receipt at the SSF of a TC_Begin indication primitive containing an **InitiateCallAttempt** or a **CreateCallSegmentAssociation** operation from the SCF (transition E5).

In this state the FSM for CSA handles instructions from the SCF and events which are received from the FSMs for CS.

During this state the following events can occur:

- The receipt of an TC-END or TC-ABORT primitive has no effect on the call; the call may continue or be completed with the information available. In this case, the FSM for CSA transits to the Idle state (transition E7), disassociating the FSM for CSA from the call.
- An operation is received from the SCF: the FSM for CSA acts according to the operation received as described below.
- An operation is received from the FSM for CS: the FSM for CSA acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to the idle state (transition E3):

ApplyCharging
CallInformationRequest
Cancel(invokeID)
ConnectToResource
DisconnectForwardConnection
DisconnectForwardConnectionWithArgument
EstablishTemporaryConnection
FurnishChargingInformation

NOTE 1: It is network operator specific whether the operation is forwarded to any FSM for CS instance using the internal coding of the OCTET STRING or not.

InitiateCallAttempt

NOTE 2: In this case, the FSM for CSA instance creates a new FSM for Call Segment instance and transmits the event to it.

MergeCallSegments

NOTE 3: In this case, the SSF deletes the "source" Call Segment and connects all the Legs in the "source" Call segment with the "target" Call Segment. The FSM for CSA transmits the event to the FSM for CS instance of the "source" CS and releases the FSM instance. Furthermore, the FSM for CSA transmits the event to the FSM for CS instance of the ."target" CS.

MoveCallSegments (for target CSA)
MoveLeg

NOTE 4: In this case, the SSF moves the leg from the "source" Call Segment to the "target" Call Segment. with which the source Call Segment is associated. The FSM for CSA transmits the event to the "source" FSM for CS instance and to the "target" FSM for CS instance.

PlayAnnouncement
PromptAndCollectUserInformation
PromptAndReceiveMessage
RequestNotificationChargingEvent
ResetTimer
ScriptClose
ScriptInformation
ScriptRun
SendChargingInformation
SplitLeg

NOTE 5: In this case, the SSF creates a Call Segment and connects the split Leg with the Call Segment. The FSM for CSA transmits the event to the FSM instance for the "source" CS. Furthermore, the FSM for CSA creates a new FSM instance for the "target" Call Segment and transmits the event to the FSM.

The following operations may be received from the SCF, causing a state transition either to the same state if an FSM for CS instance exists after this event was processed in the FSM for CSA (transition E3), or to the Idle state if all the FSM for CS instances associated with the FSM for CSA instance transit to the Idle state (transition E7).

Cancel (allRequests)
CollectInformation
Connect
Continue
ContinueWithArgument
DisconnectLeg
MoveCallSegments (for source CSA)
RequestReportBCSMEvent

The **ReleaseCall** operation may be received from the SCF. For the case the whole CSA is to be released, the FSM for CSA shall instruct all the relevant FSM for CS instances to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows:

- if the last CS has been released, and if neither CallInformationReport nor ApplyChargingReport operation has been requested, the FSM for CSA transits to the Idle state (transition E7);

- if the last CS has been released, and if CallInformationReport or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the FSM for CSA transits to the Idle state (transition E7) in the FSM for CSA;

CSs are created or deleted by the CSA, if necessary, then the operations are passed to the appropriate CS and are processed there.

The following operations may be received from one of the CS, causing a state transition either to the same state if an FSM for CS instance exists after this event was processed in the FSM for CSA (transition e4), or to the Idle state if all FSM for CS instances associated with the FSM for CSA instance transit to the Idle state (transition e6). The operations are sent to the SCF:

ApplyChargingReport
CallInformationReport
EntityReleased
EventReportBCSM

The following operations may be received from one of the CS with no resulting transition to the idle state (transition e4), the operations are sent to the SCF:

EventNotificationCharging
ReturnResult for PromptAndCollectUserInformation
ReturnResult for PromptAndReceiveMessage
ScriptEvent
SpecializedResourceReport

Any other event received in the "Active" state should be processed in accordance with the general rules in subclause 11.5.

12.5.2 FSM for Call Segment

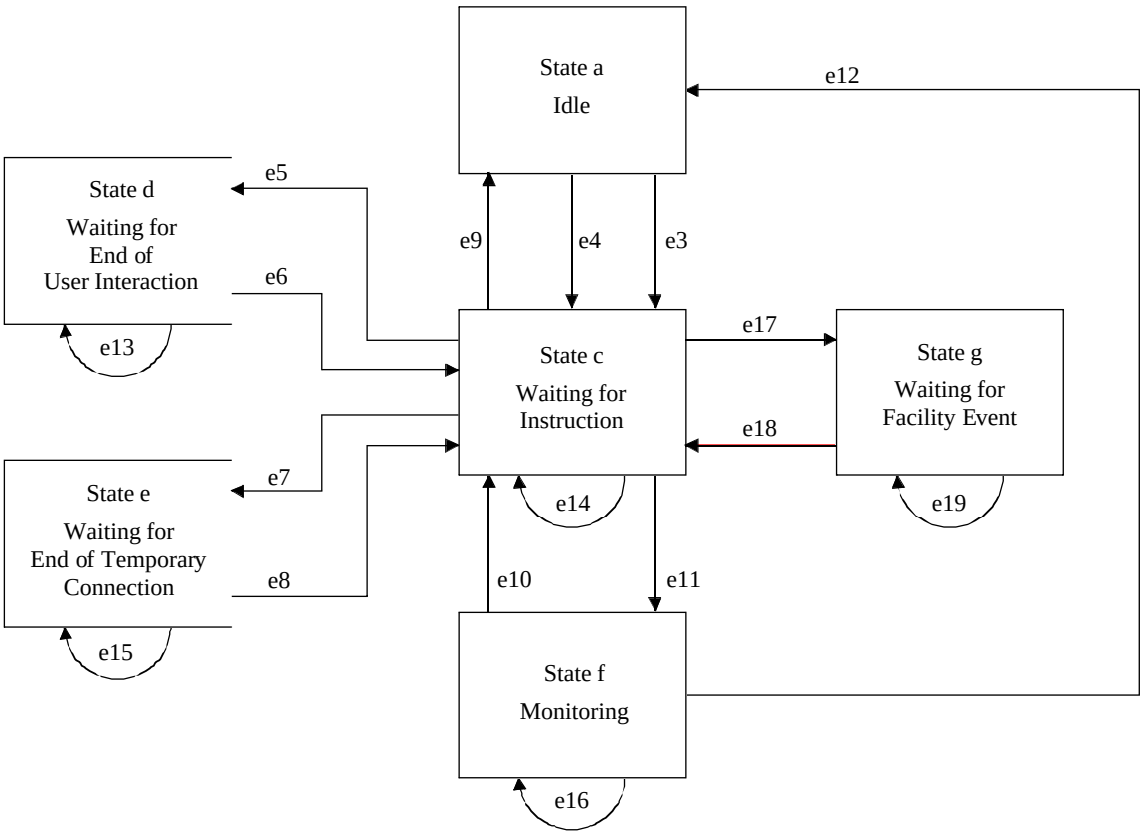


Figure 11-5: FSM for Call Segment

The SSF state diagram for the CS contains the following transitions (events):

e3 - InitiateCallAttempt received, SplitLeg received (transition when "target" CS)

e4 - TDP-R encountered

e5 - User interaction requested

e6 - User interaction ended

e7 - Temporary connection created

e8 - Temporary connection ended

e9 - Idle return from wait for instruction

e10 - EDP-R encountered

e11 - Routeing instruction received (includes Continue)

e12 - EDP-N last (see note) encountered or ReleaseCall received or Cancel(allRequests) received

e13 - Waiting For End Of User Interaction state no change

e14 - Waiting For Instructions state no change

e15 - Waiting For End Of Temporary Connection state no change

e16 - Monitoring state no change

e23 - User interaction requested

e24 - User interaction ended

e25 - Waiting For End Of User Interaction state no change

e26 - Temporary connection created

e27 - Temporary connection ended

e28 - Waiting For End Of Temporay Connection state no change

e29 - Continue has been issued

e30 - EDP-R encountered

e32 - Continue has been issued

e31 - EDP-R encountered

NOTE 1: The "last EDP-N" means that there are no other EDPs which may be encountered when an EDP-N was detected. Some of the EDPs are automatically disarmed if another EDP is encountered. The EDPs which are automatically disarmed depend on which EDP is encountered. An example is the case of the EDPs O_Answer, O_No_Answer, RouteSelectFailure or O_Called_Party_Busy. If any of these EDPs are encountered, all the other EDPs of this list are automatically disarmed.

NOTE 2: The Abandon, Disconnect and MidCall events can be encountered in any state; the transition to the next state is dependent on the current state of call processing. (Not shown in the SSF state diagram.)

NOTE 3: Non-Call Associated treatment events (refer to subclause 11.4) can be encountered in any state; the FSM for CS remains in the same state.

The SSF state diagram contains the following states:

State a Idle

State c Waiting For Instructions

State d Waiting For End Of User Interaction (WFI)

State e Waiting For End Of Temporary Connection (WFI)

State f Monitoring

State h Waiting For End Of User Interaction (Monitoring)

State i Waiting For End Of Temporary Connection (Monitoring)

In any state (except Idle), if the calling party abandons the call before it is answered (i.e. before the Active Personal Identification Code (PIC) in the BCSM), then the FSM for CS instance shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

Table 11-1

Abandon DP	operation sent to SCF	state transition
not armed	[**CallInformationReport**] + [**ApplyChargingReport**]	to Idle
armed as EDP-R	[**CallInformationReport**] + [**ApplyChargingReport**] + **EventReportBCSM**	to WaitForInstructions
armed as EDP-N	**EventReportBCSM** + [**CallInformationReport**] + [**ApplyChargingReport**]	to Idle
NOTE: The operations in brackets "[]" are only sent if the reports are pending.		

In any state (except Idle), if a call party disconnects from a stable call (i.e. from the Active PIC in the BCSM), then the SSF FSM shall process this event as follows:

Table 11-2

Disconnect DP	operation sent to SCF	state transition
not armed for that specific leg	[**CallInformationReport**] + [**ApplyChargingReport**]	to Idle
armed as EDP-R for that specific leg	[**CallInformationReport**] + [**ApplyChargingReport**] + **EventReportBCSM**	to WaitForInstructions
armed as EDP-N for that specific leg	**EventReportBCSM** + [**CallInformationReport**] + [**ApplyChargingReport**]	to Idle
NOTE: The operations in brackets "[]" are only sent if the reports are pending.		

In any state (except Idle), an **EventNotificationCharging** can be sent to the SCF, if previously requested by a RequestNotificationChargingEvent and if the charging event has been detected by the CCF. In this case no state transition takes place.

Each FSM for CS instance has an application timer, T_{SSF}, whose purpose is to prevent excessive call suspension time and to guard the association between the SSF and the SCF.

If necessary, the Timer T_{SSF} may be set in the following cases:

- when the SSF sends an InitialDP for a TDP-R (see subclause 11.5.2.2 State c: Waiting For Instructions). While waiting for the first response from the SCF, the timer T_{SSF} can be restarted only once by a ResetTimer operation. Subsequent to the first response, the timer can be restarted any number of times;

- when the FSM for CS enters the Waiting For Instructions state (see subclause 11.5.2.2) under any other condition than the ones listed in the previous case. In this case the SCF may restart the T_{SSF} timer using the ResetTimer operation any number of times;

- when the SSF enters the Waiting For End Of User Interaction state or the Waiting For End Of Temporary Connection state (see subclauses 11.5.2.3 and 11.5.2.4). In these cases the SCF may restart T_{SSF} using the ResetTimer operation any number of times (OPTIONAL).

  This "OPTIONAL" means that the application timer T_{SSF} is optionally set. Whether it is used or not depends on implementation. But it needs to be synchronized with $T_{SCF\text{-}SSF}$ in the SCSM.

In each of the above cases, T_{SSF} may have different values as defined by the application.

When receiving or sending any operation which is different from the above, the FSM for CS instance shall restart T_{SSF} with the last used value. This value is either one associated with the different cases as listed above, or received in a ResetTimer operation, whatever occurred last. In the Monitoring state (see subclause 11.5.2.5) T_{SSF} is not used.

On expiry of T_{SSF} the FSM for CS transits to the Idle state and the CCF progresses the BCSM if possible. If the FSM for CS was the last in the CSA the interaction with the SCF is aborted, otherwise the operation EntityReleased is sent to the SCF.

The valid state transitions for the FSM for CS are contained in table 11-3.

The columns present the current state in which the operation or event may occur.

The existence of an entry indicates that in the current state the operation/event is valid.

The text of the entry indicates the transition to the state after processing the operation/event.
Please note that the table does not cover the error handling (e.g. Disconnect, Abandon) and does not show how the SSF examines subsequent operations in a sequence and execute and buffers operations in accordance with the general rules described in subclause 11.5.

Table 11-3: "FSM for CS transition table" for valid FSM for CS transitions

	State a	State c	State d	State e	State h	State i	State f
	Idle	**WfI**	**WfEoU (WFI)**	**WfEoTC (WFI)**	**WfEoU (MON)**	**WfEoTC (MON)**	**Mon**
Operations							
ApplyCharging		same	same	same	same	same	same
CallInformationRequest		same					
Cancel(allRequests)		same					idle
CollectInformation		idle (note 2), Mon (note 3)					
Connect		idle (note 2), Mon (note 3)					
ConnectToResource		WfEoUI (WFI)					WfEoUI (MON)
Continue		idle (notes 1, 2), Mon (notes 1, 3)	idle (notes 1, 2, 10) WfEoU (MON) (notes 1, 3, 10)	idle (notes 1, 2, 10) WfEoTC (MON) (notes 1, 3, 10)			
ContinueWithArgument		idle (note 2), Mon (note 3)	idle (note 2) WfEoU (MON) (notes 3, 10)	idle (note 2 WfEoTC (MON) (notes 3, 10)			
CreateCallSegmentAssociation							
DFCWithArgument			WfI	WfI	Mon	Mon	
DisconnectForwardConnection			WfI (note 9)	WfI (note 9)	Mon (note 9)	Mon (note 9)	
DisconnectLeg(last)		idle					idle
DisconnectLeg(not last)		same					WfI
EstablishTemporaryConnection		WfEoTC (WFI)					WfEoTC (MON)

(continued)

Table 11-3 (continued): "FSM for CS transition table" for valid FSM for CS transitions

	State a	State c	State d	State e	State h	State i	State f
	Idle	WfI	WfEoU (WFI)	WfEoTC (WFI)	WfEoU (MON)	WfEoTC (MON)	Mon
FurnishChargingInformation		same	same	same	same	same	same
InitiateCallAttempt	WfI						
InitiateCallAttempt(new CS)		same					
MergeCallSegments(targetCS)		same					WfI
MergeCallSegments(sourceCS)		idle					idle
MoveCallSegments							
MoveLeg(last leg for sourceCS)		idle					idle
MoveLeg(targetCS; not last leg for sourceCS)		same					WfI
ReleaseCall							
ReleaseCall(for a CS)		idle					idle
RequestNotificationChargingEvent		same	same	same	same	same	same
RequestReportBCSMEvent		same					idle (note 2, same (note 3
ResetTimer		same	same (note 4	same (note 4	same (note 4	same (note 4	
SendChargingInformation		same	same	same	same	same	same
SplitLeg(sourceCS: not last)		same					WfI
SplitLeg(newCS)	WfI						
Operations for relaying to SRF							
Cancel(invokeID)			same (note 8)		same (note 8)		
PlayAnnouncement			same (note 8)		same (note 8)		
PromptAndCollectUserInformation			same (note 8)				
PromptAndReceiveMessage			same (note 8)				
ScriptClose			same (note 8)				
ScriptInformation			same (note 8)				
ScriptRun			same (note 8)				
Events							

(continued)

Table 11-3 (concluded): "FSM for CS transition table" for valid FSM for CS transitions

	State a	State c	State d	State e	State h	State i	State f
	Idle	WfI	WfEoU (WFI)	WfEoTC (WFI)	WfEoU (MON)	WfEoTC (MON)	Mon
TDP-R (InitialDP)	WfI						
Tssf		idle	idle (note 4)	idle (note 4)	Idle (note 4)	Idle (note 4)	idle
event - disconnect from SRF			WfI	WfI	Mon	Mon	
event - EDP-R					WfEoU (WFI) WFI (note 11)	WfEoTC (WFI) WFI (note 11)	WfI
event - EDP-N					Idle (note 2), same (note 3) Mon (note 11)	Idle (note 2), same (note 3) Mon (note 11)	idle (note 2), same (note 3)
event - last charging event							idle
event - Feature activation/hook flash (EDP-R)		same (note 7)	same (note 7)	same (note 7)	WfEoU (WFI)	WfEoTC (WFI)	WfI
Events for relaying from SRF							
event - SpecializedResourceReport			same (note 8)		same (note 8)		
ReturnResult from PaCUI			same (note 8)				
ReturnResult from PaM			same (note 8)				
ScriptEvent			same (note 8)				

NOTE 1: Only applicable for a single CS with no more than 2 legs, use of this operation is not valid in a multi call segment CSA.
NOTE 2: No EDPs armed and no pending requests.
NOTE 3: EDPs armed or pending requests.
NOTE 4: Use of Timer Tssf in this state is optional.
NOTE 7: Only if MidCall DP was armed to be reported in any state, except Idle.
NOTE 8: Operations/events for relaying to SRF.
NOTE 9: Only applicable for a single CS , use of this operation is not valid in a multi call segment CSA.
NOTE 10: Only allowed if applied user interaction operation is PlayAnnouncement.
NOTE 11: In case of Abandon or Disconnect for the leg that is connected to the resource.

The following subclauses give more specific information for each state in addition to the FSM for CS transition table.

12.5.2.1 State a: Idle

The FSM for CS enters the Idle state under a variety of conditions, as described below and in the FSM for CS transition table.

The FSM for CS enters the Idle state when the associated FSM for CSA instance transits to the Idle state.

The FSM for CS enters the Idle state when one of the following events occurs:

- when the call is abandoned or one or more call parties disconnect in any other state under the conditions identified in subclause 11.5.2;
- when an operations is processed in the Waiting For Instructions state, and no EDPs are armed and there are no outstanding report requests (transition e9); refer to "FSM for CS transition table".

When transiting to the Idle state, if there is a CallInformationReport and/or ApplyChargingReport pending (see subclause 11.5.2), the SSF sends a CallInformationReport and/or ApplyChargingReport operation to the SCF before returning to Idle. Once in the Idle state, if status reporting is still active the SSF deactivates it, any outstanding responses to send to the SCF are discarded.

During the state Idle the following call-associated events can occur:

- indication from the CCF that an armed TDP-R is encountered related to a possible IN call/service attempt, the FSM for CS instance sends a generic **InitialDP** to the associate FSM for CSA, as determined from DP processing, and transit to the Waiting For Instructions state (transition e4); (the rules for DP processing are described in subclause 5.? "DP Processing");
- an **InitiateCallAttempt** operation is received from the SCF: in this case a new instance of a FSM for CS has been created by the associate FSM for CSA. This FSM for CS instance transits to the Waiting For Instructions state (transition e3);
- a **SplitLeg** operation is received from the SCF: in this case a new target FSM for CS instance is created by the associate FSM for CSA. This FSM for CS instance transits to the Waiting For Instructions state (transition e3).

Any other operation received from the SCF while the FSM is in Idle state should be treated as an error. The event should be reported to the maintenance functions.

12.5.2.2 State c: Waiting For Instructions

This state is entered:

- from the Idle state, either directly as indicated above (transition e4), or on receipt of an **InitiateCallAttempt** or **SplitLeg** ("target" CS) operation from the SCF (transition e3);
- from the state Monitoring on detection of an EDP-R (transition e10);
- from the state Waiting For End Of User Interaction(WFI) on occurrence of disconnection of the SRF (transition e6);
- from the state Waiting For End Of Temporary Connection(WFI) on occurrence of disconnection of temporary connection (transition e8).

In this state the FSM for CS is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) shall be set on entering this state.

During the state Waiting For Instruction the following events can occur:

- The user dials additional digits (applies for open-ended numbering plans): the CCF should store the additional digits dialled by the user.
- The user abandons or disconnects. This should be processed in accordance with the general rules in subclause 11.5.
- The user issues a feature activation or a Switch_Hook_Flash, which shall lead to the reporting of MidCall as an EDP-R or EDP-N if the MidCall DP was armed to be reported in any state except Idle.

- The application timer T_{SSF} expires: the FSM for CS moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions, and the operation EntityReleased is sent to the FSM for CSA.

- An operation is received from the SCF: The FSM for CS instance acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e14): refer to table 11-3.

For the case where a **SplitLeg** or an **InitiateCallAttempt** (for non-initial CS) operation is received from the SCF a new CS and a new instance of an FSM for CS has been created by the associate FSM for CSA. This new FSM instance shall receive the SplitLeg or InitiateCallAttempt in the Idle state and transit to the Waiting For Instructions state (transition e3). The FSM for CSA also sends the SplitLeg operation to the source FSM for CS instance, which may be in the Waiting For Instructions or Monitoring state.

ReleaseCall (for a CS) or **DisconnectLeg** (for the last leg in the CS) operation may be received from the SCF. In this case, the FSM for CS instance shall instruct the CCF to clear the call segment and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

- if neither CallInformationReport nor ApplyChargingReport operation has been requested, the FSM for CS transits to the Idle state (transition e9);

- if CallInformationReport or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the FSM for CS transits to the Idle state (transition e9).

The **MergeCallSegments (for "source" CS)** operation or MoveLeg (for last leg in "source" CS) operation may be received from the SCF. In this case, the SSF FSM related to the "source" CS shall return to the Idle state (transition e9).

When processing the above operations, any necessary call handling information is provided to the CCF.

Any other operation received in this state should be processed in accordance with the general rules in subcluase 11.5.

12.5.2.3 State d: Waiting For End Of User Interaction (WFI)

The SSF enters this state from the Waiting For Instructions state (transition e5) on the reception of the operation **ConnectToResource**

The timer T_{SSF} is active in this state. (Whether it is used or not is optional.).

During this state the following events can occur:

- One of the following valid SCF-SRF operations for relaying is received and is correct, the operation is transferred to the SRF for execution: refer to "FSM for CS transition table".

  The SSF FSM remains in the Waiting For End Of User Interaction state (transition e13).

- One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF: refer to "FSM for CS transition table".

  The SSF FSM remains in the Waiting For End Of User Interaction state (transition e13).

- The application timer T_{SSF} expires (if it was set): the FSM for CS moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the operation EntityReleased is sent.

- An operation is received from the SCF: the FSM for CS acts according to the operation received as described below.

- The user abandons. This should be processed in accordance with the general rules in subclause 11.5.2.

- The user issues a feature activation or a Switch Hook Flash, which shall lead to the reporting of MidCall as an EDP-R or EDP-N if the MidCall DP was armed to be reported in any state except Idle.

The following operations may be received from the SCF and processed by the SSF, causing a state transition to Waiting For End Of User Interaction (Monitoring) state (transition e32): refer to "FSM for CS transition table".

- (none specified)

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e13): refer to "FSM for CS transition table".

- (none specified)

The **DisconnectForwardConnection** (only applicable for a single CS, use of this operation is not valid in a multi call segment CSA) or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the SSF in this state. Disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the Waiting For Instructions state. The disconnection is not transferred to the other party (transition e6).

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

12.5.2.4 State e: Waiting For End Of Temporary Connection (WFI)

The FSM for CS enters this state from the Waiting For Instructions state (transition e7) upon receiving an EstablishTemporaryConnection operation.

The call is routed to the assisting SSF/SRF and call handling is suspended while waiting for the end of the assisting procedure. The timer T_{SSF} is active in this state. (Whether it is used or not is optional.)

During the state Waiting For End Of Temporary Connection the following events can occur:

- The application timer T_{SSF} expires (if it was set): the FSM for CS moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the operation EntityReleased is sent.

- The receipt of an indication of disconnection of forward connection from the CCF. In this case, the SSF moves to the Waiting For Instructions state (transition e8). The disconnection is not transferred to the calling party.

- The user abandons. This should be processed in accordance with the general rules in subclause 11.5.2.

- The user issues a feature activation or a Switch_Hook_Flash, which shall lead to the reporting of MidCall as an EDP-R or EDP-N if the MidCall DP was armed to be reported in any state except Idle.

- An operation is received from the SCF; the FSM for CS acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the SSF, causing a state transition to Waiting For End Of Temporary Connection (Monitoring) state (transition e32): refer to "FSM for CS transition table".

- (none specified)

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e15): refer to "FSM for CS transition table".

- (none specified)

The **DisconnectForwardConnection** (only applicable for a single CS, use of this operation is not valid in a multi call segment CSA) or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the SSF in this state. Disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the Waiting For Instructions state. The disconnection is not transferred to the other party (transition e8).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

12.5.2.5 State f: Monitoring

The FSM for CS enters this state from the Waiting For Instructions state (transition e11) upon receiving one of several operations when one or more EDPs are armed or/and there are other reports pending (see 11.5): refer to "FSM for CS transition table".

The timer T_{SSF} is stopped on entering this state

During the state Monitoring the following events can occur: refer to table 11-3.

- If the event causing a CallInformationReport and/or an ApplyChargingReport is also detected by an armed EDP-N then the **CallInformationReport** and/or **ApplyChargingReport** shall be sent immediately after the corresponding **EventReportBCSM** is sent.
- If the event causing a CallInformationReport and/or ApplyChargingReport is also detected by an armed EDP-R then the **CallInformationReport** and/or **ApplyChargingReport** shall be sent immediately before the corresponding **EventReportBCSM** is sent.
- An operation is received from the SCF: the FSM for CS acts according to the operation received as described below.
- The user abandons or disconnects. This should be processed in accordance with the general rules in subclause 11.5.2.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e16): refer to table 11-3.

The following operations may be received from the SCF and processed by the SSF, causing a state transition to the Waiting For Instruction state (transition e10, valid for SCF initiated changes of Connection view states): refer to "FSM for CS transition table".

The following operations may be received from the SCF and processed by the SSF, causing a state transition to the Idle state (transition e12, valid for SCF initiated changes of Connection view states): refer to "FSM for CS transition table".

When the **ReleaseCall** operation is received from the SCF, the FSM for CS shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

- if neither CallInformationReport nor ApplyChargingReport operation has been requested, the FSM for CS transits to the Idle state (transition e12);
- if CallInformationReport or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the FSM for CS shall transit to the Idle state (transition e12).

The following operations may be received from the SCF and processed by the SSF, causing a state transition to Waiting For End Of User Interaction (Monitoring) state (transition e23): refer to table 11-3.

The following operations may be received from the SCF and processed by the SSF, causing a state transition to Waiting For End Of Temporary Connection (Monitoring) state (transition e26): refer to table 11-3.

- (none specified)

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

12.5.2.6 State h: Waiting For End Of User Interaction (Monitoring)

The SSF enters this state from the Monitoring state (transition e23) on the reception of the operation **ConnectToResource**

The timer T_{SSF} is active in this state. (Whether it is used or not is optional.)

During this state the following events can occur:

- One of the following valid SCF-SRF operations for relaying is received and is correct, the operation is transferred to the SRF for execution: refer to table 11-3.
- The FSM for CS remains in the Waiting For End Of User Interaction (Monitoring) state (transition e25).

One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF: refer to table 11-3.

- The FSM for CS remains in the Waiting For End Of User Interaction (Monitoring) state (transition e25).
- The application timer T_{SSF} expires (if it was set): the FSM for CS moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the operation EntityReleased is sent.
- An operation is received from the SCF: The FSM for CS acts according to the operation received as described below.
- The user abandons. This should be processed in accordance with the general rules in subclause 11.5.2.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e25): refer to "FSM for CS transition table". During the state Waiting For End Of User Interaction (Monitoring) the following events can occur:

- An EDP-N is reported to the SCF by sending an EventReportBCSM operation; the FSM for CS shall remain in this Waiting For End Of User Interaction (Monitoring) state (transition e25) if one or more EDPs are armed or there are report requests pending.
- If the event causing a CallInformationReport is also detected by an armed EDP-N then the CallInformationReport shall be sent immediately after the corresponding EventReportBCSM operation is sent.
- An EDP-R should be reported to the SCF by sending an EventReportBCSM operation; the FSM for CS should move to the Waiting For End Of User Interaction (WFI) state (transition e31).
- If the event causing a CallInformationReport is also detected by an armed EDP-R then the CallInformationReport shall be sent immediately before the corresponding EventReportBCSM operation is sent.
- An operation is received from the SCF: the FSM for CS acts according to the operation received as described below.
- The user abandons or disconnects. This should be processed in accordance with the general rules in subclause 11.5.2.

The **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the SSF in this state. Disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the Monitoring state. The disconnection is not transferred to the other party (transition e24).

Any other operation received in this state should be processed in accordance with the general rules in 11.5.

12.5.2.7 State i: Waiting For End Of Temporary Connection (Monitoring)

The FSM for CS enters this state from the Monitoring state (transition e26) upon receiving an **EstablishTemporaryConnection** operation.

The call is routed to the assisting SSF/SRF and call handling is suspended while waiting for the end of the assisting procedure. The timer T_{SSF} is active in this state. (Whether it is used or not is optional.)

During the state Waiting For End Of Temporary Connection (Monitoring) the following events can occur:

- The application timer T_{SSF} expires (if it was set): the FSM for CS moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the operation EntityReleased is sent.
- The receipt of an indication of disconnection of forward connection from the CCF. In this case, the SSF moves to the Waiting For Instructions state (transition e8). The disconnection is not transferred to the calling party.
- The user abandons. This should be processed in accordance with the general rules in subclause 11.5.2.
- An operation is received from the SCF; the FSM for CS acts according to the operation received as described below.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e28): refer to table 11-3.

During the state Waiting For End Of Temporary Connection (Monitoring) the following events can occur:

- An EDP-N is reported to the SCF by sending an EventReportBCSM operation; the FSM for CS shall remain in this Waiting For End Of User Interaction (Monitoring) state (transition e28) if one or more EDPs are armed or there are report requests pending.
- If the event causing a CallInformationReport is also detected by an armed EDP-N then the CallInformationReport shall be sent immediately after the corresponding EventReportBCSM.
- An EDP-R should be reported to the SCF by sending an EventReportBCSM operation; the FSM for CS should move to the Waiting For Instructions state (transition e30).
- If the event causing a CallInformationReport is also detected by an armed EDP-R then the CallInformationReport shall be sent immediately before the corresponding EventReportBCSM.
- An operation is received from the SCF: the FSM for CS acts according to the operation received as described below.
- The user abandons or disconnects. This should be processed in accordance with the general rules in subclause 11.5.2.

The **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the SSF in this state. Disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the Waiting For Instructions state. The disconnection is not transferred to the other party (transition e27).

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

12.6 Assisting SSF FSM

This subclause describes the SSF FSM related to the Assisting SSF.

The Assisting SSF state diagram contains the following transitions (events) (see figure 11-3):

ea1 Assist detected

ea2 Assist failed

ea3 User interaction requested

ea4 User interaction ended

ea5 Waiting For Instructions state no change

ea6 Waiting For End Of User Interaction state no change

ea7 Idle Return from Waiting For End Of User Interaction

The Assisting SSF state diagram contains the following states:

State aa Idle

State ab Waiting For Instructions

State ac Waiting For End Of User Interaction

12.6.1 State aa: Idle

The FSM for Assisting SSF enters the Idle state when one of the following occurs:

- when sending or receiving an TC-ABORT primitive due to abnormal conditions in any state;
- given a temporary connection between an Initiating SSF and the Assisting SSF, when a bearer channel disconnect is received from the initiating SSF; (transition ea2).

Once in the Idle state, if there are any outstanding responses to send to the SCF, they are discarded by the Assisting SSF.

The FSM for Assisting SSF transits from the Idle state to the Waiting For Instructions state on receipt of an assist indication at the assisting SSF from another SSF (transition ea1).

Any operation received from the SCF while the Assisting SSF is in Idle state shall be treated as an error. The event shall be reported to the maintenance functions and the transaction shall be aborted according to the procedure specified in TC (see ETS 300 287-1 [7]).

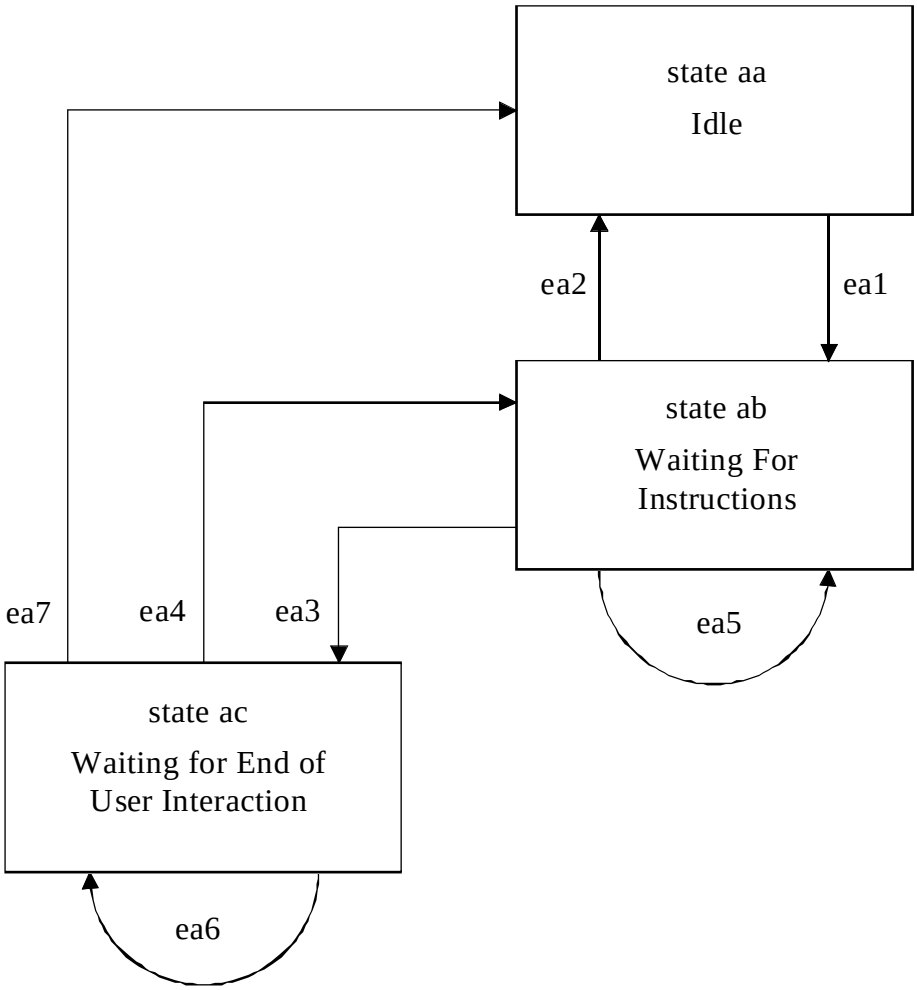


Figure 11-1: FSM for Assisting SSF

12.6.2 State ab: Waiting For Instructions

This state is entered from the Idle state on receipt of a CONnect message (CON) at an SSF from another SSF indicating that an assist is required, based on an implementation dependent detection mechanism (transition ea1).

Before entering this state, the SSF sends an **AssistRequestInstructions** operation to the SCF and the FSM for Assisting SSF is waiting for an instruction from the SCF; call handling is suspended and application timer T_{SSF} shall be set on entering this state.

During this state the following events can occur:

- The application timer T_{SSF} expires: the FSM for Assisting SSF moves to the Idle state (transition ea2) and the expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: the FSM for Assisting SSF acts according to the operation received as described below.
- A bearer channel disconnect is received and the FSM moves to the Idle state (transition ea2).

The following operations can be received from the SCF and processed by the Assisting SSF with no resulting transition to a different state (transition ea5):

ApplyCharging
FurnishChargingInformation
ResetTimer
SendChargingInformation

The following operations can be received from the SCF and processed by the Assisting SSF, causing a state transition to Waiting For End Of User Interaction state (transition ea3):

ConnectToResource

In the case where an implementation is not capable of differentiating between a Handed-off and an Assisting SSF case, it may execute the **ReleaseCall** operation in the assisting SSF. The FSM for Assisting SSF shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

- if ApplyChargingReport operation is not requested, the FSM for Assisting SSF transits to the Idle state (transition ea2);
- if ApplyChargingReport operation has been requested, the FSM for Assisting SSF sends **ApplyChargingReport** to SCF and then the FSM for Assisting SSF transits to the Idle state (transition ea2).

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

11.6.3 State ac: Waiting For End Of User Interaction

The FSM for Assisting SSF enters this state from the Waiting For Instructions state (transition ea3) on the reception of the following operation:

ConnectToResource

During this state the following events can occur:

- One of the following valid SCF-SRF operations for relaying is received and is correct, the operation is transferred to the SRF for execution:

  Cancel (invokeID)
  PlayAnnouncement
  PromptAndCollectUserInformation
  PromptAndReceiveMessage
  ScriptClose
  ScriptInformation
  ScriptRun

  The FSM for Assisting SSF remains in the Waiting For End Of User Interaction state (transition ea6).

- One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF:

  SpecializedResourceReport
  ReturnResult from PromptAndCollectUserInformation
  ReturnResult from PromptAndReceiveMessage
  ScriptEvent

  The SSF for Assisting SSF remains in the Waiting For End Of User Interaction state (transition ea6).

- The application timer T_{SSF} expires: the FSM for Assisting SSF moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted (transition ea7).

- An operation is received from the SCF: The FSM for Assisting SSF acts according to the operation received as described below.

- A bearer channel disconnect is received from the initiating SSF and the FSM moves to the Idle state (transition ea7).

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition ea6):

ResetTimer

The **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the Assisting SSF in this state, causing a transition to the Waiting For Instructions state (transition ea4). This procedure is only valid if a ConnectToResource was previously processed to cause a transition into the Waiting For End Of User Interaction state.

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

12.7 Handed-off SSF FSM

This subclause describes the SSF FSM related to the Handed-off SSF. The SSF FSM for IN-CS2 applies only to the case where final treatment is to be applied.

The Handed-off SSF state diagram contains the following transitions (events) (see figure 11-3):

eh1 Handed-off detected

eh2 Handed-off fail

eh3 User interaction requested

eh4 User interaction ended

eh5 Waiting For Instructions state no change

eh6 Waiting For End Of User Interaction state no change

eh7 Idle Return from Waiting For End Of User Interaction

The Handed-off SSF state diagram contains the following states:

State ha Idle

State hb Waiting For Instructions

State hc Waiting For End Of User Interaction

12.7.1 State ha: Idle

The FSM for Handed-off SSF enters the Idle state when one of the following occurs:

- when sending or receiving an TC-ABORT primitive due to abnormal conditions in any state.

When the bearer channel disconnects, FSM for Handed-off FSM shall also move to Idle.

Once in the Idle state, if there are any outstanding responses to send to the SCF, they are discarded by the Handed-off SSF.

The FSM for Handed-off SSF transits from the Idle state to the Waiting For Instructions state on receipt of an assist indication at the Handed-off SSF from another SSF (transition eh1).

Any operation received from the SCF while the Handed-off SSF is in Idle state should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TC (see ETS 300 287-1 [7]).

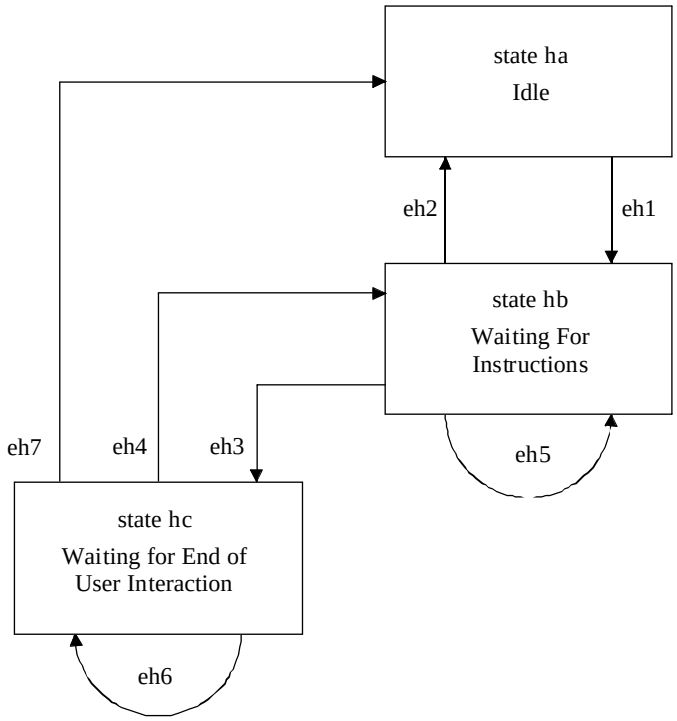


Figure 11-2: FSM for Handed-off SSF

12.7.2 State hb: Waiting For Instructions

This state is entered from the Idle state on receipt of a connect at an SSF from another SSF indicating that a hand-off is required, based on an implementation dependent detection mechanism (transition eh1).

Before entering this state, the SSF sends an AssistRequestInstructions operation to the SCF and the FSM for Handed-off SSF is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) should be set on entering this state.

During this state the following events can occur:

- The application timer T_{SSF} expires: the FSM for Handed-off SSF moves to the Idle state (transition eh2) and the expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: the FSM for Handed-off SSF acts according to the operation received as described below.
- A bearer channel disconnect is received and the FSM moves to the Idle state (transition eh2).

The following operations may be received from the SCF and processed by the Handed-off SSF with no resulting transition to a different state (transition eh5):

ApplyCharging
FurnishChargingInformation
ResetTimer
SendChargingInformation

The following operations can be received from the SCF and processed by the Handed-off SSF, causing a state transition to Waiting For End Of User Interaction state (transition eh3):

ConnectToResource

If the **ReleaseCall** operation is received from the SCF, the Handed-off SSF FSM shall instruct the CCF to clear the call and ensure that any CCF resources allocated to the call are de-allocated, processing shall continue as follows:

- if ApplyChargingReport operation is not requested, the Handed-off SSF FSM transits to the Idle state (transition eh2);
- if ApplyChargingReport operation has been requested, the Handed-off SSF sends ApplyChargingReport to SCF and then the Handed-off SSF FSM transits to the Idle state (transition eh2).

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

Note that multiple Handoff procedures are not covered by the present document.

12.7.3 State hc: Waiting For End Of User Interaction

The FSM for Handed-off SSF enters this state from the Waiting For Instructions state (transition eh3) on the reception of one of the following operations:

ConnectToResource

During this state the following events can occur:

- One of the following valid SCF-SRF operation for relaying is received and is correct, the operation is transferred to the SRF for execution:

  Cancel(invokeID)
  PlayAnnouncement
  PromptAndCollectUserInformation
  PromptAndReceiveMessage
  ScriptClose
  ScriptInformation
  ScriptRun

  The FSM for Handed-off SSF remains in the Waiting For End Of User Interaction state (transition eh6).

- One of the following valid SRF-SCF operations for relaying is received and is correct, the operation is transferred to the SCF:

  SpecializedResourceReport
  ReturnResult from PromptAndCollectUserInformation
  ReturnResult from PromptAndReceiveMessage
  ScriptEvent

  The SSF for Handed-off SSF remains in the Waiting For End Of User Interaction state (transition eh6).

- When the SRF indicates to the SSF the end of user interaction by initiating disconnection the FSM for Handed-off SSF returns to the Waiting For Instructions state (transition eh4).
- The application timer T_{SSF} expires: the FSM for Handed-off SSF moves to the Idle state, the CCF routes the call if possible (e.g. default routeing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted. (transition eh7).
- An operation is received from the SCF: The FSM for Handed-off acts according to the operation received as described below.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition eh6):

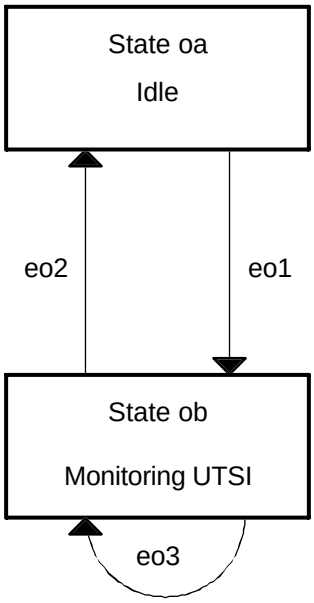
ResetTimer

The **DisconnectForwardConnection** or **DisconnectForwardConnectionWithArgument** operation may be received from the SCF and processed by the Handed-off SSF in this state, causing a transition to the Waiting For Instructions state (transition eah4). This procedure is only valid if a ConnectToResource was previously processed to cause a transition into the Waiting For End Of User Interaction state.

Any other operation received in this state should be processed in accordance with the general rules in subclause 11.5.

12.8 User Service Interaction (USI) FSM

The SSF_USI FSM illustrates state transitions for requesting and cancelling the monitoring of the reception of User To Service Information (UTSI) operations and the sending of STUI operations. This FSM has two states which are "Monitoring UTSI" and "Idle":



USI - User Service Information
UTSI - User to Service Information
STUI - Service to User Information

Figure 11-3: FSM for SSF_USI

The SSF_USI FSM transitions are defined in the following way:

eo1 The SCF requests the SSF to monitor the receipt of an UTSI Information Element (IE) for a given *USIServiceIndicator* value by sending a RequestReportUTSI operation for a particular party indicated by the leg ID by setting the "USIMonitorMode" value to "monitoringActive".

eo2 The SCF requests the SSF to stop monitoring the reception of an UTSI IE for a given *USIServiceIndicator* value by sending a RequestReportUTSI operation for a particular party indicated by the leg ID by setting the "USIMonitorMode" value to "monitoringInactive".

eo3 The SCF either sends an STUI IE by means of a SendSTUI operation to the user indicated by the leg ID for a given *USIServiceIndicator* value; or receives by means of a ReportUTSI operation.

With the same operation, the SCF requests the SSF to monitor or to stop monitoring the receipt of an UTSI IEs with a given *USIServiceIndicator* value.

NOTE: As an SCF controls or monitors the call, the SSF FSM is in any state except "Idle"; but the OCCRUI mechanism does not cause any transition in the SSF FSM.

When the FSM for the CallSegment returns to the Idle state then also the FSM for SSF_USI returns to Idle.

The SCF may send the operation SendSTUI in the state Idle of the FSM for SSF_USI.

13 SCF AE procedures

13.1 General

This subclause provides the definition of the SCF AE procedures related to the SCF-SSF/SRF/SDF/CUSF interfaces. The procedures are based on the use of SS7; other signalling systems can also be used.

In addition, other capabilities may be supported in an implementation-dependent manner in the SCP, AD or SN.

NOTE: This subclause is for information only since the FSM for the SCF has not been fully simulated and verified.

The AE, following the architecture defined in ITU-T Recommendations Q.700 [19], Q.1400 [30] and in ETS 300 287-1 [7] includes TC and one or more ASEs called TC-users. The following subclauses define the TC-user ASE and SACF/MACF rules, which interface with TC using the primitives specified in ETS 300 287-1 [7].

The procedure may equally be used with other message-based signalling systems supporting the application layer structures defined. By no means is this text intended to dictate any limitations to Service Logic Programs (SLPs).

In case interpretations for the AE procedures defined in the following differ from detailed procedures and the rules for using the TC service, the statements and rules contained in clauses 17 and 18 shall be followed.

13.2 Model and Interfaces

The functional model of the AE-SCF is shown in figure 12-4; the ASEs interface with supporting protocol layers to communicate with the SSF, SRF, CUSF and SDF, and interface to the SLPs and maintenance functions. The scope of this ITU-T Recommendation is limited to the shaded area in figure 12-4.

NOTE: The SCF FSM includes several Finite State Machines.

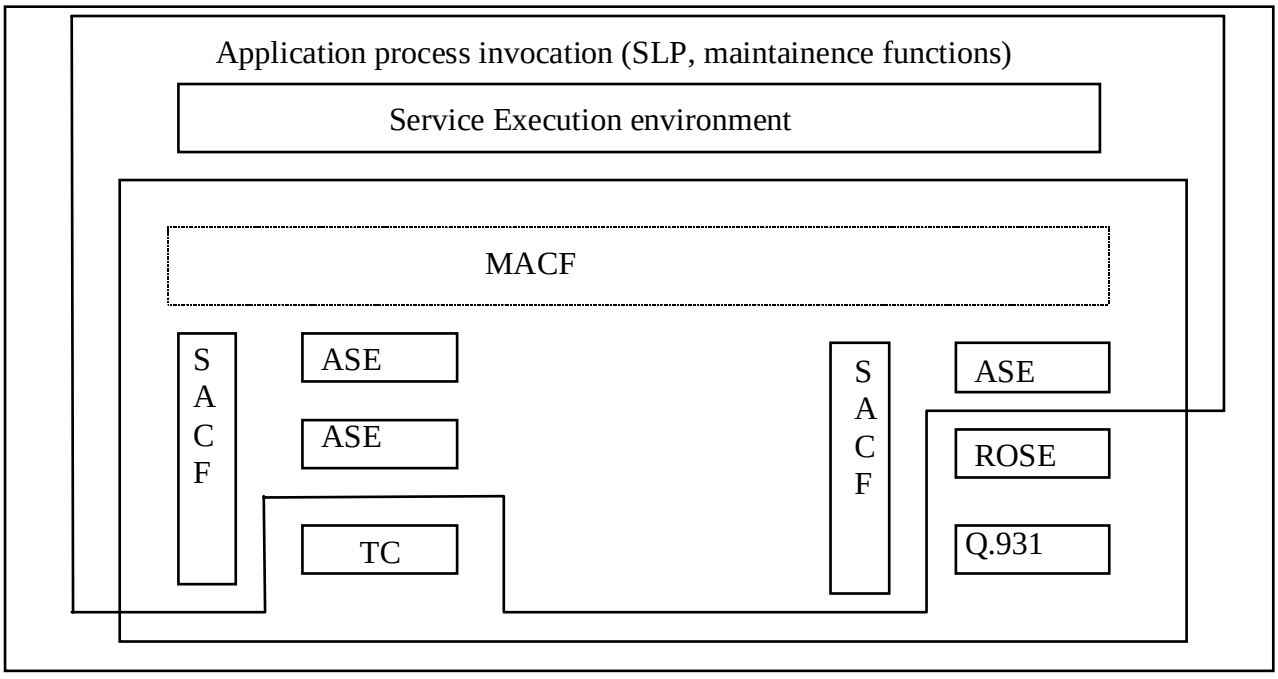


Figure 12-4: Functional model of SCF AE

jmm Figures are uneditable.

jmm Figures need renumbering

The interfaces shown in figure 12-4 use the TC-user ASE primitives specified in ETS 300 009-1 [3]) (interface (1)) and N-primitives specified in ETS 300 009-1 [3] (interface (2)). The operations and parameters of INAP are defined in Clause X of this ETS.

jmm identify this clause

13.3 Relationship between the SCF FSM and the SLPs/Maintenance Functions

The primitive interface between the SCF FSM and the SLPs/maintenance functions is an internal interface and is not a subject for standardization.

In the SCSM instance, there are four kinds of FSMs: SSF/SRF, SDF, SCF and CUSF related states.

SSF/SRF related states consist of FSMs for SSF/SRF, for CSA, for Call Segment for Specialized Resource, for Handed-off SSF, and for Assisting SSF.

The following text systematically describes the procedural aspects of the interface between the SCF and other functional entities, with the main goal of specifying the proper order of operations rather than entities' functional capabilities. Consequently, this text describes only a subset of the SCF functional capabilities.

The relationship between the SLP and the SCF FSM may be described as follows (for both cases where a call is initiated by an end user and by IN SL):

- If a request for IN call processing is received from the SSF, an instance of an SCF Call State Model (SCSM) is created, and the relevant SLP is invoked;

- When initiation of a call is requested from SL, an instance of the SCSM is created.

In either case, the SCF FSM handles the interaction with the SSF FSM (and the SRF FSM and SDF FSM) as required, and notifies the SLP of events as needed.

The management functions related to the execution of operations received from the SCF are executed by the SCF Management Entity (SCME). Multiple requests may be executed concurrently and asynchronously by the SCF, which explains the need for a single entity that performs the tasks of creation, invocation, and maintenance of the SCF FSM objects.

The SCME is comprised of the SCME-Control and multiple instances of SCME FSMs. The SCME-Control interfaces different SCSMs and the FEAM. Figure 12-5 shows the SCF FSM structure.

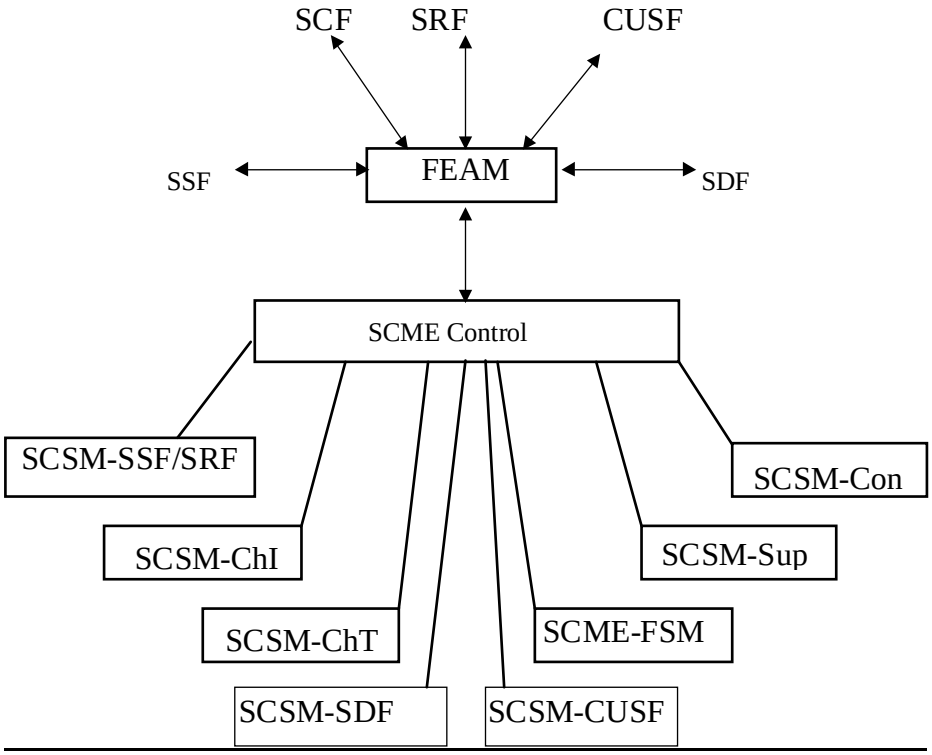


Figure 12-5: SCF FSM Structure

The SCME-FSM handles the interaction between the SCF and the SCF management functions.

The SSF/SRF FSM (SCSM-SSF/SRF) handles the interaction with the SSF/SRF FSM.

The SDF FSM (SCSM-SDF) handles the interaction with the SDF FSM.

The CUSF FSM (SCSM-CUSF) handles the interaction with the CUSF FSM.

The SCSM-Sup and SDSM-Con handle the interactions between SCFs, for the supporting and controlling role respectively.

The SCSM-ChI and SCSM-ChT handle the interactions between SCFs for chaining initiation and termination.

The management functions related to the execution of operations received from the SSF, SRF, SDF, CUSF or co-operating SCF are executed by the SCME. The SCME is comprised of an SCME control and several instances of SCME FSMs. The SCME control interfaces the different SCSMs (i.e. SCSM-SSF) and SCME-FSMs respectively as well as the FEAM.

The SCME Control maintains the associations with the SSF, SRF, SDF, CUSF and co-operating SCFs on behalf of all instances of the SCF FSMs (e.g., SCSM-SSF, SCSM-ChI). These instances of the SCF FSMs occur concurrently and asynchronously as SCF related events occur, which explains the need for a single entity that performs the task of creation, invocation and maintenance of the SCF FSMs. In particular, the SCME Control performs the following tasks:

1 Interprets the input messages from other FEs and translates them into corresponding SCSM events;

2 Translates the SCSM outputs into corresponding messages to other FEs;

3 Performs some asynchronous (with call processing) activities (one such activity is flow control). It is the responsibility of the SCME-control to detect nodal overload and send the Overload Indication (e.g., Automatic Call Gap) to the SSF to place flow control on queries. Other such activities include non-call associated treatment due to changes in Service Filtering, Call Gapping, or Resource Monitoring status and also provision of resource status messages to the SSF;

4 Supports persistent interactions between the SCF and other FEs; and

5 Performs asynchronous (with call processing) activities related to management and supervisory functions in the SCF and creates an instance of a SCME FSM. For example, the SCME provides the non-call associated treatment due to changes in Service Filtering. Therefore, the SCME-Control separates the SCSM from the Service Filtering by creating instances of SCME FSMs for each context of related operations.

The different contexts of the SCME FSMs may be distinguished based on the address information provided in the initiating operations. In the case of service filtering, this address information is given by Filtering Criteria, i.e., all ActivateServiceFiltering operations using the same address, address the same SCME FSM handling this specific service filtering instance. For example, ActivateServiceFiltering operations providing different Filtering Criteria cause the invocation of new SCME FSMs.

Finally, the FEAM relieves the SCME of low-level interface functions. The functions of the FEAM include:

1 Establishing and maintaining interfaces to the SSF, SRF,SDF,CUSF and co-operating SCFs;

2 Passing and queuing (when necessary) the messages received from the SSF, SRF,SDF,CUSF and co-operating SCF to the SCME Control;

3 Formatting, queuing (when necessary), and sending the messages received from the SCME Control to the SSF, SRF, SDF, CUSF and co-operating SCF.

Note that although the SCSM includes a state and procedures concerning queue management, this type of resource management only represents one way of managing network call queues. Another alternative is to let the SSF/CCF manage call queues; however, the technical details of how the SSF/CCF performs queue management is beyond the scope of IN. As such, the Resource Control Object (RCO) (see subclause 12.4.9) and the queuing state of the SCSM (State 2.3), along with its relevant sub-states, events and procedures, are only required and applicable in the case where queue management is performed in the SCF.

Due to the unique nature of the Disconnect and Abandon event, it can be received in any state. Because of this, it is not modelled in all conditions.

13.4 Partial SCME State Transition Diagram

The key parts of SCME State Diagram are described in figures 12-6, 12-7, and 12-8

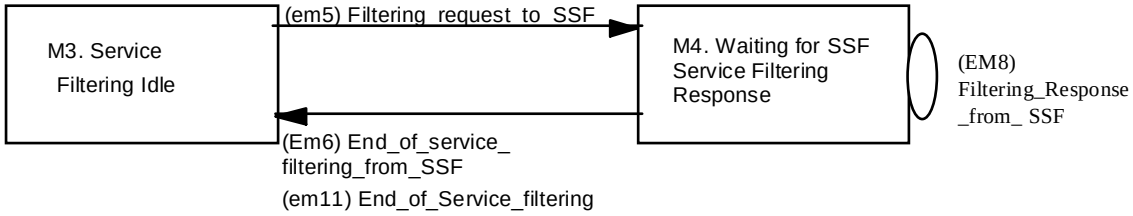


Figure 12-6: The Service Filtering FSM in the SCME

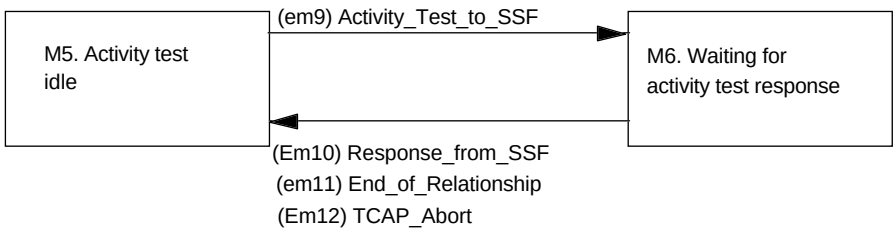


Figure 12-7: The Activity Test FSM in the SCME

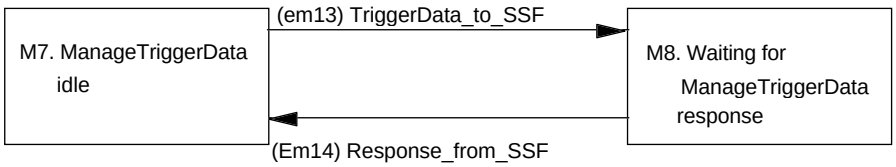


Figure 12-8: The ManageTriggerData FSM in the SCME

The SCME handles the following operations:

ActivateServiceFiltering;
ServiceFilteringResponse;
CallGap;
ActivityTest;
ManageTriggerData.

Issuing the **CallGap** operation does not cause state transitions in the SCME. The rest of the above operations are described below.

The operations that are not listed above do not affect the state of the SCME; these operations are passed to the relevant SCSM.

13.4.1 State M1:

No state M1 supported.

13.4.2 State M2:

No state M2 supported.

13.4.3 State M3: "Service filtering idle"

The following event is considered in this state:

(em5) Filtering_Request_to_SSF
This is an internal event, caused by SL's need to filter service requests to the SSF, and by transmission of the ActivateServiceFiltering operation. This event causes a transition to state M4, waiting for SSF service filtering response.

NOTE: Events are enumerated, and the number of an event is prefixed with either the letter "E" (for external events) or "e" (for internal ones) and included in parentheses in the beginning of the event name. The scope of event names and numbers is defined by the state machine in which these events appear; the same applies to state names.

13.4.4 State M4: "Waiting for SSF service filtering response"

In this state, the SCF is waiting for the service filtering response from the SSF. The following events are considered in this state:

(Em6) End_of_Service_Filtering_Response_from_SSF
This is an external event, caused by reception of the response at the end of the service filtering duration to the request service filtering previously issued to the SSF. This event causes a transition out of this state to state M3, service filtering idle;

(em7) End_of_Service_Filtering
This is an internal event, caused by the expiration of service filtering duration timer in the SCF. This event causes a transition to state M3, service filtering idle.

(em8) Filtering_Response_from_SSF
This is an external event, caused by reception of the response to the request service filtering operation previously issued to the SSF. This event does not cause a transition out of this state, and the SCME remains in state M4, waiting for SSF service filtering response.

When service filtering is active, another service filtering operation could be sent to the SSF that has the same filtering criteria; this second "filter" replaces the first one.

13.4.5 State M5: "Activity test idle"

The following event is considered in this state:

(em9) Activity_test_to_SSF
This is an internal event, caused by caused by the expiration of activity test timer in the SCF, and by transmission of the ActivityTest operation. This event causes a transition to state M6, waiting for activity test response.

NOTE: Activity test also applicable for SCF-SCF and SCF - CUSF interfaces.

13.4.6 State M6: "Waiting for activity test response"

In this state, the SCF is waiting for the activity test response from the SSF. The following events are considered in this state:

(Em10) Activity_Test_Response_from_SSF
This is an external event, caused by reception of the response to the activity test previously issued to the SSF. This event causes a transition out of this state to state M5, activity test idle;

(em11) End_of_Relationship
This is an internal event, caused by the expiration of ActivityTest operation timer in the SCF. This event causes a transition to state M5, activity test idle.

(Em12) TC_Abort
This is an external event, caused by reception of a P-Abort from TC in response to the ActivityTest operation previously issued to the SSF/SCF/CUSF. This event causes a transition to state M5, activity test idle.

NOTE: Activity test also applicable for SCF-SCF and SCF - CUSF interfaces.

13.4.7 State M7: "ManageTriggerData idle"

The following event is considered in this state:

(em13) TriggerData_to_SSF
This is an internal event, caused by transmission of the ManageTriggerData operation. This event causes a transition to state M8, waiting for ManageTriggerData response.

13.4.8 State M8: "Waiting for ManageTriggerData response"

In this state, the SCF is waiting for the ManageTriggerData response from the SSF. The following events are considered in this state:

(em14) Response_from_SSF
This is an external event, caused by reception of the response to the ManageTriggerData previously issued to the SSF. This event causes a transition out of this state to state M7, ManageTriggerData idle;

13.4.9 The RCO

The RCO is part of the SCF management entity that controls data relevant to resource information.

The RCO consists of:

1 A data structure that (by definition) resides in the SDF and can be accessed only via the RCO's methods; and

2 The RCO methods.

For purposes of this ITU-T Recommendation, no implementation constraints are placed on the structure. The only requirement to the structure is that, for each supported resource, it:

1 Stores the resource's status (e.g. busy or idle); and

2 Maintains the queue of SCSMs that are waiting for this resource. For continuous monitoring, the RCO maintains its knowledge of the status of the resources through use of the request every status change report operation.

The following three methods are defined for the RCO:

1 Get_Resource: This method is used to obtain the address of an idle line on behalf of an SCSM. If the resource is busy, the SCSM is queued for it;

2 Free_Resource: This method is used when a disconnect notification from the SSF is received. The method either advances the queue (if it is not empty) or marks the resource free (otherwise); and

3 Cancel: This method is used when either the queuing timer has expired or the call has been abandoned.

13.5 The SCSM

In the SCSM, there are four kinds of FSMs: SSF/SRF, SDF, SCF and CUSF related states.

The SCSM instance transmits all events from SCME-Control to the appropriate FSM (for SSF/SRF, SDF, SCF, CUSF or for SCME). If the FSM instance does not exist yet, the SCME-Control creates it and transmits the event to it. The SSF/SRF FSM will then create instances of the FSM for CSA, for Specialized Resource, for Handed-off SSF or Assisting SSF, as required. The FSM for CSA will create instances of FSMs for Call Segments, as required.

General rules applicable to more than one state are as follows:

In every state, if there is an error in a received operation, the SLP and the maintenance functions are informed. Generally the SCSM remains in the same state; however, different error treatment is possible in specific cases as described in subclause 16. Depending on the class of the operation, the error can be reported to the SSF, SRF, or SDF (see ETS 300 287-1 [7]).

The general rules for one or a sequence of components sent in one or more TC messages, which may include a single operation or multiple operations, are specified in 11.5 SSF state transition diagram (they are not described here).

13.5.1 SSF/SRF Related states (SCSM-SSF/SRF)

SSF/SRF related states are contained in the FSMs for SSF/SRF interface, for CSA, for Call Segment, for specialized Resource, for Handed off SSF and for Assisting SSF. The interactions between these FSMs are shown in figure 12-9.

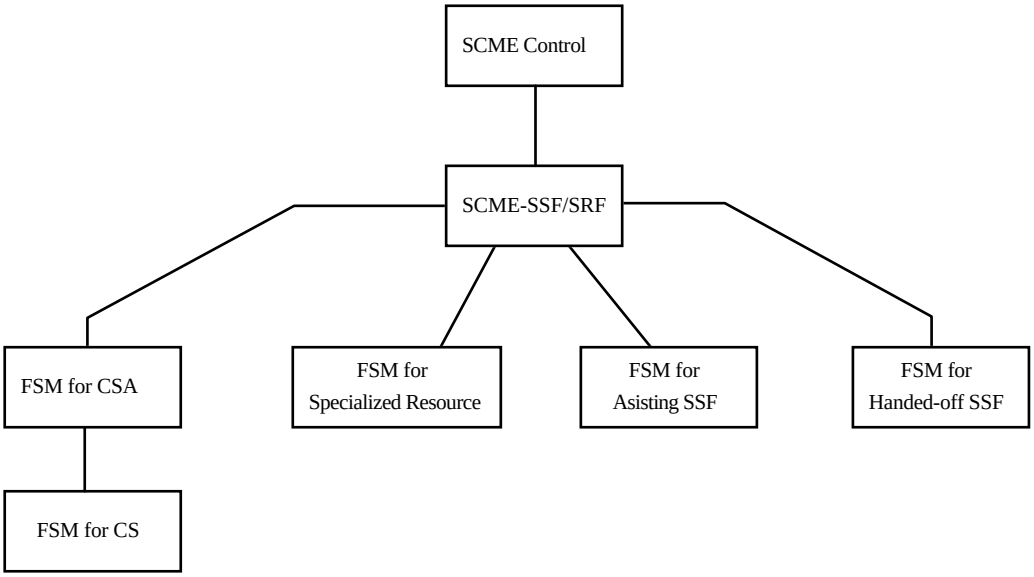


Figure 12-9: FSM interactions for SCSM-SSF/SRF

The call-control-related operations relevant to the SCF-SSF-interface (except the SCME related operations) are categorized into

1 Call-processing-related operations; and

2 Non-call-processing-related operations.

Call-processing-related operations are grouped into following two sets:

CollectInformation
Connect
Continue
ContinueWithArgument

and

InitiateCallAttempt
ConnectToResource
DisconnectForwardConnection
DisconnectForwardConnectionWithArgument
EstablishTemporaryConnection
ReleaseCall

For the first set of call-processing operations, the SCF may in general not send two operations of the same set in a series of TC messages or in a component sequence to the same CS instance in the SSF , but send them only one at a time. Two operations of the first set shall in general be separated by at least one EDP-R message received by the SCSM or the sending of a CPH operation (e.g. SplitLeg). The same applies for any operation of the first set followed by ConnectToResource or EstablishTemporaryConnection.

The non-call-processing operations include the rest of the operations at the SCF-SSF interface (but not the SCME related operations). When the SL needs to send operations in parallel, they are sent in the component sequence.

In the following, each FSM is described. The letter, selected from "S", "I", "C","R", "H", and "A", which is prefixed to the state numbers and the event numbers, indicates the FSM for SSF/SRF interface, for CSA, for Call Segment, for Specialized Resource, for Handed-off SSF, and for Assisting SSF, respectively.

13.5.1.1 FSM for SSF/SRF interface

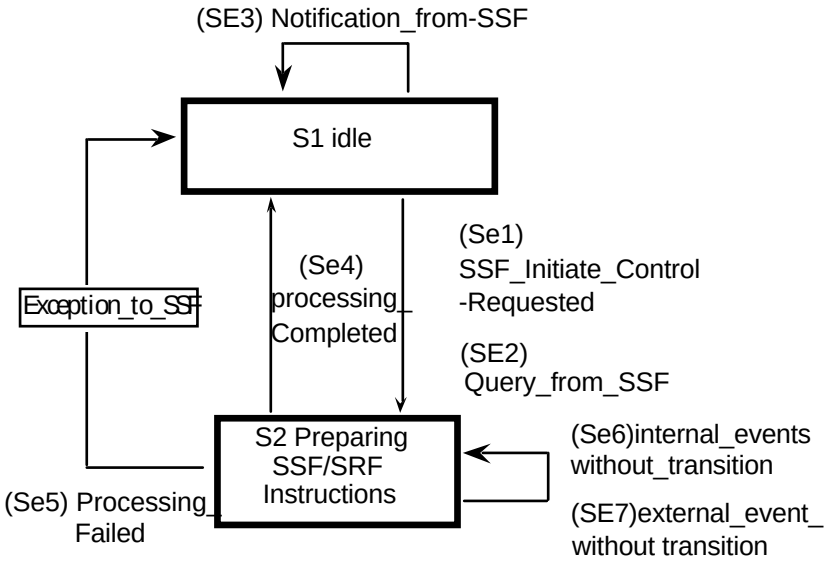


Figure 12-10: FSM for SSF/SRF interface

Figure 12-10 shows the general State Diagram of the FSM for SSF/SRF interface as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following sub-subclauses. The state of Preparing SSF/SRF Instructions has internal sub-FSMs composed of the sub-states.

The FSM for SSF/SRF interface has an application timer, $T_{ASSIST/HAND-OFF}$, whose purpose is to prevent excessive assist/hand-off suspension time. The FSM for SSF/SRF interface sets the timer $T_{ASSIST/HAND-OFF}$ when the SCSM sends the **EstablishTemporaryConnection** or **Connect** operation with a correlation ID. This timer is stopped when the FSM for SSF/SRF interface receives the **AssistRequestInstructions** operation from the assisting/handed-off SSF or assisting SRF. On expiration of $T_{ASSIST/HAND-OFF}$, the FSM for SSF/SRF interface informs SLPI and the maintenance functions, and the FSM for SSF/SRF interface remains in the "**Preparing SSF/SRF Instructions"** state.

13.5.1.1.1 State S1: "Idle"

The following events are considered in this state:

(Se1) SSF_Initiate_Control_Requested
This is an internal event caused by the SL's need to have a new control relationship with SSF. The FSM for CSA requests to transmit the **InitiateCallAttempt** operation to the SSF. This event causes a transition to the state S2, **Preparing SSF/SRF Instructions.**

(SE2) Query_from_SSF
This is an external event caused by a reception of the following operation:

InitialDP (for TDP-R).

This event causes a transition to the state S2, **Preparing SSF/SRF Instructions**. The FSM for SSF/SRF interface creates a new FSM for CSA instance, and transmits this event to the FSM.

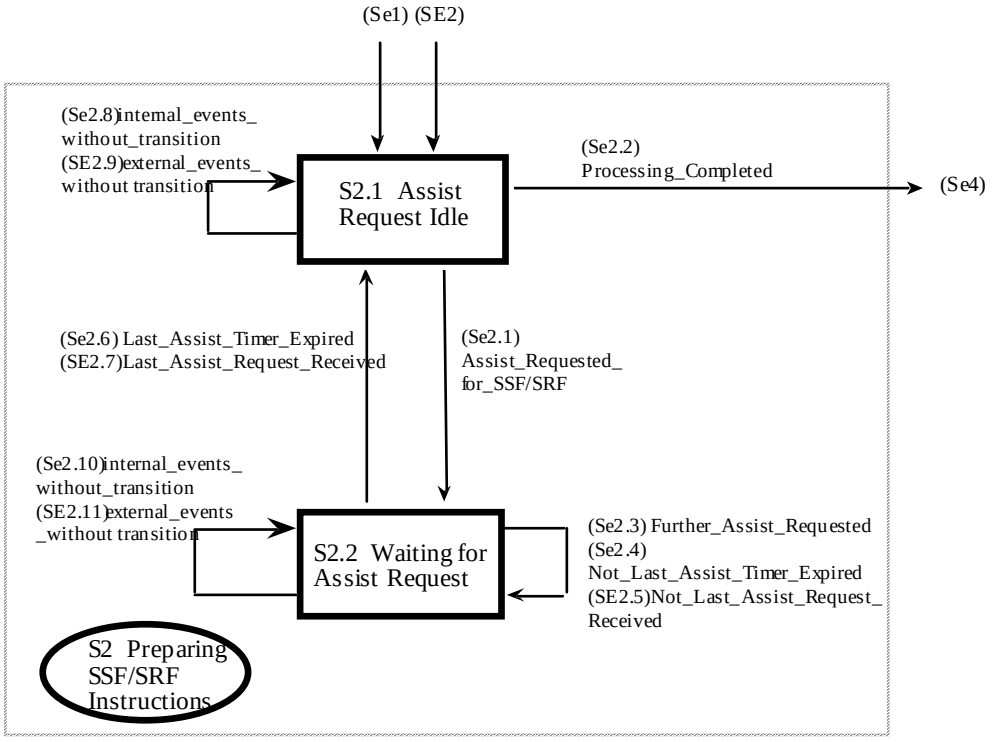


Figure 12-11: Partial expansion of state S2 FSM

13.5.1.1.2 State S2: "Preparing SSF/SRF Instructions"

The following events are considered in this state:

(Se4) Processing_Completed
This is an internal event caused by the end of service. In this case, the SCF has completed the processing for SSF and SRF. This event causes a transition to the state S1, **Idle**.

(Se5) Processing_Failed
This (internal) event causes an appropriate exception processing and a transition back to the state S1, **Idle**.

NOTE: Here and further in the present document, the exception processing is not defined. However, it is assumed that it needs to include releasing all the involved resources and sending an appropriate response message to the SSF. This implies that all substates handle event Se5, but it is not modelled.

(Se6) internal_events_without_transition
This is an internal event caused by the SLPI or the other associate FSM instances. The FSM for SSF/SRF interface may send an operation to the corresponding FE. In this case, associate FSMs still exist. This event causes a transition to the same state.

(SE7) external_events_without transition
This is an external event caused by receiving an event from the other FEs. The FSM for SSF/SRF interface will process the event and, if necessary, pass the event to the relevant associate FSMs. In this case, associate FSMs still exist. This event causes a transition to the same state.

In this state, any events received by the FSM for SSF/SRF interface which are relevant to associate FSMs are sent to those FSMs.

In this state, when all associate FSMs are released and there are no pending application timers, $T_{ASSIST/HAND-OFF}$, the FSM for SSF/SRF interface instance transits to the state 1, **Idle** and is released. However, when all associate FSMs are released and there is a pending application timer, $T_{ASSIST/HAND-OFF}$, the FSM for SSF/SRF interface instance remains in this state.

To describe further the procedures relevant to this state, this state is divided into two sub-states, which are described in the following two subclauses. (This subdivision is illustrated in figure 12-11.)

12.5.1.1.2.1 State S2.1: "Assist Request Idle"

The following events are considered in this state:

(Se2.1) Assist_Requested_for_SSF/SRF
This is an internal event caused by necessity of a new relationship with SSF or SRF for user interaction. In this case, SCSM sends one of the following operations to initiating SSF with an SRF address or an SSF address for routeing:

EstablishTemporaryConnection (for Assisting SSF/SRF);
Connect (for Handed-off SSF).

This event causes a transition to the state S2.2, **Waiting for Assist Request.** The FSM for SSF/SRF interface starts the timer $T_{ASSIST/HAND-OFF}$.

(Se2.2) Processing_Completed
This is an internal event. This event causes the transition that maps into the FSM event (Se4) for SSF/SRF interface .

(Se2.8) internal_events_without_transition
This is an internal event caused by the SLPI or the other associate FSM instances. The FSM for SSF/SRF interface may send an operation (except an **EstablishTemporaryConnection** operation and a **Connect** operation for hand-off) to the corresponding FE. In this case, associate FSMs still exist. This event causes a transition to the same state.

(SE2.9) external_events_without transition
This is an external event caused by receiving an event from the other FEs. The FSM for SSF/SRF interface will process the event and, if necessary, pass the event to the relevant associate FSMs. In this case, associate FSMs still exist. This event causes a transition to the same state.

12.5.1.1.2.2 State S2.2: "Waiting for Assist Request"

In this state, the FSM for SSF/SRF waits for the **AssistRequestInstructions** operation from the Handed-off/Assisting SSF (SSF relay case) or from the SRF (Direct SCF-SRF case). The following events are considered in this state:

(Se2.3) Further_Assist_Requested
This is an internal event caused by necessity of a new relationship with SSF or SRF for user interaction. In this case, SCSM sends one of the following operations to initiating SSF with an SRF address or an SSF address for routeing:

EstablishTemporaryConnection (for Assisting SSF/SRF)
Connect (for Handed-off SSF)

This event causes a transition to the same state. The FSM for SSF/SRF interface starts the new timer $T_{ASSIST/HAND-OFF}$.

(Se2.4) Not_Last_Assist_Timer_Expired
This is an internal event caused by the expiration of the timer $T_{ASSIST/HAND-OFF}$ which is one of pending timers. In this case, the FSM for SSF/SRF interface informs the SLPI, and remains in the same state.

(SE2.5) Not_Last_Assist_Request_Received
This is an external event caused by a reception of the **AssistRequestInstructions** operation. In this case, the FSM for SSF/SRF interface stops the corresponding timer, creates a new FSM instance (instance of FSM for Specialized Resource, Handed-off SSF, or Assisting SSF), and transmits the event to the new FSM instance. The FSM for SSF/SRF interface remains in the same state.

(Se2.6) Last_Assist_Timer_Expired
This is an internal event caused by the expiration of the timer $T_{ASSIST/HAND-OFF}$ which is the last pending timer. In this case, the FSM for SSF/SRF interface informs the SLPI, and transits to the state S2.1, **Assist Request Idle**. Any other pending timers are ignored.

(SE2.7) Last_Assist_Request_Received

This is an external event caused by a reception of the **AssistRequestInstructions** operation. In this case, the FSM for SSF/SRF interface stops the corresponding timer, creates a new FSM instance (instance of FSM for Specialized Resource, Handed-off SSF, or Assisting SSF), and transmits the event to the new FSM instance. The FSM for SSF/SRF interface transits to the state S2.1, **Assist Request Idle**.

(Se2.10) internal_events_without_transition

This is an internal event caused by the SLPI or the other associate FSM instances. The FSM for SSF/SRF interface may send an operation (except an **EstablishTemporaryConnection** operation and a **Connect** operation for hand-off) to the corresponding FE. In this case, any associate FSMs which are waiting for the **AssistRequestInstructions** operation still exist. This event causes a transition to the same state.

(SE2.11) external_events_without transition

This is an external event caused by receiving an event (except an **AssistRequestInstructions** operation) from the other FEs. The FSM for SSF/SRF interface will process the event and, if necessary, pass the event to the relevant associate FSMs. In this case, any associate FSMs which are waiting for the **AssistRequestInstructions** operation still exist. This event causes a transition to the same state.

13.5.1.2 FSM for CSA

Figure 12-12 shows the general State Diagram of the FSM for CSA as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

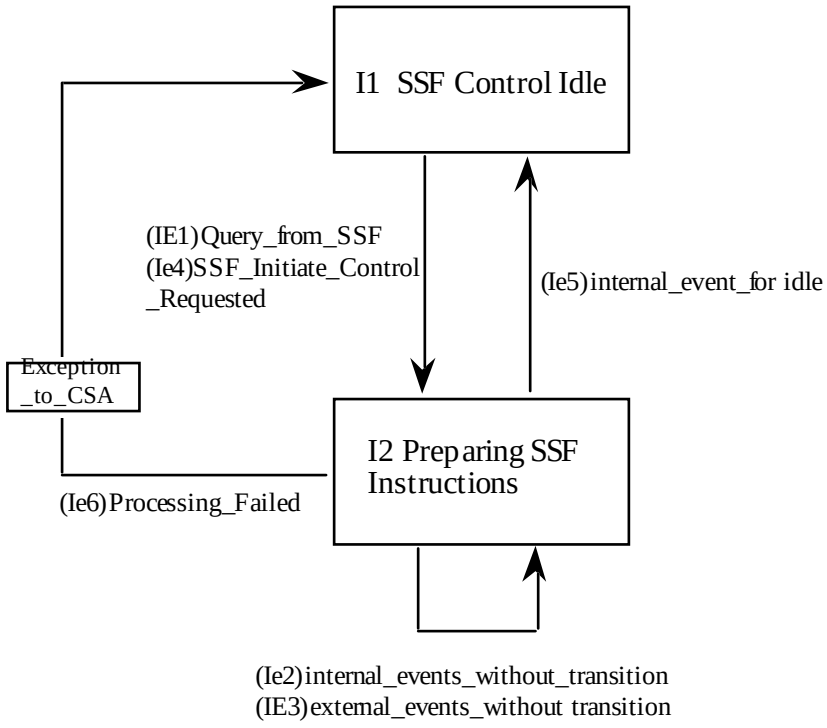


Figure 12-12: FSM for CSA

The FSM for CSA receives external events from the FSM for SSF/SRF interface and either processes them directly or passes them to the relevant associate FSM for CS. It receives internal events from the SLPI or an associate FSM for CS instructing it to send operations to the FSM for SSF/SRF interface for sending to external FEs. It will also receive internal notification of operations sent by the FSM for SSF/SRF interface which affect the FSM for CSA

13.5.1.2.1 State I1: "SSF Control Idle"

The FSM for CSA enters the SSF Control Idle state when one of the following occurs:

- when all the FSM for Call Segment instances associated with the FSM for CSA instance are released.

When the FSM for CSA enters the SSF Control Idle state, the associate FSM for SSF/SRF interface must be notified.

The following events are considered in this state:

(IE1) Query_from_SSF
This is an external event caused by a reception of the following operation:

InitialDP (for TDP-R).

This event causes a transition to the state I2, **Preparing SSF Instructions**. The FSM for CSA creates a new FSM for Call Segment instance, and transmits this event to the FSM.

(Ie4) SSF_Initiate_Control_Requested
This is an internal event caused by the SL's need to have a new control relationship with SSF. This event occurs in the following cases.

- The FSM for Call Segment requests the FSM for CSA to transmit the **InitiateCallAttempt** operation to the SSF.

- The FSM for Call Segment requests the FSM for CSA to transmit the **CreateCallSegmentAssociation** operation to the SSF.

This event causes a transition to state I2, **Preparing SSF Instructions.**

13.5.1.2.2 State I2: "Preparing SSF Instructions"

In this state, the FSM for CSA instance handles instructions from the SCF and events which are received from the FSM instances for CS or the FSM for SSF/SRF interface.

The following events are considered in this state:

(Ie2) internal_events_without_transition
This is an internal event caused by the following cases.

- When the SLPI instructs the FSM for CSA instance to send the following operations to the FSM for SSF/SRF interface:

FurnishChargingInformation
Cancel (allRequests)
ReleaseCall
MoveLeg
SplitLeg

NOTE 1: In this case, the SSF creates a Call Segment and connects the split Leg with the Call Segment. The FSM for CSA transmits the event to the FSM for the "source" CS instance. Furthermore, the FSM for CSA creates a new FSM for the new Call Segment instance and transmits the event to the FSM.

MergeCallSegments

NOTE 2: In this case, the SSF deletes the "source" Call Segment and connects the Leg in the source Call segment with the "target" Call Segment. The FSM for CSA transmits the event to the FSM for the "source" CS instance and releases the FSM instance. Furthermore, the FSM for CSA transmits the event to the FSM for the "target" CS instance.

RequestReportBCSMEvent

- When the FSM for SSF/SRF interface has sent the following operation:

  MoveCallSegments

- When the associate FSM for CS instance requests the sending of the following operations:

  ApplyCharging
  CallInformationRequest
  RequestNotificationChargingEvent
  SendChargingInformation
  ResetTimer
  Cancel(invokeID)
  ConnectToResource
  EstablishTemporaryConnection
  DisconnectForwardConnection
  DisconnectForwardConnectionWithArgument
  PlayAnnouncement
  PromptAndCollectUserInformation
  PromptAndReceiveMessage
  InitiateCallAttempt
  Connect
  CollectInformation
  ScriptInformation
  ScriptRun
  ScriptClose
  Continue
  ContinueWithArgument
  RequestReportBCSMEvent
  Release Call
  DisconnectLeg

- When the application timer Tassist/hand-off in the FSM for SSF/SRF interface expires.

In this case, any associate FSMs still exist. This event causes a transition to the same state.

(IE3) external_events_without transition
This is an external event caused by receiving an event from the other FEs. The FSM for CSA will process the event and, if necessary, pass the event to the relevant associate FSMs.

EventReportBCSM
CallInformationReport
ApplyChargingReport
EntityReleased
EventNotificationCharging
SpecializedResourceReport
ReturnResult from PromptAndReceiveMessage
ScriptEvent
ReturnResult for PromptAndCollectUserInformation.
AssistRequestInstruction

In this case, any associate FSMs still exist. This event causes a transition to the same state.

(Ie5) internal_event_for idle
This is an internal event caused by the following cases.

- When the last FSM for CS instance transits to the idle state.

  In this case, any associate FSMs no longer exist. This event causes a transition to the state I1, **SSF Control Idle**.

(Ie6) Processing_Failed
This (internal) event causes an appropriate exception processing and a transition to the state I1, **SSF Control Idle**.

13.5.1.3 FSM for Call Segment

Figure 12-13 shows the general state diagram of the FSM for Call Segment as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

The following operations (the CPH operations) are of class 1 and require reception of a Return Result:

DisconnectLeg
MergeCallSegments
MoveCallSegments
MoveLeg
SplitLeg.

Reception of the Return Result for the operations mentioned above does not result in a state transition in the FSM for CS. Furthermore, this Return Result may be received in any state of the FSM for CS.

Reception of Return Result may support keeping an accurate CV in SCF and decide what subsequent action can be taken by the SL. If various components (e.g. including CPH operations) are grouped into a single TC message the return result will provide the invoke ID. This allows the SL to correlate the result to a previously sent CPH operation and hereby distinguish between the grouped components (operations) sent.

The FSM for CS has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to restart the timer, $T_{SSF,}$ to guard the release of the call segment which is waiting for instructions from the SCF. The use of timer $T_{SCF\text{-}SSF}$ for the FSM for CS is mandatory.

The timer $T_{SCF\text{-}SSF}$ is set in the following cases:

i) when the SCF receives an **InitialDP** operation. In this case, this timer is restarted when a first request, other than **ResetTimer** operation, is sent to the SSF. On the expiration of timer $T_{SCF\text{-}SSF,}$ the FSM for CS may restart T_{SSF} once using the **ResetTimer** operation, and restart timer $T_{SCF\text{-}SSF}$. On the second expiration of $T_{SCF\text{-}SSF}$, the FSM for CS informs the SLPI and the maintenance functions, and the FSM for CS transits to the state C1, **CS Control Idle**;

ii) when the FSM for CS instance enters the **Preparing CS Instructions** state (C2.1) under any other condition than the case i). In this case, the FSM for CS may restart T_{SSF} using the **Reset Timer** operation any number of times;

iii) when the FSM for CS enters the Queuing substate (see subclause 12.5.1.3.2.3, State C2.3: "**Queuing**"). In this case, on the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for CS may restart T_{SSF} using the **ResetTimer** operation any number of times; and

iv) when the SCF enters the **Suspended and User Interaction** state (C3). In this case, on the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for CS may restart T_{SSF} using the **ResetTimer** operation any number of times

In each of the above cases, $T_{SCF\text{-}SSF}$ may have different values as defined by the application. The values of $T_{SCF\text{-}SSF}$ are smaller than the respective values of T_{SSF}.

When receiving or sending any other operation, the SCF should restart $T_{SCF\text{-}SSF.}$. Whenever the value of T_{SSF} is changed by the SCF sending a ResetTimer operation to the SSF, the value of $T_{SCF\text{-}SSF}$ must be changed accordingly.

13.5.1.3.1 State C1: "CS Control Idle"

The FSM for CS must enter the **CS Control Idle** state when the associate FSM for CSA instance transits to the **SSF Control Idle** state.

When the FSM for CS enters the **CS Control Idle** state, the associate FSM for CSA shall be notified.

The following events are considered in this state:

(Ce1) New_Call_Segment_from_SLPI
This is an internal event caused by the SLPI when there is a need to send the following operation to the SSF.

InitiateCallAttempt.

This event causes a transition to the state C2, **Preparing CS Instructions**.

(CE2) Query_from_SSF
This is an external event caused by a reception of the following operation from the SSF.

InitialDP.

This event causes a transition to the state C2, **Preparing CS Instructions**.

(Ce3) New_Call_Segment_for_Split
This is an internal event caused by the SLPI when there is a need to send the following operation to the SSF.

SplitLeg(target CS).

This event causes a transition to the state C2, **Preparing CS Instructions**.

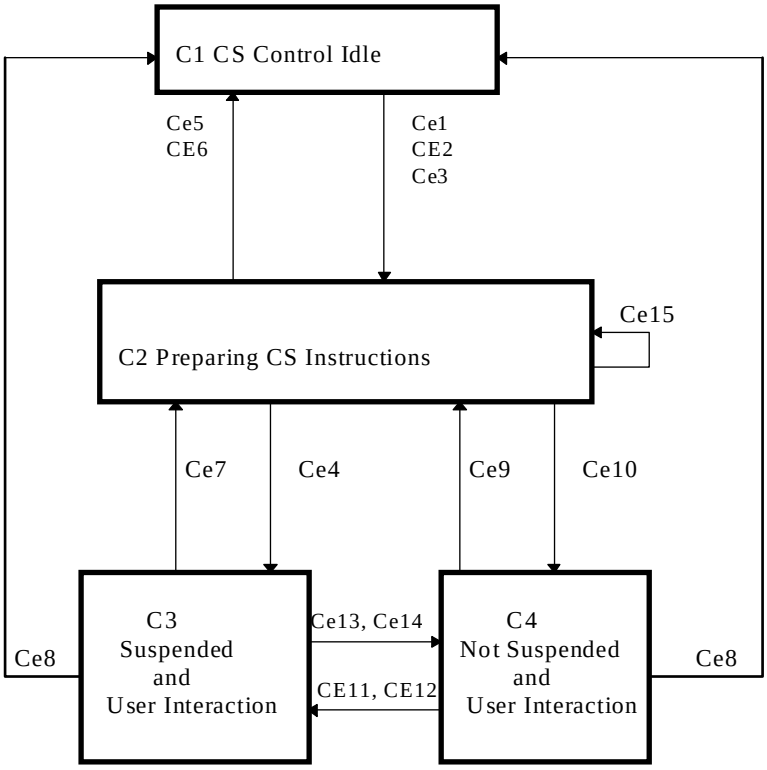


Figure 12-13: FSM for Call Segment

The SCF state diagram for the CS contains the following transitions (events):

Ce1 New_CS_Creation_from_SLPI

CE2 Query_from_SSF

Ce3 New_CS_Creation_for_Split

Ce4 SR_Fac._Needed (Call Processing Suspended)

Ce5 Processing_Completed

CE6 Last_Event_Received

Ce7 Continue_SCF_Processing

Ce8 Processing_Completed_And_User_Interaction

Ce9 CS_Monitoring_Needed

Ce10 SR_Facilities_Needed (Call Processing Not Suspended)

CE11 CS_Instruction_Needed_During _Established_Temporary_Connection

CE12 CS_Instruction_Needed_During_User_Interaction

Ce13 CS_Monitoring_Needed_During_Established_Temporary_Connection

Ce14 CS_Monitoring_Needed_During_User_Interaction

Ce15 Preparing CS Instructions state no change

13.5.1.3.2 State C2: "Preparing CS Instructions"

The following events are considered in this state:

(Ce4) SR_Facilities_Needed (Call Processing Suspended)
This is an (internal) event caused by the SL's need for additional information from the call party; hence the necessity to set up a connection between the call-party and the SRF. This event causes a transition to state C3, **Suspended and User Interaction**.

(Ce5) Processing_Completed
This is an internal event, caused by the end of processing for the CS. This event causes a transition to the state C1, **CS Control Idle**.

(CE6) Last_Event_Received
This is an external event caused by a reception of a last event from the SSF.

(Ce7) Continue _SCF_Processing
This is an internal event caused by the necessity of continuing SCF processing. In this case, the SCF has obtained all the information from the SRF that is needed to instruct the SSF to continue with the call.

(Ce9) CS_Monitoring_Needed
This is an internal event, caused by the necessity of monitoring. In this case, the SCF has obtained all the information from the SRF that is needed to instruct the SSF to continue with the call.

(Ce10) SR_Facilities_Needed (Call Processing not Suspended)
This is an (internal) event caused by the SL's need for additional information exchange with the call party, while waiting for request or notification from the SSF. There is therefor the necessity to set up a connection between the call party and the SRF. This event causes a transition to state C4, Not Suspended **and User Interaction**.

(Ce15) Preparing CS Instructuins no change
This is an internal event caused by the processing of e.g. a CPH operation like SplitLeg (source CS) , MoveLeg (source and target CS) Disconnect leg (not last leg in CS) and MergeCallSegments.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in figure 2-12.5.1.3.

jmm *which* figure??

13.5.1.3.2.1 State C2.1: "Preparing CS instructions"

The following events are considered in this state:

(Ce2.1) Non-Call_Processing_Instructions
This is an internal event caused by the following cases.

- When the SL needs to send an operation such as the following to the SSF.

  ApplyCharging
  CallInformationRequest
  RequestNotificationChargingEvent
  SendChargingInformation

- When the following operations have been sent to the SSF by an associate FSM for CSA:

  Cancel (allRequests)
  RequestReportBCSMEvent

- When the application timer $T_{SCF\text{-}SSF}$ expires. In this case, the FSM for CS sends a ResetTimer operation to the corresponding CS FSM in the SSF.

This event causes a transition back to state C2.1, **Preparing CS Instructions**.

(Ce2.2) SR_Facilities_Needed
This is an internal event, caused by the SL when there is a need to use the SRF. This event is mapped as the FSM event (Ce4).

(Ce2.3) Call_Processing_Instruction_Ready (monitoring not required)
This is an internal event caused by the SL when the final call-processing-related operation is ready and there is neither an armed EDP nor an outstanding **CallInformationReport** or **ApplyChargingReport** operation. It causes one of the following operations to be issued to the SSF:

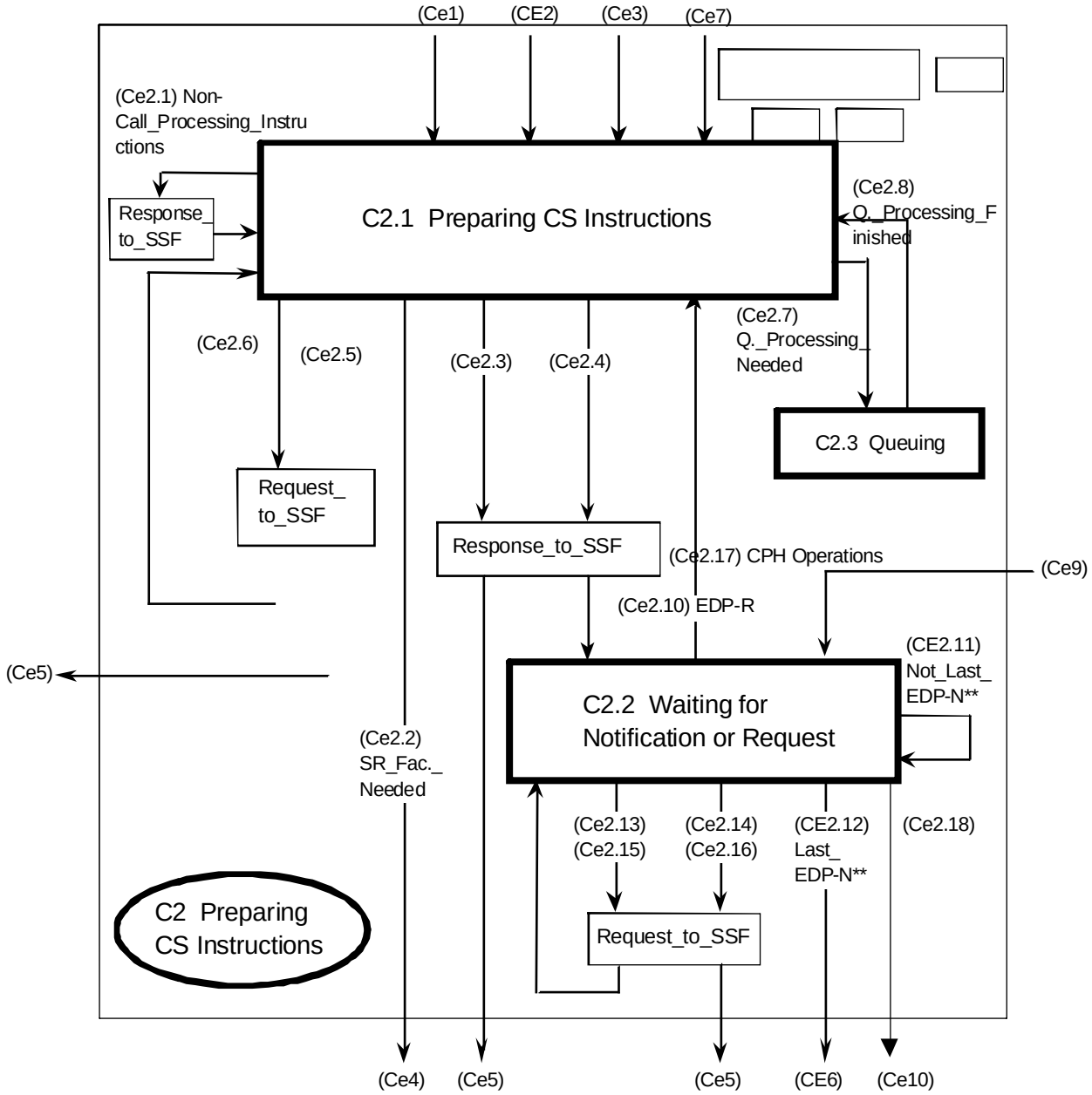
CollectInformation
Connect
Continue
(Only applicable for a single CS . Use of this operation is not valid in a multi call segment CSA.)

ContinueWithArgument
ReleaseCall

In addition, one or more of the following operations may be issued to the SSF prior to the operations listed above:

Cancel (allRequests)
RequestReportBCSMEvent (to disarm all the armed EDPs)
RequestNotificationChargingEvent(to set mode transparent for charging events)
SendChargingInformation

This event is mapped as the FSM event (Ce5).



(Ce2.3): Call_Processing_Instruction_Ready (Monitoring* not Required)
(Ce2.4): Call_Processing_Instruction_Ready (Monitoring* Required)
(Ce2.5): CS_or_Leg_Control_Last_Instruction
(Ce2.6), (: CS_or_Leg_Control_Continuing_Instruction
(Ce2.13): Notification_or_Request_Continuing_Instruction
(Ce2.14): Monitoring_Cancel_Instruction
(Ce2.15): Release_Call_Instruction (Call Information Report or Apply Charging Report has been requested)
(Ce2.16): Release_Call_Instruction (Neither Call Information Report nor Apply Charging Report has been requested)
(Ce2.17) CPH Operations

Figure 12-14: Partial expansion of state C2 FSM for CS

Editor's note: Figure 12-14 contains 4 boxes without any text in upper right corner, These boxes should be deleted but the figure supplied is uneditable.

(Ce2.4) Call_Processing_Instruction_Ready (monitoring required)

This is an internal event caused by the SL when a call-processing-related operation is ready and the monitoring of the call is required (e.g. an EDP is armed, or there is an outstanding **CallInformationReport** or **ApplyChargingReport**). It causes one of the following operations to be issued to the SSF:

CollectInformation
Connect
Continue (Only applicable for a single CS . Use of this operation is not valid in a multi call segment CSA.)
ContinueWithArgument
ReleaseCall

In addition, one or more of the following operations may be issued to the SSF prior to the operations listed above:

ApplyCharging
CallInformationRequest
RequestReportBCSMEvent
RequestNotificationChargingEvent
SendChargingInformation

This event causes a transition to the state C2.2, **Waiting for Notification or Request**.

(Ce2.5) CS_or_Leg_Control_Last_Instruction

This event is an internal event caused by the following cases.

- When the SLPI needs to send an operation such as the following to the SSF:

  DisconnectLeg (for last leg).
  MoveLeg (for last leg)
- When the following operations have been sent to the SSF by an associate FSM for CSA instance:

  MergeCallSegments (for "source" CS).

This event is mapped as the FSM event (Ce5).

(Ce2.6) CS_or_Leg_Control_Continuing_Instruction

This event is an internal event caused by the following cases.

- When the SLPI needs to send an operation such as the following to the SSF:

  DisconnectLeg (for not last leg).
- When the following operations have been sent to the SSF by an associate FSM for CSA instance:

  MergeCallSegments (for "target" CS)**,**
  SplitLeg (for "source" CS),
  MoveLeg (for not last leg)

Reception of the Return Result for the operation causes a transition back to the same state.

(Ce2.7) Queuing_Processing_Needed

This is an internal event caused by the SL when queuing of the call is required This event causes a transition into State C2.3, **Queuing**.

(Ce2.21) MidCall EDP

This event is an external event caused by a reseption of EventReportBCSM for MidCall EDP if MidCall DP was armed to be reported in "any state".

13.5.1.3.2.2 State C2.2: "Waiting for Notification or Request"

The following events are considered in this state:

(Ce2.10) EDP-R

This is an external event, caused by a reception of the following operation:

EventReportBCSM (for EDP_R)

This event causes a transition to the state C2.1, **Preparing CS Instructions**.

(Ce2.11) Not_Last_EDP-N

This is an external event, caused by a reception of one of the following operations:

ApplyChargingReport
CallInformationReport
EventReportBCSM (for EDP_N)
EventNotificationCharging

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. This event causes a transition back to the state C2.2**, Waiting for Notification or Request**.

Refer to the meaning of "last EDP-N" which is described in the event (CE2.12).

(CE2.12) Last_EDP-N

This is an external event, caused by a reception of one of the following operations:

ApplyChargingReport
CallInformationReport
EntityReleased
EventReportBCSM (for EDP_N)

In this case, there is no outstanding armed EDP and neither pending **CallInformationReport** nor pending **ApplyChargingReport**. This event is mapped as the FSM event (CE6).

NOTE: The "last EDP-N" means that there are no other EDPs which may be encountered when an EDP-N was detected. Some of the EDPs are automatically disarmed if another EDP is encountered. The EDPs which are automatically disarmed depend on which EDP is encountered. An example is the case of the EDPs O_Answer, O_No_Answer, RouteSelectFailure or O_Called_Party_Busy. If any of these EDPs are encountered, all the other EDPs of this list are automatically disarmed.

(Ce2.13) Notification_or_Request_Continuing_Instruction

This event is an internal event caused by the following cases.

- When the SLPI needs to send an operation such as the following to the SSF.

ApplyCharging
FurnishChargingInformation
RequestNotificationChargingEvent
SendChargingInformation

- When the following operations have been sent to the SSF by the FSM for CSA:

RequestReportBCSMEvent (for the purpose of which:

i) one or more of EDPs will be armed,

ii) a part of armed EDPs will be disarmed, or

iii) all of armed EDPs will be disarmed when there are other pending requests);

This event causes a transition back to the state C2.2, **Waiting for Notification or Request**.

(Ce2.14) Monitoring_Cancel_Instruction

This is an internal event caused when the associate FSM for CSA instance has sent one of the following operations to the SSF:

RequestReportBCSMEvent (for the purpose of which all of armed EDPs will be disarmed when there are no more other pending requests)
Cancel (allRequests).

This event is mapped as the FSM event (Ce5).

(Ce2.15) Release_Call_Instruction (Call Information Report or Apply Charging Report has been Requested)

This is an internal event caused when the associate FSM for CSA instance has sent the following operation to the SSF:

ReleaseCall (when there is outstanding CallInformationReport or ApplyChargingReport)

This event causes a transition back to the state C2.2, **Waiting for Notification or Request**.

(Ce2.16) Release_Call_Instruction (neither Call Information Report nor Apply Charging Report has been Requested)
This is an internal event caused when the associate FSM for CSA instance has sent the following operation to the SSF:

ReleaseCall (when there is no outstanding CallInformationReport or ApplyChargingReport)
This event is mapped as the FSM event (Ce5).

(Ce2.17) CPH Operation Instruction
This event is an internal event caused by the following case.

- When the SLPI needs to send an operation such as the following to the SSF:

  DisconnectLeg (for not last leg)
- When the following operations have been send to the SSF by the associate FSM for CSA instance:

  MergeCallSegments (for "target" CS)
  MoveLeg
  SplitLeg (for "source" CS)

Reception of the Return Result causes a transition back to the state C2.1, Preparing CS Instructions

(Ce2.18) SR_Facilities_Needed
This is an internal event, caused by the SL when there is a need to use the SRF. This event is mapped as the FSM event (Ce10).

13.5.1.3.2.3 State C2.3: "Queuing"

When the SCF is processing the query from the SSF/CCF, it may find that the resource to which the call shall be routed is unavailable. One possible reason causing the resource to be unavailable is the "busy" condition.

NOTE: The manner in which the status of the resources is maintained is described in subclause 12.4.7.

Such a resource may be an individual line or trunk or a customer-defined group of lines or trunks. In the latter case, the word "busy" means that all lines or trunks in the group are occupied; and the word "idle" means that at least one line or trunk in the group is idle.

If the resource is busy, the SCF may put the call on queue and resume it later when the resource is idle. The following operations can be sent in this state:

ApplyCharging
CallInformationRequest
RequestReportBCSM Event
RequestNotificationChargingEvent
ResetTimer
SendChargingInformation

The following events are considered in this state:

(Ce2.8) Queuing_Processing_Finished
This is an internal event caused by the SLP when it is ready to prepare the call-related operation for sending to the SSF. This event causes a transition to State C2.1, **Preparing CS Instructions**.

This state further expands into an FSM, which is depicted in figure 3-12.5.1.3.

jmm *which* figure??

This FSM does not explicitly describe all possible combinations of resource monitoring functions used for queuing. The following possibilities may be used in implementations:

- monitoring based on issuing by the SCSM of the **RequestReportBCSMEvent** operation and subsequent reception of the **EventReportBCSM** operation to report the availability of the resource. Both the Request and Report occur in a single different call context. In this case, operations to the SDF or equivalent SCF functionality may be used for scanning the status of resources.

(Ce2.23) MidCallEDP
This event is an external event caused by a reseption of EventReportBCSM for MidCall EDP if MidCall DP was armed to be reported in "any state".

13.5.1.3.2.3.1 State C2.3.1: Preparing CS Instructions

In this state, the FSM for CS prepares the instructions for the SSF to complete the call. The following events are considered in this state:

(Ce2.3.1) Instruction_Ready
This is an internal event that takes place only when the required resource is available. In this case, the FSM for CSA has obtained the address of the free resource via the Get_Resource method of the RCO (see subclause 12.4.7). This event maps into the event (Ce2.8).

(Ce2.3.2) Non-Call_Processing_Instructions
This is an internal event caused by the following cases.

- When the SL needs to send an operation such as the following to the SSF:

  ApplyCharging
  CallInformationRequest
  FurnishChargingInformation
  RequestNotificationChargingEvent
  SendChargingInformation

- When the following operations has been sent to the SSF by the associate FSM for CSA instance:

  RequestReportBCSMEvent

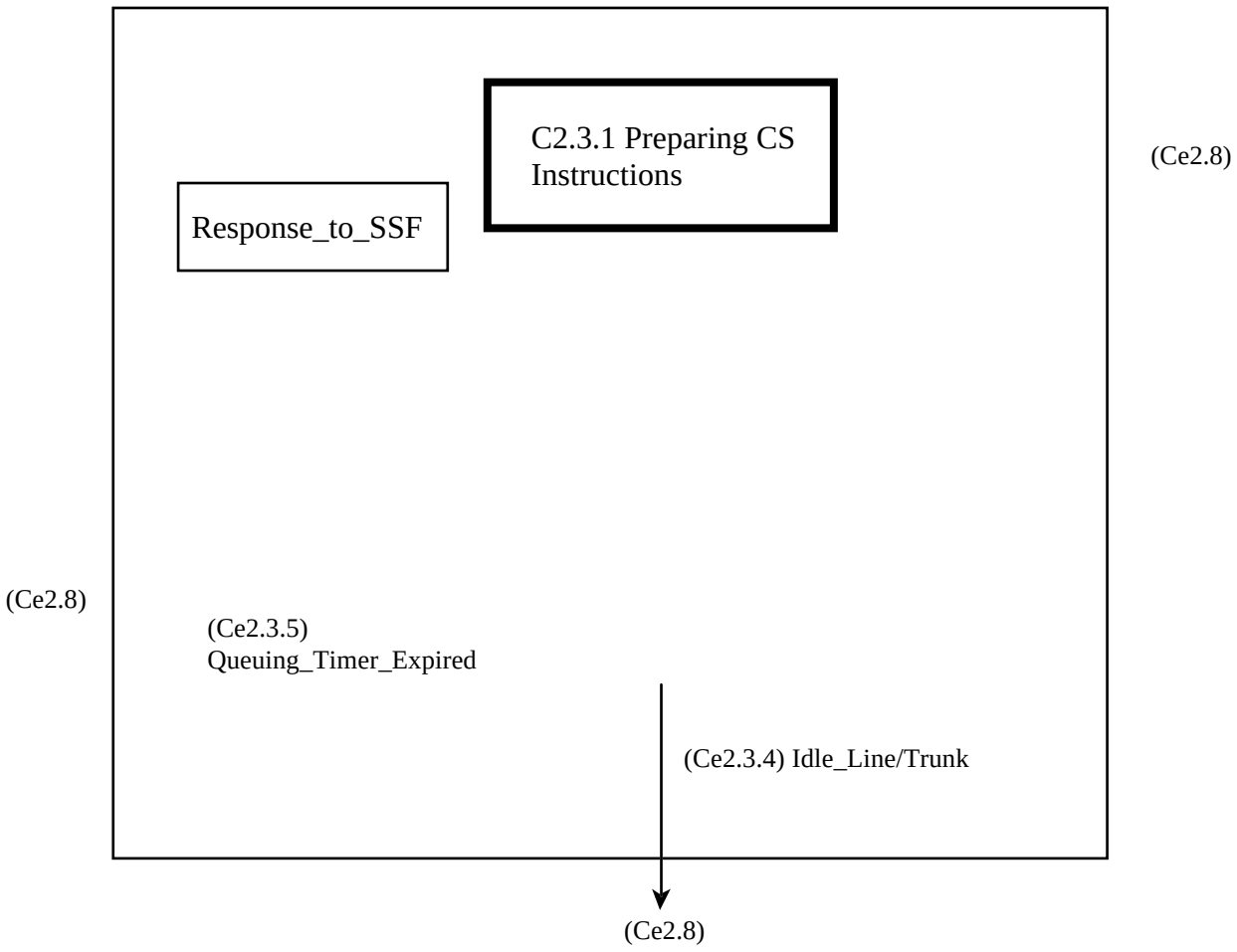


Figure 12-15: Partial expansion of state C2.3 FSM for CS

- When the application timer $T_{SCF-SSF}$ expires. In this case, the FSM for CS sends a **ResetTimer** operation to the SSF for the corresponding CS.

This event causes a transition back to State C2.3.1, **Preparing SSF Instructions**.

(Ce2.3.3) Busy_Line/Trunk
This is an internal event caused by the RCO when no terminating line/trunk is available. This event causes a transition to State C2.3.2, **Queuing**.

13.5.1.3.2.3.2 State C2.3.2: "Queuing"

In this state, the FSM for CS is awaiting an indication from the RCO to proceed with routeing a call to an idle trunk/line. The support of playing various announcements is also provided when the FSM for CS is in this state. In this ITU-T Recommendation, the relevant further expansion of the state is not provided; however, it is not different from that of the FSM for CS States C3. Nevertheless, if announcements are completed before the call is dequeued and the SSF FSM has transitioned to state **Waiting for Instructions**, the operation should be sent to set the T_{SSF} with an appropriate value. Once the FSM for CS enters this state, the Queuing Timer is started. The role of this timer is as follows:

- the Queuing Timer limits the time that a call can spend in the queue, and its value may be customer-specific.

The following events are considered in this state:

(Ce2.3.4) Idle_Line/Trunk
This is an internal event, which maps into the event (Ce2.8).

(Ce2.3.5) Queuing_Timer_Expired
This is an internal event, which results in processing the Cancel method of the RCO and maps into the event (Ce2.8) (following procedures depends on the decision of the SL that may play (or not play) the terminating announcement).

13.5.1.3.3 State C3: "Suspended and User Interaction"

The following events are considered in this state:

(Ce7) Continue_SCF_Processing
In this case, the SCF has obtained all the information from the SRF that is needed to instruct the SSF to complete the call. This event causes a transition to state C2, **Preparing CS instructions**.

(Ce8) Processing_Completed
This is an internal event, caused by the end of processing for the CS. This event causes a transition to the state C1, **CS Control Idle**.

(Ce11) CS_Instruction_Needed_During_Established_Temporary_Connection
This is an internal event, caused by an EDP-R while being in state C4.3 Establishing Temporary Connection Monitoring. This event causes a transition to the state C3.3, **Establishing Temporary Connection**.

(Ce12) CS_Instruction_Needed_During_User_Interaction
This is an internal event, caused by an EDP-R while being in the state C4.2 User Interaction Monitoring. This event causes a transition to the state C3.2, **User Interaction**.

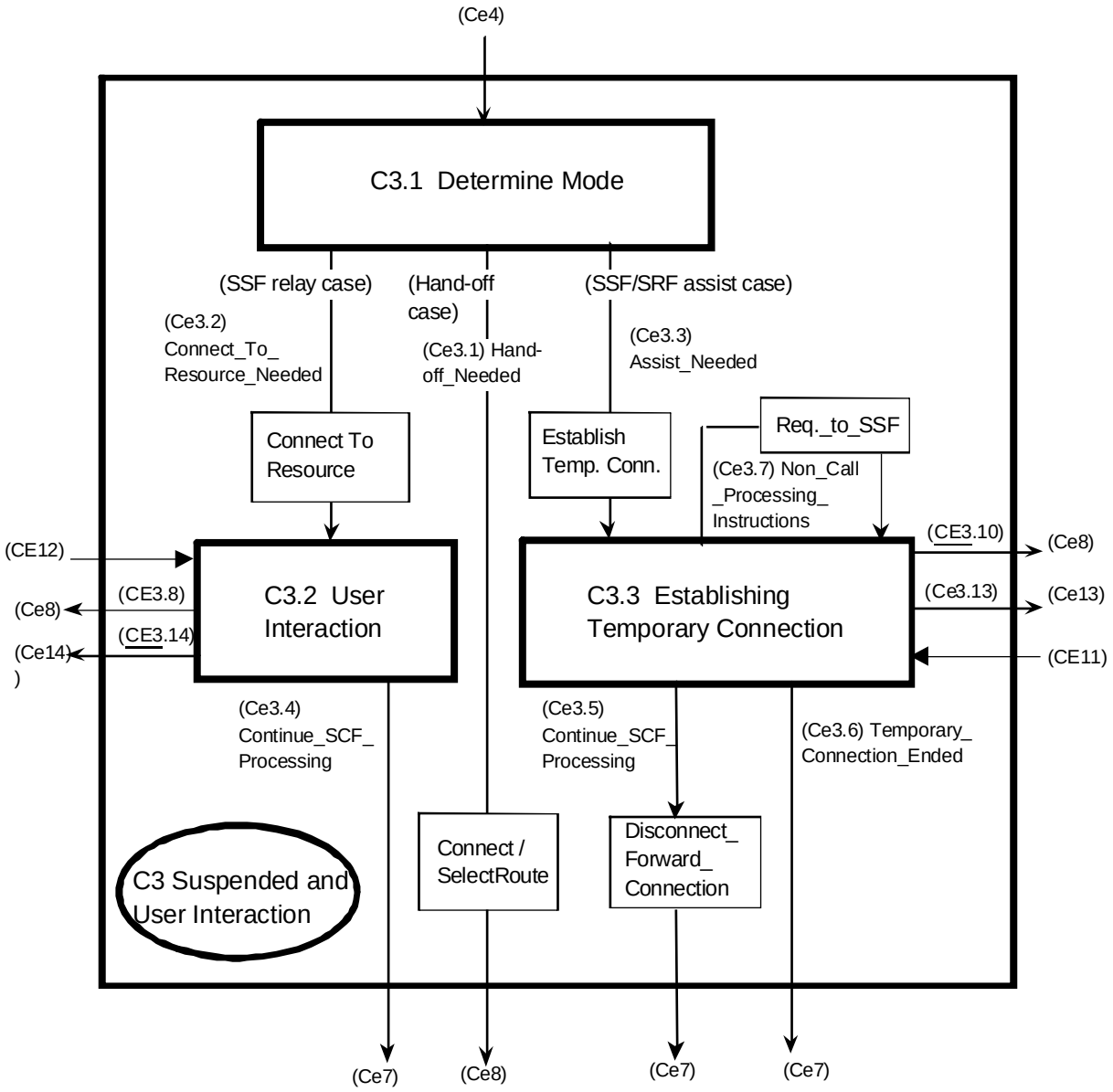
(Ce13) CS_Monitoring_Needed_During_Established_Temporary_Connection
This is an internal event, caused by the necessity of monitoring, while being in state C3.3 Establishing Temporary Connection. This event causes a transition to the state C4.3, **Establishing Temporary Connection Monitoring**.

(Ce14) CS_Monitoring_Needed_During_User_Interaction
This is an internal event, caused by the necessity of monitoring, while being in the state C3.2 User Interaction. This event causes a transition to the state C4.2, **User Interaction Monitoring**.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in figure 12-16.

13.5.1.3.3.1 State C3.1: "Determine Mode"

The following events are considered in this state:



(CE3.8), (CE3.10): Last_CS_Event_Received
(CE3.9), (CE3.11): Not_Last_CS_Event_Received
(CE3.12): CS_Control_Requested

Figure 12-16: Partial expansion of state C3 FSM for CS(Ce3.1) Hand-off_Needed
This is an internal event that takes place only with the hand-off case. In this case, the SCF sends the Connect **operation with the handed-off SSF address to the initiating SSF. This event is mapped as the FSM event (Ce8).**

(Ce3.2) Connect_To_Resource_Needed
This is an internal event that takes place only in the case of initiating SSF relay. In this case, the SCF sends the **ConnectToResource** operation to the initiating SSF. This event causes a transition to the state C3.2, **User Interaction**. In this case, the FSM for CS instance notifies the associate FSM for CSA instance of this event (User_Interaction_Requested).

(Ce3.3) Assist_Needed
This is an internal event that takes place when either the assisting SSF or the direct SCF-SRF relation is needed. In

this case, the SCF sends the **EstablishTemporaryConnection** operation to the initiating SSF with the assisting SSF address or the assisting SRF address. This event causes a transition to the state C3.3, **Establishing Temporary Connection**.

13.5.1.3.3.2 State C3.2: "User Interaction"

The following events are considered in this state:

(Ce3.4) Continue_SCF_Processing
In this case, the SCF has obtained all the information from the SRF that is needed to instruct the SSF to proceed the call. This event is mapped as the FSM event (Ce7).

(CE3.8) Last_CS_Event_Received
This is an external event caused by a reception of a last event from the corresponding CS. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (Ce8).

(Ce3.14) CS_Monitoring_Needed_During_User_Interaction
This is an internal event, caused by the SL because of the need for monitoring during User Interaction. This event causes a transition to the state C4.2, **User Interaction Monitoring**. This event is mapped as the FSM event (Ce14).

To describe further the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in figure 12-17

13.5.1.3.3.2.1 State C3.2.1: "User Interaction"

The following events are considered in this state:

(Ce3.2.1) Request_To_SRF
This event is an internal event caused by sending one or more of the following operations to the SSF.

PlayAnnouncement
PromptAndCollectUserInformation
ScriptRun
ScriptInfo
ScriptClose
PromptAndReceiveMessage

This event causes a transition back to the state C3.2.1, **User Interaction**.

(Ce3.2.2) Final_Request_To_SRF (with Final Response Requested)
This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-initiated disconnect. In this case, the SCF sends the **PlayAnnouncement** (containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) or **PromptAndCollectUserInformation** operation or ScriptRun, ScriptInformation, ScriptClose, or PromptAndReceiveMessage operation with permission of SRF-initiated disconnect to the SRF. In this case, the FSM for CS transits back to the same state.

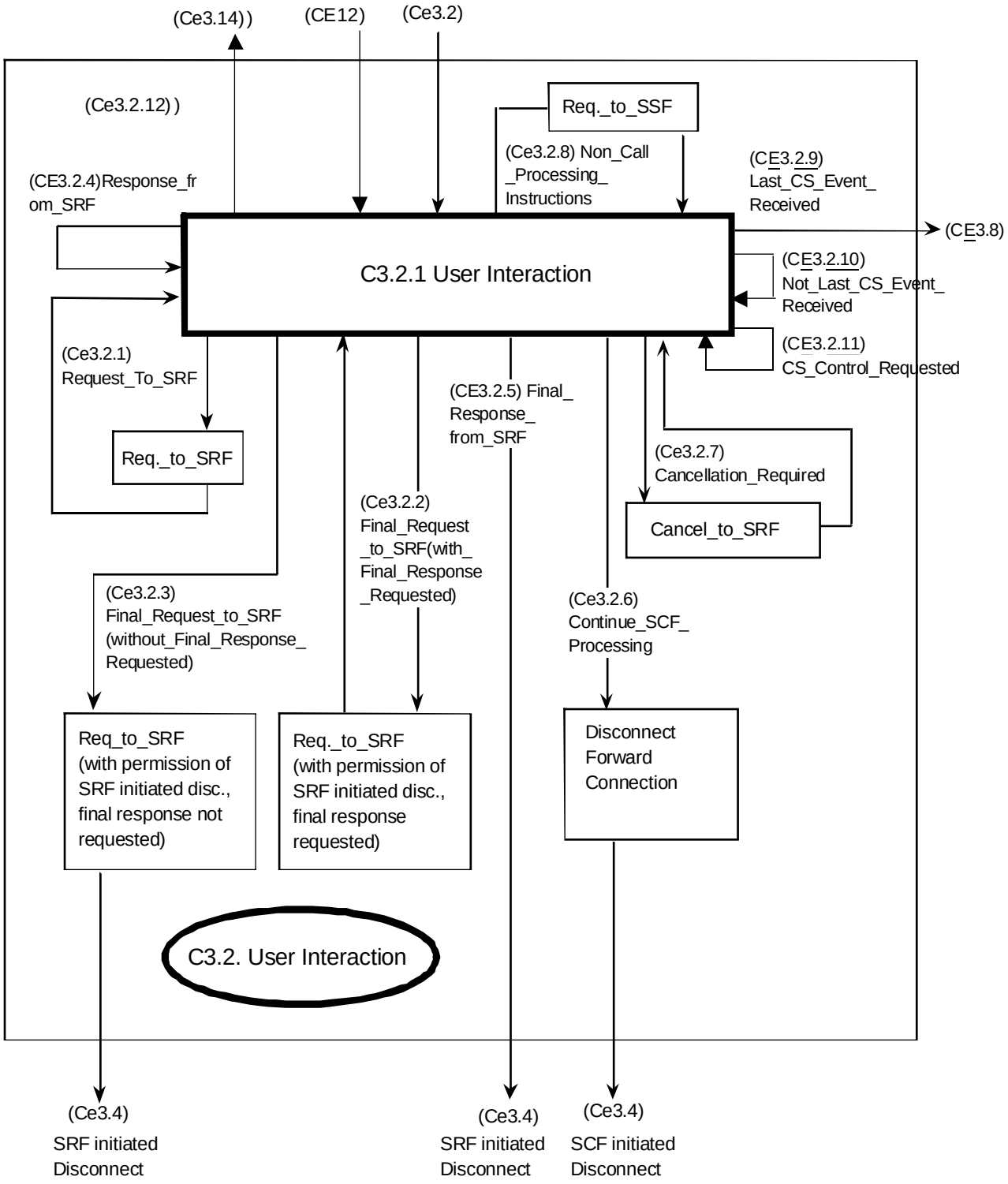


Figure 12-17: Partial expansion of state C3.2 FSM for CS

(Ce3.2.3) Final_Request_To_SRF (without Final Response Requested)

This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-initiated disconnect, while no **SpecializedResourceReport** operation has been requested to be returned to the SCF when an announcement is completed. In this case, the SCF sends the **PlayAnnouncement** (not containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) or **ScriptRun**, **ScriptInformation** or **ScriptClose** operation with permission of SRF-initiated disconnect to the SRF. This event is mapped as the FSM event (Ce3.4).

(Ce3.2.4) Response_from_SRF

This is an external event caused by the reception of **SpecializedResourceReport, or ScriptEvent,** or return result from **PromptAndReceiveMessage,** or return result from **PromptAndCollectUserInformation** operation. On the receipt of either, the FSM for CS transits back to the same state.

(CE3.2.5) Final_Response_from_SRF

This is an external event caused by the reception of **SpecializedResourceReport, or ScriptEvent** or return result from **PromptAndReceiveMessage,** or return result from **PromptAndCollectUserInformation** operation with permission of SRF-initiated disconnect. This event is mapped as the FSM event (Ce3.4).

(Ce3.2.6) Continue_SCF_Processing

This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and SRF by means of SCF initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** or DisconnectForwardConnectionWithArgument operation to the initiating SSF. This event is mapped as the FSM event (Ce3.4).

(Ce3.2.7) Cancellation_Required

This is an internal event that takes place when the SLPI cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCF sends the **Cancel** operation to the SSF. The FSM for CS transits back to the same state.

(Ce3.2.8) Non-Call_Processing_Instructions

This is an internal event caused by the SL when there is a need to send one or more of the following operations to the SSF:

ApplyCharging
FurnishChargingInformation
RequestNotificationChargingEvent
SendChargingInformation

The FSM for CS transits back to the same state.

(CE3.2.9) Last_CS_Event_Received

This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (for Abandon/Disconnect EDP-N)
EntityReleased

In this case, there is neither an outstanding armed EDP, a pending **CallInformationReport,** nor a pending **ApplyChargingReport** operation. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (CE3.8).

(CE3.2.10) Not_Last_CS_Event_Received

This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (for Abandon/Disconnect EDP-N)

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. Th FSM for CS transits back to the same state.

(CE3.2.11) CS_Control_Requested

This is an external event caused by a reception of the following operation from the SSF:

EventReportBCSM (for Abandon/Disconnect EDP-R)

The FSM for CS transits back to the same state.

(Ce3.2.12) Monitoring_Needed_During_User_Interaction

This is an internal event caused by the SL when there is a need to bring the SSF in the processing state, by sending one of the following operations to the SSF:

Continue; (only single call segment CSA)

ContinueWithArgument

This event causes a transition to state C4.2 User Interaction Monitoring. This event is mapped as the FSM event (Ce3.14).

13.5.1.3.3.3 State C3.3: "Establishing Temporary Connection"

The following events are considered in this state:

(Ce3.5) Continue_SCF_Processing

This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the assisting SSF/SRF by means of SCF initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** operation to the initiating SSF. This event is mapped as the FSM event (Ce7).

(Ce3.6) Temporary_Connection_Ended

This is an internal event caused by the notification from the associate FSM for CSA instance because of the end of the user interaction for the assisting SRF. This event is mapped as the FSM event (Ce7).

(Ce3.7) Non-Call_Processing_Instructions

This is an internal event caused by the SL when there is a need to send one or more of the following operations to the SSF:

ApplyCharging
FurnishChargingInformation
RequestNotificationChargingEvent
SendChargingInformation

The FSM for CS transits back to the same state.

(CE3.10) Last_CS_Event_Received

This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (for Abandon/Disconnect EDP-N)
EntityReleased

In this case, there is no outstanding armed EDP, pending **CallInformationReport,** or pending **ApplyChargingReport** operation. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (Ce8).

(CE3.11) Not_Last_CS_Event_Received

This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (for Abandon/Disconnect EDP-N)

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. The FSM for CS transits back to the same state.

(CE3.12) CS_Control_Requested

This is an external event caused by a reception of the following operation from the SSF:

EventReportBCSM (for Abandon/Disconnect EDP-R)

The FSM for CS transits back to the same state.

(CE3.13) CS_Monitoring_Needed_During_Established_Temporary_Connection

This is an internal event, caused by the SL when there is a need for monitoring, while being in state C3.3 Establishing Temporary Connection. In this case one of the following operations is sent to the SSF:

Continue
ContinueWithArgument

This event causes a transition to the state C4.3, **Establishing Temporary Connection Monitoring**. This event is mapped as the FSM event (Ce13).

13.5.1.3.4 State C4: "Not Suspended and User Interaction"

The following events are considered in this state:

(Ce8) Processing_Completed
This is an internal event, caused by the end of processing for the CS. This event causes a transition to the state C1, **CS Control Idle**.

(Ce9) CS_Monitoring_Needed
This is an internal event, caused by the necessity of monitoring. This event causes a transition to the state C2.2, **Waiting for Notification or Request**.

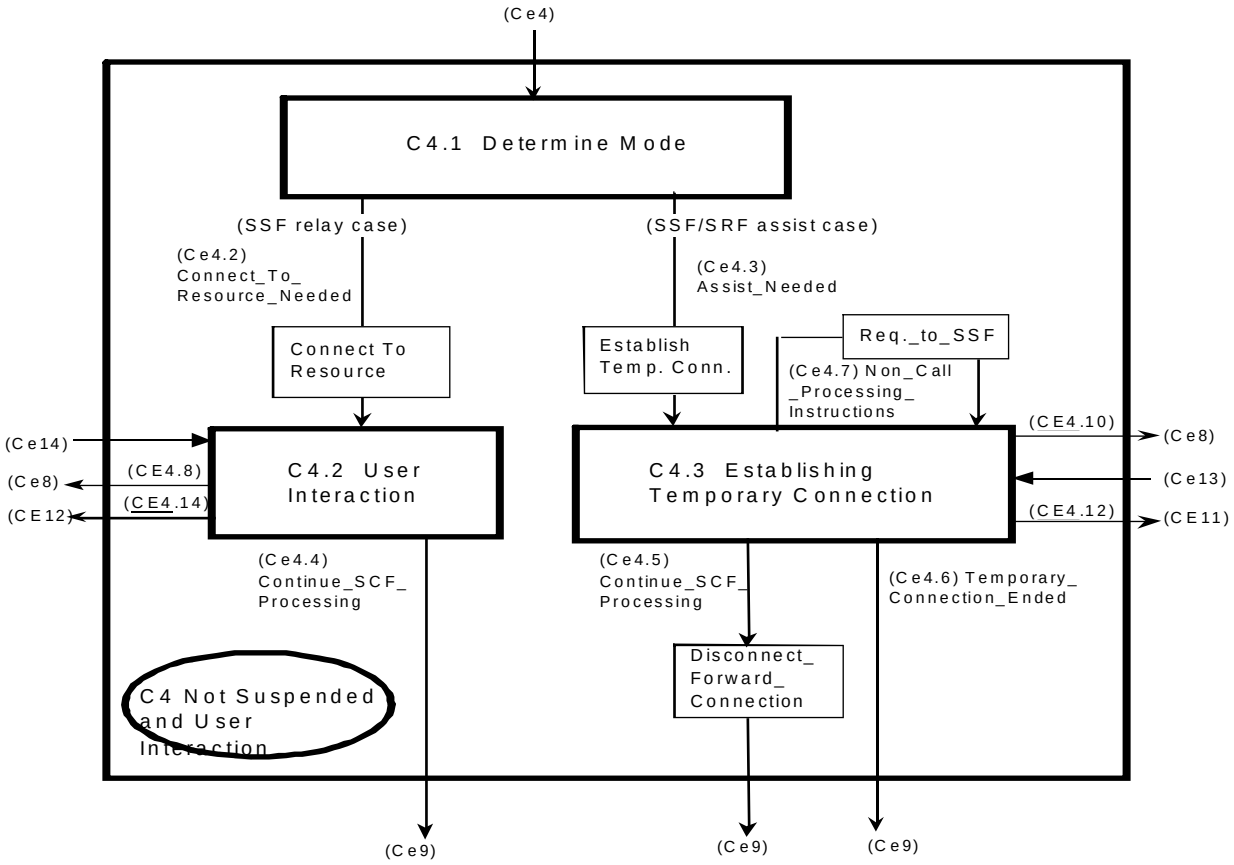
(Ce11) CS_Instruction_Needed_During_Established_Temporary_Connection
This is an internal event, caused by an EDP-R while being in state C4.3 Establishing Temporary Connection Monitoring. This event causes a transition to the state C3.3, **Establishing Temporary Connection**.

(Ce12) CS_Instruction_Needed_During_User_Interaction
This is an internal event, caused by an EDP-R while being in the state C4.2 User Interaction Monitoring. This event causes a transition to the state C3.2, **User Interaction**.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in figure 12-18.

13.5.1.3.4.1 State C4.1: "Determine Mode Monitoring"

The following events are considered in this state:



[editor note: change C3 to C4, i.e.: " figure 12-15: Partial Expansion of the State C4 FSM for CS" and change (Ce4]) into (Ce10) in the figure for the arrow to C4.1 Determine Mode]

Figure 12-18: Partial expansion of state C3 FSM for CS

(Ce4.2) Connect_To_Resource_Needed
This is an internal event that takes place only in the case of initiating SSF relay. In this case, the SCF sends the **ConnectToResource** operation to the initiating SSF. This event causes a transition to the state C4.2, **User Interaction**. In this case, the FSM for CS instance notifies the associate FSM for CSA instance of this event (User_Interaction_Requested).

(Ce4.3) Assist_Needed
This is an internal event that takes place when either the assisting SSF or the direct SCF-SRF relation is needed. In this case, the SCF sends the **EstablishTemporaryConnection** operation to the initiating SSF with the assisting SSF address or the assisting SRF address. This event causes a transition to the state C3.3, **Establishing Temporary Connection**.

13.5.1.3.4.2 State C4.2: "User Interaction"

The following events are considered in this state:

(CE4.4) User_Interaction_Finished
In this case, the SCF has exchanged all required information with the user to go back to the pure monitoring mode. This event causes a transition to the state C2.2, **Waiting for Notification or Request**. This event is mapped as the FSM event (Ce9).

(CE4.8) Last_CS_Event_Received
This is an external event caused by a reception of a last event from the corresponding CS. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (Ce8).

(CE4.14) CS_Instruction_Needed_During_User_Interaction
This is an external event caused by a reception of an EDP-R. This event causes a transition to the state C3.2, **User Interaction**. This event is mapped as the FSM event (CE12).

To describe further the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses. This subdivision is illustrated in figure 12-18.

13.5.1.3.4.2.1 State 4.2.1: "User Interaction"

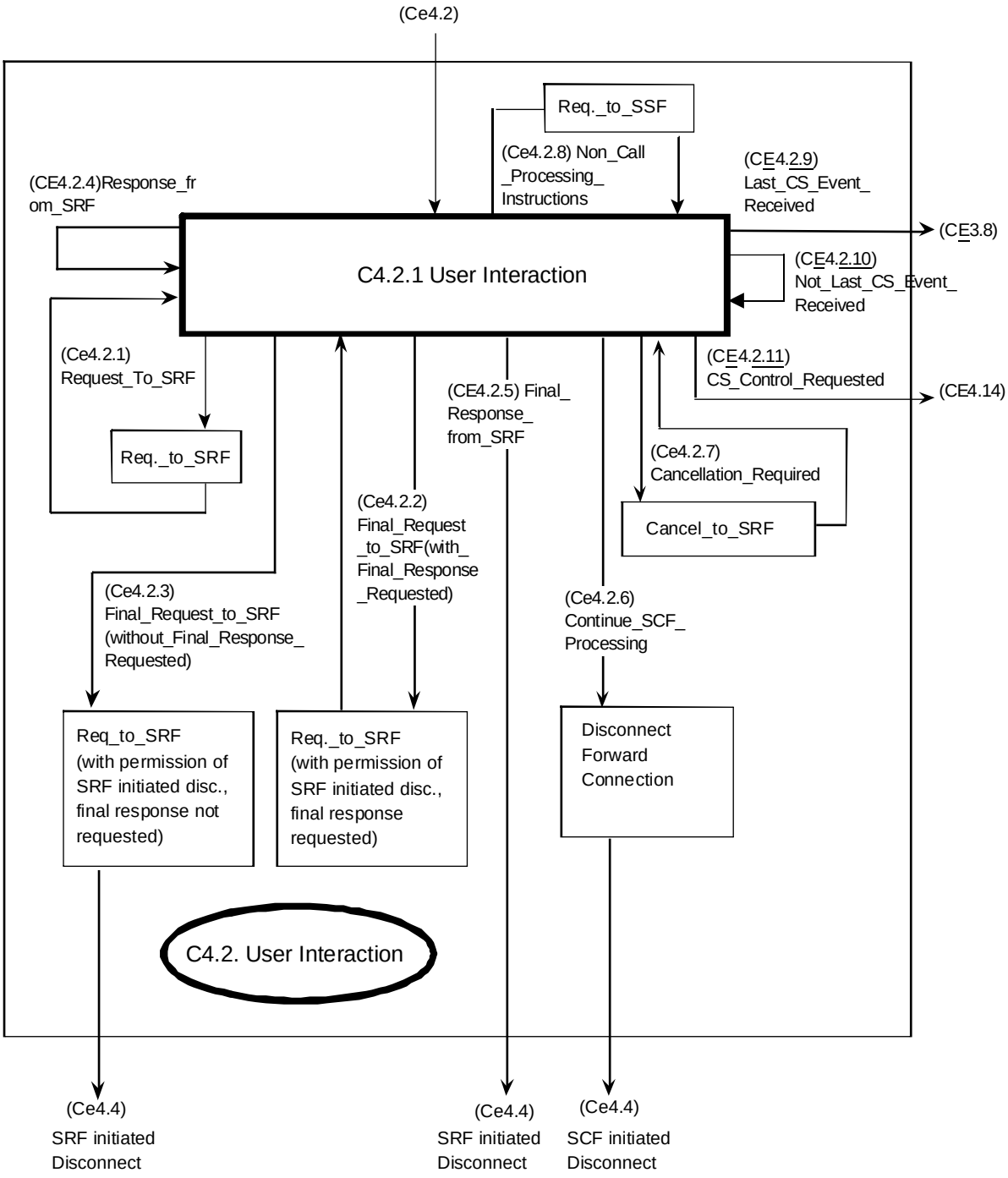
The following events are considered in this state:

(Ce4.2.1) Request_To_SRF
This event is an internal event caused by sending one or more of the following operations to the SSF.

PlayAnnouncement

This event causes a transition back to the state C4.2.1, **User Interaction Monitoring**.

(Ce4.2.2) Final_Request_To_SRF (with Final Response Requested)
This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-initiated disconnect. In this case, the SCF sends the **PlayAnnouncement** (containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) with permission of SRF-initiated disconnect to the SRF. In this case, the FSM for CS transits back to the same state.



[Editor note: SubsequenteFigure numbering in the following figures wrong, should be in sequentiel order, i.e. next figure 12-17 etc...].

Figure 12-19: Partial expansion of state C4.2 FSM for CS

(Ce4.2.3) Final_Request_To_SRF (without Final Response Requested)

This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF-initiated disconnect, while no **SpecializedResourceReport** operation has been requested to be returned to the SCF when an announcement is completed. In this case, the SCF sends the **PlayAnnouncement** (not containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) with permission of SRF-initiated disconnect to the SRF. This event is mapped as the FSM event (Ce4.4).

(Ce4.2.4) Response_from_SRF
This is an external event caused by the reception of **SpecializedResourceReport**. On the receipt, the FSM for CS transits back to the same state.

(Ce4.2.5) Final_Response_from_SRF
This is an external event caused by the reception of **SpecializedResourceReport** with permission of SRF-initiated disconnect. This event is mapped as the FSM event (Ce4.4).

(Ce4.2.6) User_Interaction_Finished
This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and SRF by means of SCF initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** operation to the initiating SSF. This event is mapped as the FSM event (Ce4.4).

(Ce4.2.7) Cancellation_Required
This is an internal event that takes place when the SLPI cancels a previous **PlayAnnouncement** operation. In this case, the SCF sends the **Cancel** operation to the SSF. The FSM for CS transits back to the same state.

(Ce4.2.8) Non-Call_Processing_Instructions
This is an internal event caused by the SL when there is a need to send one or more of the following operations to the SSF:

ApplyCharging
RequestNotificationChargingEvent
SendChargingInformation

The FSM for CS transits back to the same state.

(CE4.2.9) Last_CS_Event_Received: This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (for Abandon/Disconnect EDP-N)
EntityReleased

In this case, there is neither an outstanding armed EDP, a pending **CallInformationReport,** nor a pending **ApplyChargingReport** operation. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (CE4.8).

(CE4.2.10) Not_Last_CS_Event_Received
This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (EDP-N)

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. The FSM for CS transits back to the same state.

(CE4.2.11) CS_Instruction_Needed_During_User_Interaction
Event caused by a reception of the following operation from the SSF:

EventReportBCSM (EDP-R)

This event causes a transition to the state C4.2, **User Interaction**. This event is mapped as the FSM event (Ce4.14).

13.5.1.3.4.3 State C4.3: "Establishing Temporary Connection"

The following events are considered in this state:

(Ce4.5) Continue_Processing
This is an internal event that takes place when the FSM for CS instance finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the assisting SSF/SRF by means of SCF initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** operation to the initiating SSF. This event is mapped as the FSM event (Ce9).

(Ce4.6) Temporary_Connection_Ended
This is an internal event caused by the notification from the associate FSM for CSA instance because of the end of the user interaction for the assisting SRF. This event is mapped as the FSM event (Ce9).

(Ce4.7) Non-Call_Processing_Instructions
This is an internal event caused by the SL when there is a need to send one or more of the following operations to the SSF:

ApplyCharging
RequestNotificationChargingEvent
SendChargingInformation

The FSM for CS transits back to the same state.

(CE4.10) Last_CS_Event_Received
This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (EDP-N)
EntityReleased

In this case, there is no outstanding armed EDP, pending **CallInformationReport,** or pending **ApplyChargingReport** operation. This event causes a transition to the state C1, **CS Control Idle**. This event is mapped as the FSM event (Ce8).

(CE4.11) Not_Last_CS_Event_Received
This is an external event caused by a reception of one of the following operations from the SSF:

CallInformationReport
ApplyChargingReport
EventReportBCSM (EDP-N)

In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. The FSM for CS transits back to the same state.

(CE4.12) CS_Instruction_Needed_During_Established_Temporary_Connection
This is an external event caused by a reception of the following operation from the SSF:

EventReportBCSM (EDP-R)

This event causes a transition to the state C3.3, **Establishing Temporary Connection**. This event is mapped as the FSM event (Ce11).

13.5.1.4 FSM for Specialized Resource

Figure 12-20 shows the general State Diagram of the FSM for Specialized Resource as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

13.5.1.4.1 State R1: "SRF Control Idle"

The following events are considered in this state:

(RE1) Assist_Request_Instructions_from_SRF
This is an external event caused by a reception of an **AssistRequestInstructions** operations from the SRF. This event causes a transition to the state R2, **Controlling SRF**.

13.5.1.4.2 State R2: "Controlling SRF"

The following events are considered in this state:

(Re2) Request_to_SRF

This is an internal event caused by the SLPI when there is a need to send one or more of the following operations to the SSF:

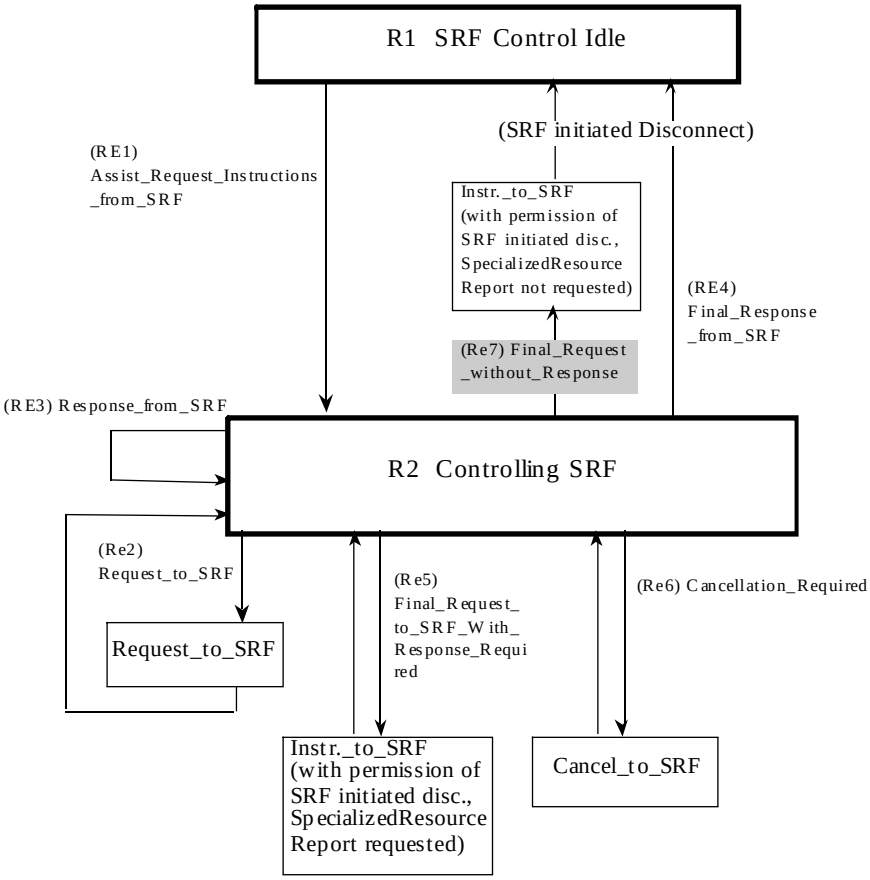


Figure 12-20: FSM for Specialized Resource

PlayAnnouncement
PromptAndCollectUserInformation
ScriptRun
ScriptInfo
ScriptClose
PromptAndReceiveMessage

This event causes a transition back to the same state.

(RE3) Response_from_SRF

This is an external event caused by a reception of one of the following:

SpecializedResourceReport
Return Result for PromptAndCollectUserInformation
Return Result for PromptAndReceiveMessage
ScriptEvent

This event causes a transition back to the same state.

(RE4) Final_Response_from_SRF
This is an external event caused by a reception of one of the following after SRF-initiated disconnect:

SpecializedResourceReport
Return Result for PromptAndCollectUserInformation.
Return Result for PromptAndReceiveMessage
ScriptEvent

This event causes a transition to the state R1, **SRF Control Idle**.

(Re5) Final_Request_to_SRF_with_Respose_Required
This is an internal event caused by the SLPI when there is a need to send one of the following operations with permission of SRF-initiated disconnect to the SSF:

ScriptRun
PromptAndReceiveMessage
PlayAnnouncement
PromptAndCollectUserInformation

This event causes a transition back to the same state.

(Re6) Cancellation_Required
This is an internal event that takes place when the SLPI cancels the previous **PA** or **PromptAndCollectUserInformation** operation. In this case, the SCSM sends the Cancel operation to the SRF, and transits back to the same state.

(Re7) Final_Request_without_Response
This is an internal event that takes place when the SCSM finishes the user interaction and requests the disconnection of bearer connection between the initiating SSF and the SRF by means of SRF Initiated disconnect, while no **SpecializedResourceReport** operation is requested to be returned to the SCF when an announcement is completed. In this case, the SCF sends the **PlayAnnouncement** (not containing a request for returning a **SpecializedResourceReport** operation as an indication of completion of the operation) with permission of SRF-initiated disconnect to the SRF. ScriptRun, ScriptClose and ScriptInformation operations are also valid at this event. This event causes a transition to the state R1, **SRF Control Idle**.

13.5.1.5 FSM for Assisting SSF

Figure 12-21 shows the general State Diagram of the FSM for assisting SSF as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

The FSM for assisting SSF has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to restart the timer, $T_{SSF,}$ to guard the association between the assisting SSF and the SCF.

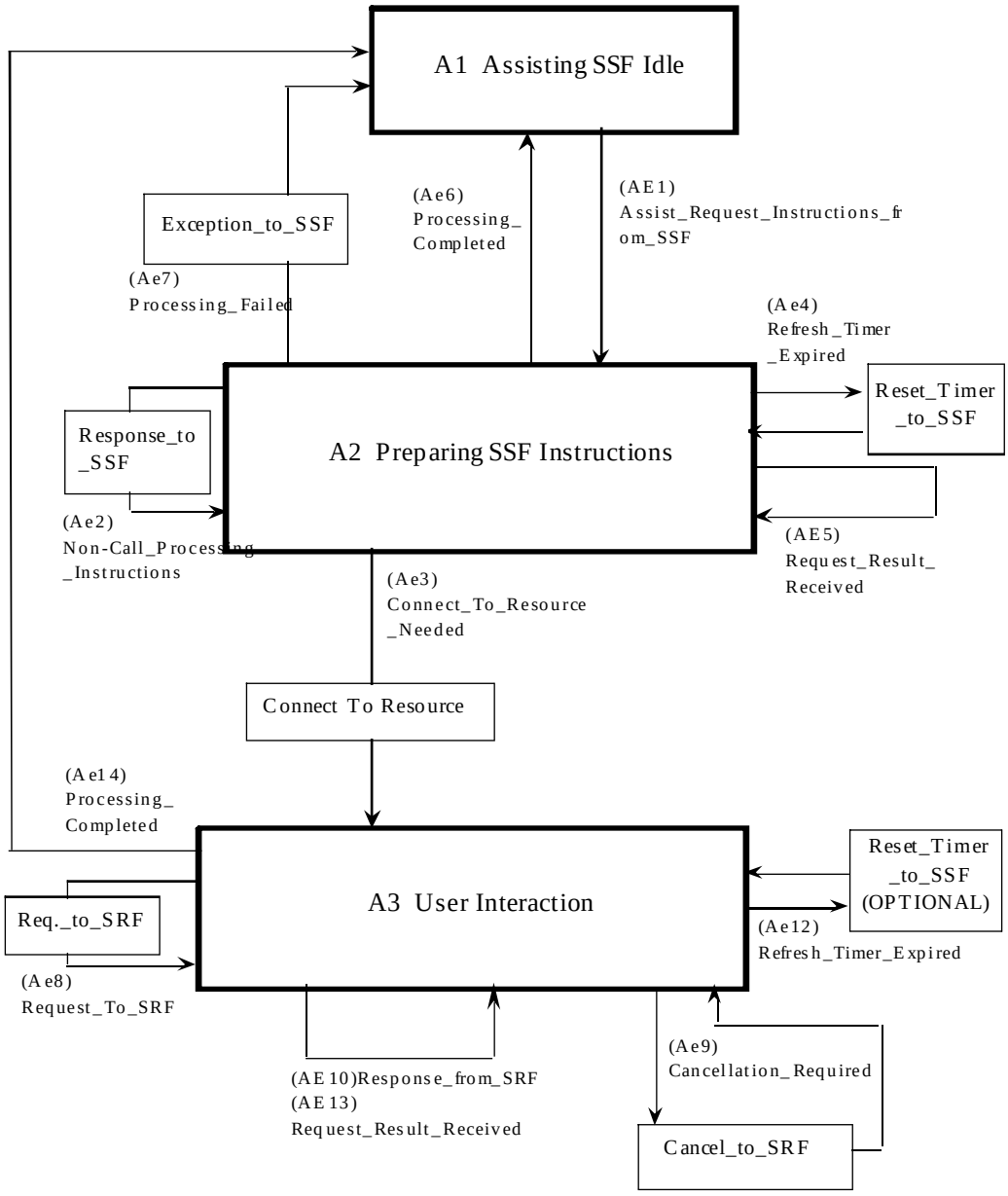


Figure 12-21: FSM for Assisting SSF

Timer $T_{SCF\text{-}SSF}$ is set in the following cases:

- when the SCF receives an **AssistRequestInstructions** operation. In this case, this timer is restarted when a first request, other than **ResetTimer** operation, is sent to the assisting SSF. On the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for assisting SSF may restart T_{SSF} once using the **ResetTimer** operation, and restart timer $T_{SCF\text{-}SSF}$. On the second expiration of $T_{SCF\text{-}SSF}$, the FSM for assisting SSF informs the SLPI and the maintenance functions, and the FSM for assisting SSF transits to the state A1, Assisting SSF Idle;

- when FSM for assisting SSF enters the "User Interaction" state. In this case, on the expiration of $T_{SCF\text{-}SSF}$, the SCF may restart T_{SSF} using the **ResetTimer** operation any number of times (OPTIONAL).

NOTE: The word "OPTIONAL" refers to the use of the application timer $T_{SCF\text{-}SSF}$. Whether it is used depends on an implementation, but, if used, it must be synchronized with T_{SSF} in the assisting SSF FSM.

In both cases, $T_{SCF\text{-}SSF}$ may respectively have different values as defined by the application. The values of $T_{SCF\text{-}SSF}$ are smaller than the respective values of T_{SSF}.

When receiving or sending any other operation, the SCF should restart $T_{SCF\text{-}SSF}$.

13.5.1.5.1 State A1: "Assisting SSF Idle"

The following events are considered in this state:

(AE1) Assist_Request_Instructions_from_SSF
This is an external event caused by reception of the following operation:

AssistRequestInstructions
This event causes a transition to the state A2, **Preparing SSF Instructions**.

13.5.1.5.2 State A2: " Preparing SSF Instructions"

The following events are considered in this state:

(Ae2) Non-Call_Processing_Instructions
This is an internal event caused by the SLPI when there is a need to send one or more of the following operations to the SSF:

ApplyCharging
FurnishChargingInformation
SendChargingInformation
This event causes a transition back to the state A2, **Preparing SSF Instructions**.

(Ae3) Connect_To_Resource_Needed
This is an internal event. In this case, the SCF sends the **ConnectToResource** operation to the assisting SSF. This event causes a transition to the state A3, **User Interaction**.

(Ae4) Refresh_Timer_Expired
This is an internal event, which results in sending the **ResetTimer** operation to the assisting SSF and a transition back to the same state.

(AE5) Request_Result_Received
This is an external event caused by a reception of the following operation from the assisting SSF:

ApplyChargingReport
This event causes a transition to the same state.

(Ae6) Processing_Completed
This is an internal event, caused by the end of processing for the assisting SSF. This event causes a transition to the state A1, **Assisting SSF Idle**.

(Ae7) Processing_Failed: This (internal) event causes an appropriate exception processing and a transition back to the state A1, **Assisting SSF Idle**.

13.5.1.5.3 State A3: " User Interaction"

The following events are considered in this state:

(Ae8) Request_To_SRF
This event is a internal event caused by sending one or more of the following operations to the assisting SSF.

PlayAnnouncement
PromptAndCollectUserInformation
ScriptRun
ScriptInfo
ScriptClose
PromptAndReceiveMessage
This event causes a transition back to the state A3, **User Interaction**.

(Ae9) Cancellation_Required

This is an internal event that takes place when the SLPI cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCF sends the **Cancel** operation to the assisting SSF. The FSM for assisting SSF transits back to the same state.

(AE10) Response_from_SRF

This is an external event caused by the reception of **SpecializedResourceReport** or **Return Result for PromptAndCollectUserInformation** operation or Return Result for **PromptAndReceiveMessage** or **ScriptEvent**. On the receipt of any, the FSM for assisting SSF transits back to the same state.

(Ae12) Refresh_Timer_Expired

This is an internal event, which results in sending the **ResetTimer** operation to the assisting SSF and a transition back to the same state.

(AE13) Request_Result_Received

This is an external event caused by a reception of the following operation from the assisting SSF:

ApplyChargingReport

This event causes a transition to the same state.

(Ae14) Processing_Completed

This is an internal event, caused by the end of processing for the assisting SSF. This event causes a transition to the state A1, **Assisting SSF Idle**.

13.5.1.6 FSM for Handed-off SSF

Figure 12-22 shows the general State Diagram of the FSM for handed-off SSF as relevant to the procedures concerning the SCF FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses. The Hand-Off FSM for IN CS2 applies only to the case where final treatment is to be applied

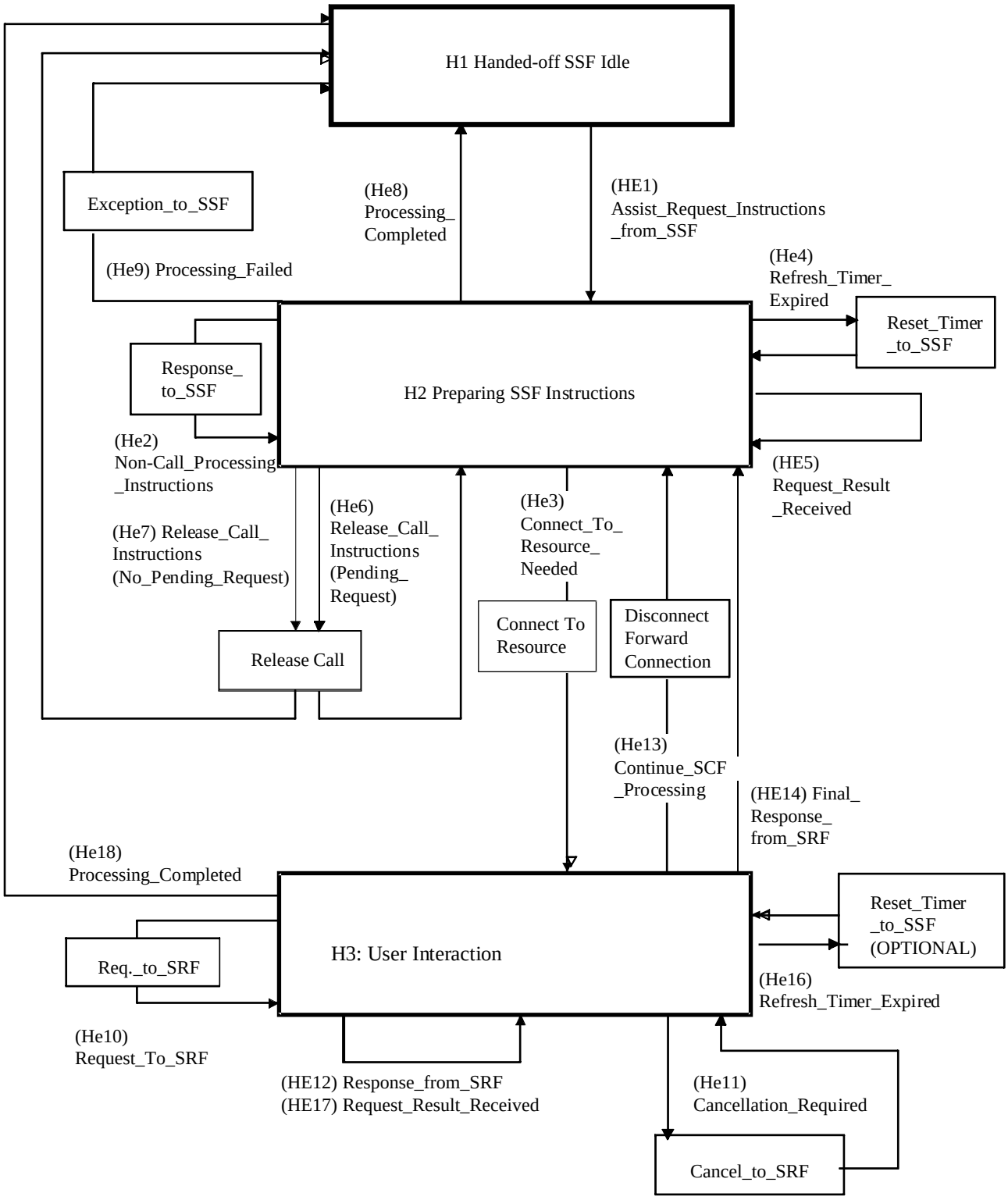


Figure 12-22: FSM for Handed-off SSF

The FSM for handed-off SSF has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to restart the timer, T_{SSF}, to guard the association between the handed-off SSF and the SCF.

Timer $T_{SCF\text{-}SSF}$ is set in the following cases:

- when the SCF receives an **AssistRequestInstructions** operation. In this case, this timer is restarted when a first request, other than **ResetTimer** operation, is sent to the handed-off SSF. On the expiration of timer $T_{SCF\text{-}SSF}$, the FSM for handed-off SSF may restart T_{SSF} once using the **ResetTimer** operation, and restart timer $T_{SCF\text{-}SSF}$. On the second expiration of $T_{SCF\text{-}SSF}$, the FSM for handed-off SSF informs the SLPI and the maintenance functions, and the FSM for handed-off SSF transits to the state H1, **Handed-off SSF Idle**;

- when FSM for handed-off SSF enters the "**User Interaction**" state. In this case, on the expiration of $T_{SCF\text{-}SSF}$, the SCF may restart T_{SSF} using the **ResetTimer** operation any number of times (OPTIONAL).

NOTE: The word "OPTIONAL" refers to the use of the application timer $T_{SCF\text{-}SSF}$. Whether it is used depends on an implementation, but, if used, it must be synchronized with T_{SSF} in the handed-off SSF FSM.

In both cases, $T_{SCF\text{-}SSF}$ may respectively have different values as defined by the application. The values of $T_{SCF\text{-}SSF}$ are smaller than the respective values of T_{SSF}.

When receiving or sending any other operation, the SCF should restart $T_{SCF\text{-}SSF}$.

13.5.1.6.1 State H1: "Handed-off SSF Idle"

The following events are considered in this state:

(HE1) Assist_Request_Instructions_from_SSF
This is an external event caused by reception of the following operation:

AssistRequestInstructions

This event causes a transition to the state H2, **Preparing SSF Instructions**.

13.5.1.6.2 State H2: " Preparing SSF Instructions"

The following events are considered in this state:

(He2) Non-Call_Processing_Instructions
This is an internal event caused by the SLPI when there is a need to send one or more of the following operations to the SSF:

ApplyCharging
FurnishChargingInformation
SendChargingInformation

This event causes a transition back to the state H2, **Preparing SSF Instructions**.

(He3) Connect_To_Resource_Needed
This is an internal event. In this case, the SCF sends the **ConnectToResource** operation to the handed-off SSF.
This event causes a transition to the state H3, **User Interaction**.

(He4) Refresh_Timer_Expired
This is an internal event, which results in sending the **ResetTimer** operation to the handed-off SSF and a transition back to the same state.

(HE5) Request_Result_Received
This is an external event caused by a reception of the following operation from the handed-off SSF:

ApplyChargingReport
This event causes a transition to the same state.

(He6) Release_Call_Instructions (Pending_Request)
This is an internal event caused by sending a ReleaseCall operation when there are one or more pending requests (CallInformationRequest or ApplyCharging). This event causes a transition to the same state.

(He7) Release_Call_Instructions (No_Pending_Request)
This is an internal event caused by sending a ReleaseCall operation when there are no pending requests (neither CallInformationRequest nor ApplyCharging). This event causes a transition to the state H1, Handed-off SSF Idle.

(He8) Processing_Completed
This is an internal event, caused by the end of processing for the handed-off SSF. This event causes a transition to the state H1, **Handed-off SSF Idle**.

(He9) Processing_Failed
This (internal) event causes an appropriate exception processing and a transition back to the state H1, **Handed-off SSF Idle**.

13.5.1.6.3 State H3: " User Interaction "

The following events are considered in this state:

(He10) Request_To_SRF
This event is a internal event caused by sending one or more of the following operations to the handed-off SSF.

PlayAnnouncement
PromptAndCollectUserInformation
ScriptRun
ScriptInfo
ScriptClose
PromptAndReceiveMessage
This event causes a transition back to the state H3, **User Interaction**.

(He11) Cancellation_Required
This is an internal event that takes place when the SLPI cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCF sends the **Cancel** operation to the handed-off SSF. The FSM for handed-off SSF transits back to the same state.

(HE12) Response_from_SRF
This is an external event caused by the reception of **SpecializedResourceReport** or **Return Result for PromptAndCollectUserInformation** operation or **Return Result for PromptAndReceiveMessage** or ScriptEvent. On the receipt of any, the FSM for handed-off SSF transits back to the same state.

(He13) Continue_SCF_Processing
This is an internal event that takes place when the FSM instance for handed-off SSF finishes the user interaction and requests the disconnection of bearer connection between the handed-off SSF and SRF by means of SCF initiated disconnect. In this case, the SCF sends the **DisconnectForwardConnection** or **DisconnectForwardConnectionwithArgument** operation to the handed-off SSF. The FSM for handed-off SSF transits to the state H2, **Preparing SSF Instructions**.

(HE14) Final_Response_from_SRF
This is an external event caused by the reception of **SpecializedResourceReport** or **Return Result for PromptAndCollectUserInformation** operation or Return Result **for PromptAndReceiveMessage** or **ScriptEvent** with the permission of SRF-initiated disconnect. The FSM for handed-off SSF transits to the state H2, **Preparing SSF Instructions**.

(He16) Refresh_Timer_Expired
This is an internal event, which results in sending the **ResetTimer** operation to the handed-off SSF and a transition back to the same state.

(HE17) Request_Result_Received
This is an external event caused by reception of the following operations from the handed-off SSF:

ApplyChargingReport
This event causes a transition to the same state.

(He18) Processing_Completed
This is an internal event, caused by the end of processing for the handed-off SSF. This event causes a transition to the state H1, **Handed-off SSF Idle**.

13.5.2 SDF related states (SCSM-SDF)

The interaction with the SDF is possible from any state of the SCF. In the following subclauses, the SDF-related states are specified. The model describes the relationship of one SCF with one SDF. If an SCF needs to access another SDF a new FSM should be instantiated.

In what follows, the states and events are enumerated independent of the rest of the SCSM; the discussion is accompanied by figure 12-23.

13.5.2.1 State 1: "Idle"

The following event is considered in this state:

(e1) Bind_Request
This is an internal event, caused by the need of the SL to create an association with an SDF in order to begin accessing data. This event causes a transition to the State 2, **Wait for subsequent requests**.

13.5.2.2 State 2: "Wait for subsequent requests"

In this state, subsequent operations to be sent with the **Bind** operation (in the same message) to the SDF are expected. The following two events are considered in this state:

(e2) Request_to_SDF
This is an internal event caused by the reception of an operation. The operation is buffered until the reception of a delimiter (or a timer expiration). The SCSM remains in the same state; and

(e3) Request_to_SDF_with_Bind
This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing the argument of the **Bind** operation and other operations' arguments, if any, is sent to the SDF. This event causes a transition out of this state to the State 3, **Wait for Bind result**.

13.5.2.3 State 3: "Wait for Bind result"

In this state, the SCF is waiting for the response from the SDF. Two events are considered in this state:

(E4) Bind_Error
This is an external event, caused by the reception of a error to the **Bind** operation previously issued to the SDF. This event causes a transition out of this state to State 1, **Idle**.

(E5) Response_from_SDF_with_Bind
This is an external event, caused by the reception of a **Bind** result combined with the responses to other operations previously issued to the SDF (if any). This event causes a transition to State 4, **SDF Bound**; and

13.5.2.4 State 4: "SDF Bound"

In this state, the SCF has established an authenticated access to the SDF, and is waiting for requests to the SDF from the SL or is waiting for responses to the operations previously issued to the SDF. Three events are considered in this state:

(e6) Request_to_SDF
This is an internal event, caused by the need of the SL to access data in the SDF. The SCSM remains in the same state;

(E7) Response_from_SDF
This is an external event, caused by the reception of responses to the operations previously issued to the SDF. The SCSM remains in the same state; and

(e8) Unbind_request
This is an internal event, caused by the need of the SL to terminate the authenticated access to the SDF. This event causes a transition state to State 1, **Idle**.

In addition to the above model, an System Description Language (SDL) description of SCSM is shown in annex A.

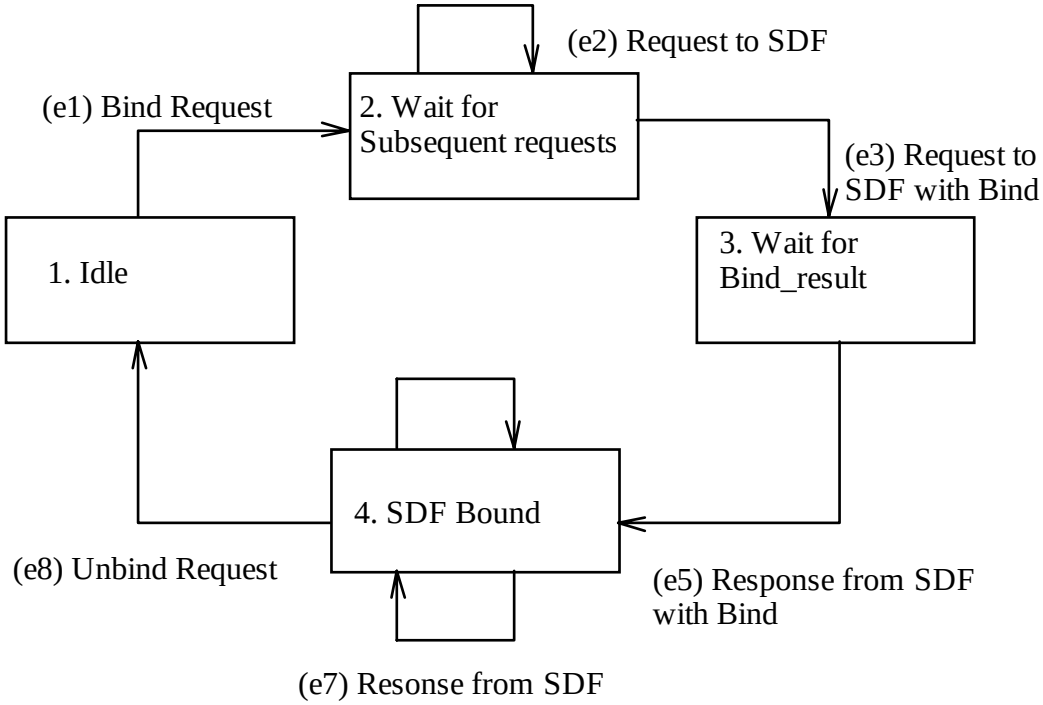


Figure 12-23: The SDF-related states

13.5.3 SCF Related states

The interaction with another SCF is possible from any state of the invoking SCF. In the following subclauses, the SCF related states are specified. The models describe the relationship of one SCF with another SCF on both sides of the relationship. If an SCF needs to access another SCF a new FSM should be instantiated.

In what follows, the states and events are numbered independently of the rest of the SCSM. The first subclause describes the SCF FSM when the SCF has initiated a dialogue with another SCF. The second subclause describes the other part of the SCF-SCF relationship, i.e. when the SCF is responding to a request from another SCF.

13.5.3.1 Controlling SCF FSM (SCSM-Con)

13.5.3.1.1 State 1: Idle

The following event is considered in this state:

(e1) SCF Bind_Request
This is an internal event, caused by the need of the SL to establish a relationship with another SCF and to receive co-operation from it, to proceed further with a call. This event causes a transition to state 2, "Preparing Handling Information Request". It causes the SCFBind operation to be issued to the supporting SCF:

13.5.3.1.2 State 2: Preparing Handling Information Request

In this state, the SCF is preparing a request to the supporting SCF to be assisted in the processing of the call. The following events are considered in this state:

(e2) Send Handling Information Request sent: it causes a HandlingInformationRequest operation to be sent the supporting SCF. This event causes a transition to state 3: "Waiting for Bind Result";

(E24) SCF Bind error
This is an external event caused by SCF Bind operation error reception. It causes a transition back to state1, "Idle".

(e25) SCFUnbind
This is an internal event causing sending of SCFUnbind (e.g. because the SCFBind result timer is expired) . It causes a transition back to state Idle'.

13.5.3.1.3 State 3: Waiting for Bind Result

In this state, the SCF is waiting the result of the SCFBind request. The following events are considered in this state:

(E3) SCF Bind error
This is an external event caused by the reception of the error result of the previously issued SCFBind operation. It causes a transition back to state 1, "Idle".

(E4) SCF Bind successful
This is an external event caused by the reception of the successful result of the previously issued SCFBind operation. It causes a transition to state 4, "Assisted Mode".

(e26) Timer expiration
This is an internal event caused by the expiration of a guard timer. It causes a transition to state 'Preparing SCF Unbind Request.

13.5.3.1.4 State 4: Assisted Mode

In this state, the SCF has sent a request to the supporting SCF to be assisted in the processing of the call. A relationship has been created between the two SCFs. They are cooperating to provide functionality's needed to process a call. The following three events can occur in this state:

(E5) Assist_Completed
This is an external event caused by the reception of the result of a previously issued operation. The result indicates that the SCF can continue with the processing of the call and that the assistance of an assisting SCF is not further required. This event causes a transition to the state 7, "Preparing SCF Unbind request**";** unless there is a result pending, in which case it stays in state 4, pending the receipt of the requested result.

This event is caused by a reception of the following operation:

HandlingInformationResult

(e6) End_Assist
This is an internal event caused by the need of the SL to stop its relationship with the supporting SL. This event causes a transition to the state 7, "Preparing Unbind request".

(E7) Additional_Information_Required_from_Supporting_SCF
This is an external event caused by the reception of a operation from the supporting SCF requesting additional information to be able to assist the controlling SCF. This event causes a transition out of this state to state 5 , "Preparing Additional Information". This event is caused by a reception of following operation:

ProvideUserInformation
NetworkCapabilityRequest

(e8) Notification_Provided_to_Supporting_SCF_without_Confirmation_Request
This is an internal event caused by the need of the SL to provide notification for the supporting SCF without the confirmation request. This event causes a transition to the same state, state 4 , "Assisted Mode". It causes one of the following operations to be issued to the supporting SCF:

NotificationProvided
ReportChargingInformation

(e21) Notification_Provided_to_Supporting_SCF_with_Confirmation_Request
This is an internal event caused by the need of the SL to provide notification for the supporting SCF with the confirmation request. The controlling SCF awaits the confirmation from the supporting SCF and this event causes a transition out of this state to state 6, "Waiting for Response from Supporting SCF". It causes one of the following operations to be issued to the supporting SCF:

ConfirmedReportChargingInformation
ConfirmedNotificationProvided

(e22) Handling Information Request sent
This is an internal event caused by the need of the SL, having received the previously requested Handling Information Result, to request additional information to the supporting SCF. It causes the following operation to be issued to the supporting SCF:

HandlingInformationRequest
The controlling SCF waits the response from the supporting SCF, and this event does not cause any transition out of this state.

(E23) Confirmation Provided
This is an external event caused by the reception of one of the following operation results:

Return Result of ConfirmedReportChargingInformation
Return Result of ConfirmedNotification Provided
It does not cause any transition out of this state

(E19) Establish Charging Record
This is an external event caused by the reception of the **EstablishChargingRecord** operation. It does not cause any transition out of this state.

(E20) Request Notification
This is an external event caused by the reception of the **RequestNotification** operation. It does not cause any transition out of this state.

(E24) Referral from Supporting SCF
This is an external event cause by the reception of a Referral error as the response to the **HandlingInformationRequest** operation. This event is communicated to the internal SL and does not cause a state transition out of this state.

13.5.3.1.5 State 5: Preparing Additional Information

In this state, the SCF is preparing additional information for the supporting SCF so that it can be assisted in the processing of the call. Two events are considered in this state:

(e11) Additional_Information_Provided_to_Supporting_SCF
This is an internal event. The SCF has collected all the information needed from the supporting SCF and sends the result to the previously issued operation requesting additional information. This event causes a transition to state 4 , Assisted Mode. It causes the return result of the following operation to be issued to the supporting SCF:

Return Result of ProvideUserInformation
Return Result of NetworkCapability
(e12) End_Assist
This is an internal event caused by the need of the SL to stop its cooperation with the supporting SL. This event causes a transition to the state 7 , Preparing SCF Unbind request .

13.5.3.1.6 State 7: Preparing SCFUnbind request

In this state, the SCF is preparing to send a SCFUnbind operation, to terminate the relationship with the supporting SCF. It causes the following event:

(e17) SCF Unbind request
This is an internal event that occurs when SCF Unbind operation is sent. This closes the relationship and causes a transition back to the state 1, "Idle".

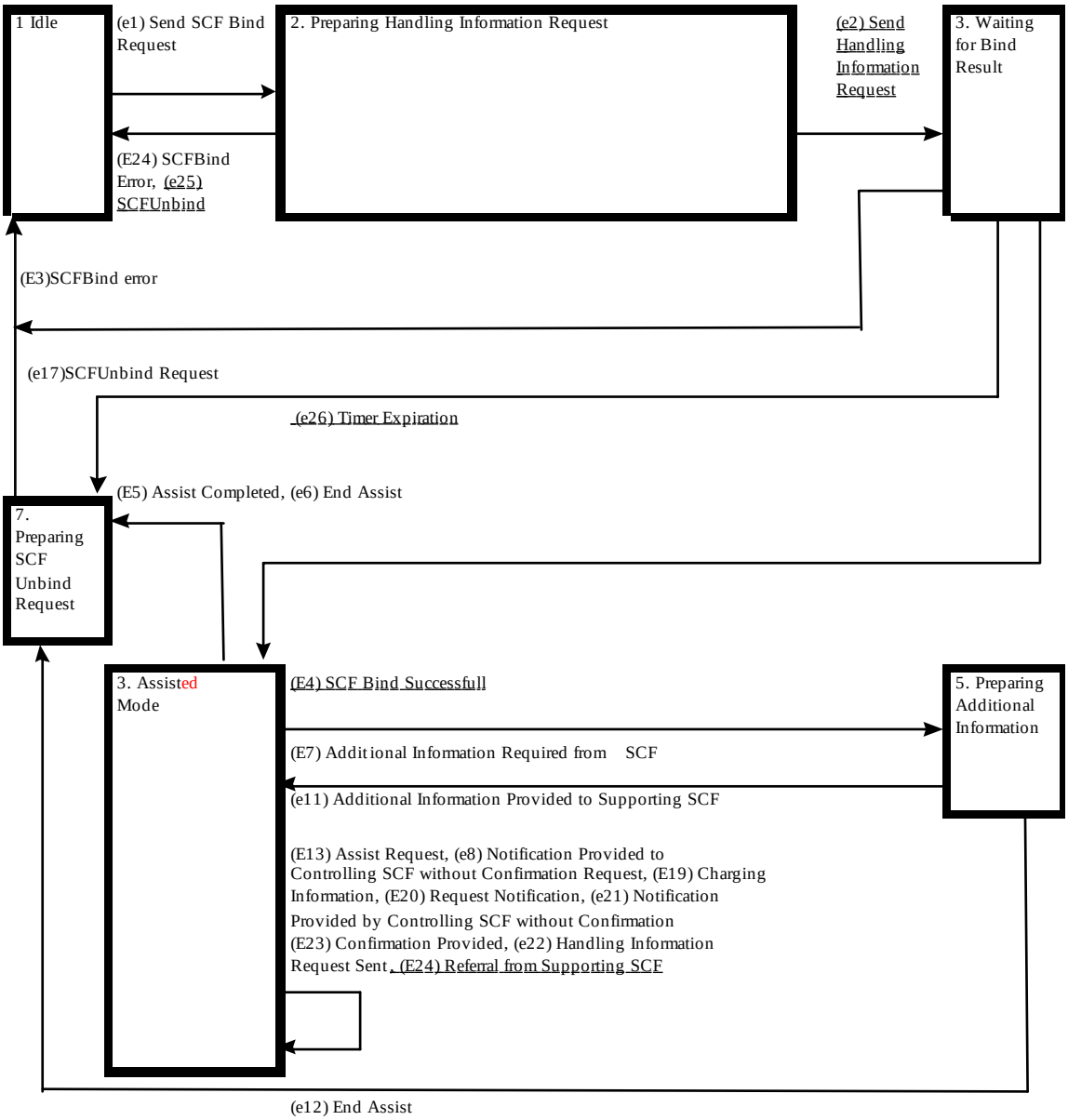


Figure 12-24: SCSM-Con (Controlling SCF FSM)

13.5.3.2 Supporting SCF FSM (SCSM-Sup)

13.5.3.2.1 State 1: Idle

In this state the SCF is waiting for request from other SCFs. No SLPI is started yet. Only the following event is accepted in this state:

(E1) SCF Bind Request Received
This is an external event caused by the reception of a SCF Bind operation from the controlling SCF for providing assistance in the processing of the call. This event causes a transition out of this state to state 2, "Processing SCFBind" .

13.5.3.2.2 State 2: Processing SCF Bind

In this state, the SCF is processing SCF Bind operation. The following events are considered in this state:

(e2) Send positive SCFBind result. This is an internal event causing the sending of a positive **Return Result from SCFBind**. It does not causes a state transition.

(e3) Bind error
This is an internal event that occurs when the supporting SCF does not establish a relationship with the controlling SCF. It causes an error **Return Result from SCF Bind** to be sent to the controlling SCF. This event causes a transition out of this state to state 1, "Idle".

(e24) Timer expiration
This is an internal event caused by guard timer expiration. It causes a transition back to state 1 "Idle".

(E4)Request from controlling SCF
This is an external event caused by the reception of **HandlingInformationRequest** operation. The **HandlingInformationRequest** operation is queued since the supporting SCF SL is involved in the SCFBind processing (it will be processed as soon as the supporting SCF FSM moves to Assisting Mode state). It does not cause a transition out of the state.

(e28) SCFBindAccepted & Handling Information Request Received. This is an internal event caused by the fact that the **SCFBind** operation is accepted and that a **HandlingInformationRequest** operation has been received. It does not cause any operation to be sent. It causes a transition to Assisting Mode state.

(E11) SCF Unbind Request received
This is an external event caused by the reception of a request from controlling SCF to end the relationship through SCFUnbind operation. This event causes a transition back to state 1 " Idle".

13.5.3.2.3 State 3: Assisting mode

In this state, the SCF is sending operations to provide assistance to the controlling SCF and receives indications from the controlling SCF. The following events are considered in this state.

(e5) Assist_Provided
This is an internal event caused by the sending of a response to the assistance request previously received from the controlling SCF, provided that the SCF has not foreseen any further assistance. There is no state transition. It causes the following operation to be issued to the controlling SCF:

HandlingInformationResult

(E6) SCF Unbind request received
This is an external event caused by the reception of an **SCFUnbind** request from controlling SCF. It gives an indication from the controlling SCF that the SCF-SCF relationship needs to be ended. This event causes a transition back to state 1, "Idle".

(e7) Information_Needed_from_Controlling_SCF
This is an internal event caused by the need of the SL to receive additional information from the controlling SCF to provide assistance. This event causes a transition out of this state to state 4 "Waiting for Additional Information". It causes the following operation to be issued to the controlling SCF:

ProvideUserInformation
NetworkCapabilityRequest

(E8) Notification_Provided_by_Controling_SCF_without_Confirmation_Request
This is an external event caused by the receipt of the notification which was requested to the controlling SCF. This event causes a transition to the same state, state 3 , "Assisting Mode". This event is caused by a reception of the following operations:

ReportChargingInformation
NotificationProvided

(E20) Notification_Provided_by_Controling_SCF_with_Confirmation
This is an external event caused by the reception of one of the following operations:

ConfirmedReportChargingInformation
ConfirmedNotificationProvided

This event does not causes a transition out of this state

(e12): Notification Requested
This is an internal event caused by the sending of a **RequestNotification**-operation to the controlling SCF. This event causes a transition to the same state, state 3, "Assisting Mode".

(e13) Charging Information

This is an internal event caused by the sending of a **EstablishChargingRecord** operation to the controlling SCF. This event causes a transition to the same state, state 3, "Assisting Mode".

(e21) ConfirmationProvided

This is an internal event caused by the need of the SL to respond the confirmation previously received request. It causes one of the following operation to be issued to the controlling SCF:

ReportChargingInformationConfirmation
NotificationProvidedConfirmation

This event does not causes a transition out of "Assisting mode" state.

(e17) Additional_Information_Result

This is an internal event caused by the sending of a response to the previously received request from the controlling SCF." It causes the following operation to be issued to the controlling SCF:

HandlingInformationResult

This event does not causes a transition out of this state 3 "Assisting Mode"

(e22) Timer expiration

This is an internal that occurs when no operation has been received from the controlling SCF for an appropriate period of time. The supporting SCF determines that the relation is no longer exists and moves back to "Idle" state.

(E25) Received Handling Information Request

This is an external event caused by the reception of a new **HandlingInformationRequest** operation. It can be accepted only if there is no previous **HandlingInformationResult** operation pending. This event does not cause a state transition out of this state.

(e31)Referral to controlling SCF

This is an internal event caused by the decision to instruct the controlling SCF to send the pending **HandlingInformationRequest** to another SCF for processing. A Referral error for the **HandlingInformationRequest operation** is sent to controlling SCF. It does not cause a transition out of Assisting Mode state.

13.5.3.2.4 State 4: Waiting for Additional Information

In this state, the SCF waits for information to be provided by the controlling SCF. This information should help the SCF to provide assistance to the controlling SCF. The following two event are considered in this state:

(E14) Additional_Information_Provided_by_Controlling_SCF

This is an external event caused by the reception of the response to a previously issued operation requesting further information. This event causes a transition out of this state to state 3, "Assisting mode". This event is caused by a reception of the following operation's return result:

Return Result of ProvideUserInformation
Return Result of NetworkCapability

(E15) SCF Unbind request received

This is an external event caused by the reception of an **SCFUnbind** request from controlling SCF. It gives an indication from the controlling SCF that the SCF-SCF relationship needs to be ended. This event causes a transition back to state 1, "Idle".

(e22) Timer expiration

This is an internal that occurs when no operation has been received from the controlling SCF for an appropriate period of time. The supporting SCF determines that the relation no longer exists and moves back to"Idle"state.

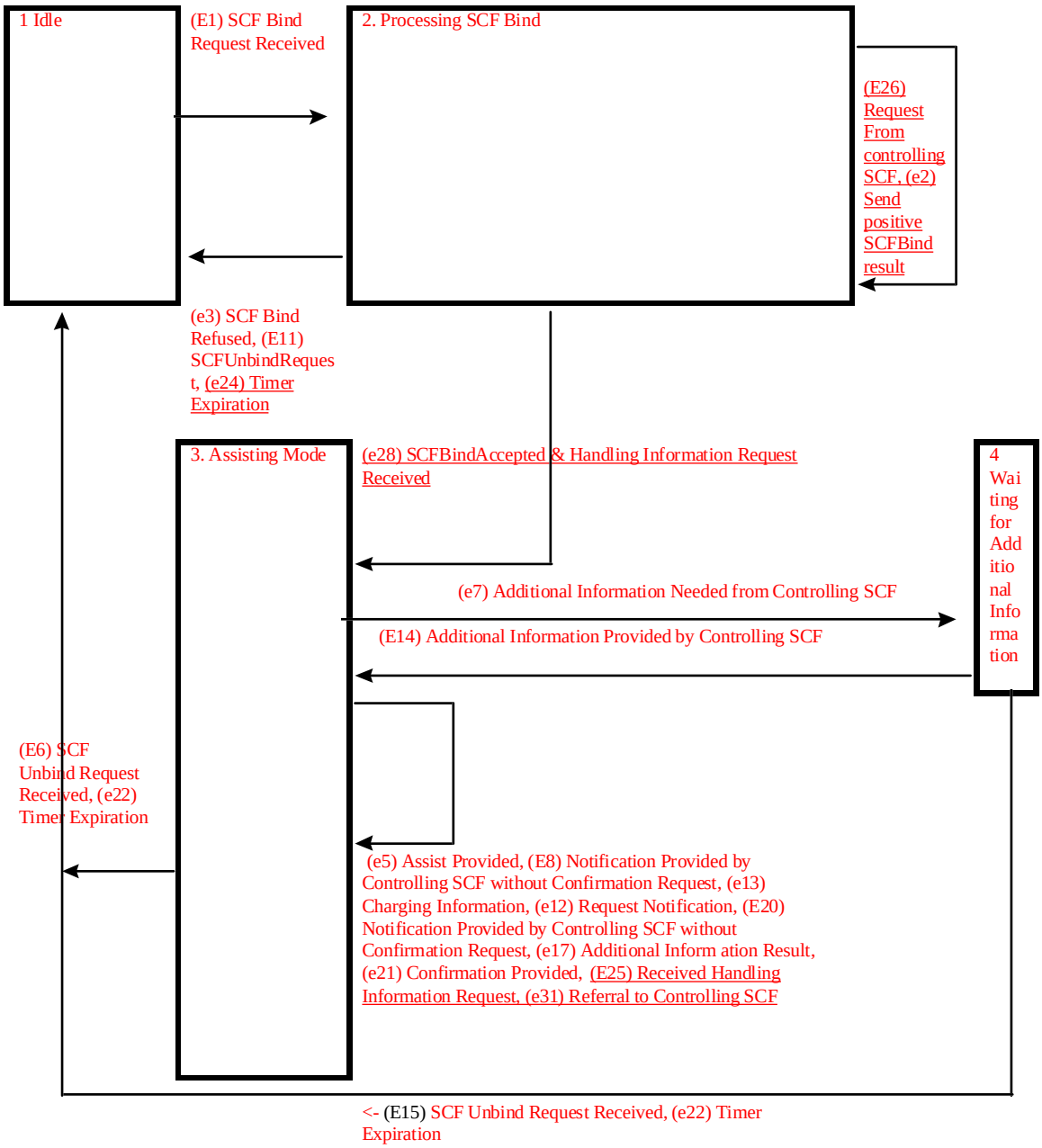


Figure 12-25: SCSM-Sup (Supporting SCF FSM)

13.5.3.2.5 SCF state transition models for chaining

As for the chaining procedure, an SCF can act as a chaining initiator and as a chaining terminator. Therefore, there are two FSMs as described below.

In the following FSMs, the possibility of sending the SCFBind operation together with ChainedhandlingInformationRequest operation in one TC message is taken into consideration.

13.5.3.2.5.1 SCF state transition models for chaining initiation (SCSM-ChI)

The Finite State Machine for an SCF interacting with another SCF when acting as a chaining initiator is depicted in figure 12-26.

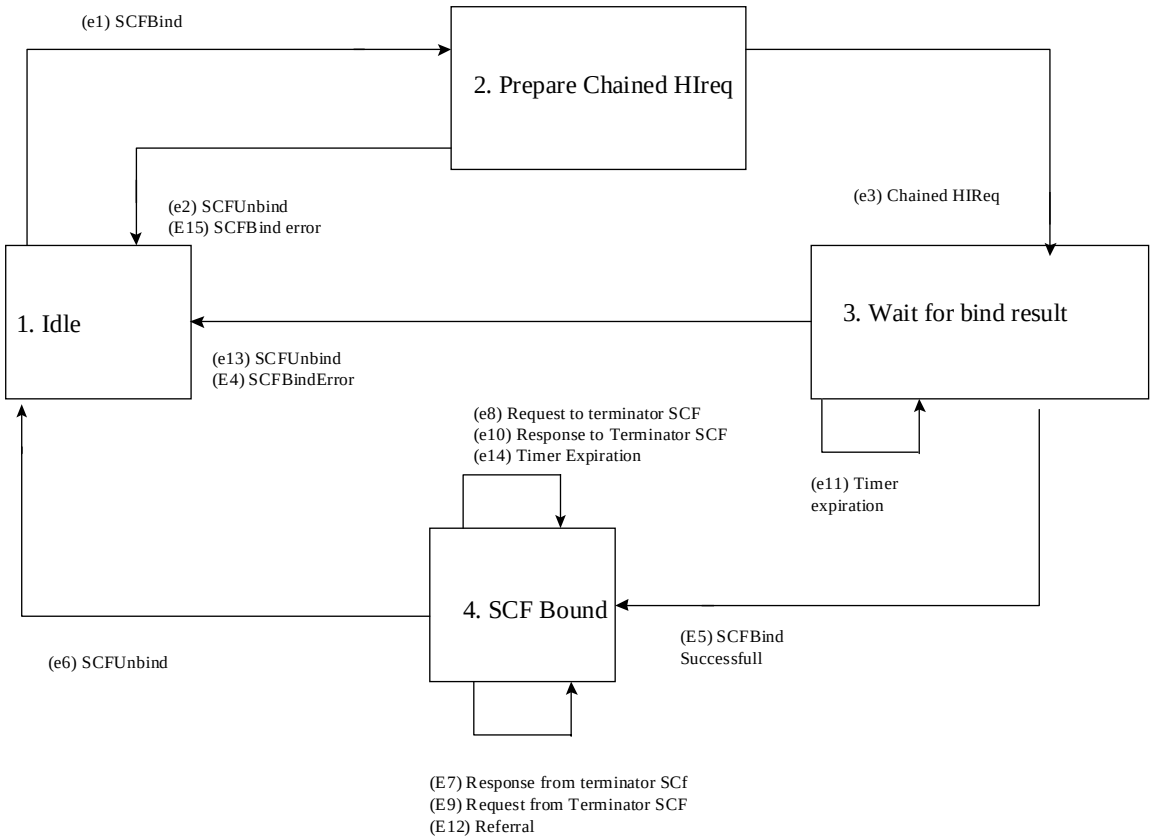


Figure 12-26: SCSM-ChI (Chaining Initiator SCF FSM)

State 1: "Idle"

The only event accepted in this state is:

(e1) Send SCFBind
This is an internal event caused by the need to send a SCFBind operation to the SCSM-ChT. The scfBind operation is prepared and saved. This event causes a transition out of this state to State 2 "Wait for **Chained Handling Information Request**".

State 2: "Wait for Chained Handling Information Request"

In this state, a scfBind operation has been prepared for sending and the FSM is waiting to see if a ChainedHandlingInformationRequest operation is to be sent with the scfBind operation. The following events can occur:

(e3) Send Bind with Requests
This is an internal event caused by either the receiving of HandlingInformationRequest operation to be chained or an internal delimiter indicating no HandlingInformationRequest operation is present. It causes the scfBind operations (and ChainedHandlingInformationRequest operation) to be sent to the SCSM-ChT. It causes a transition to state 3 "Wait for Bind Result".

(e2) SCFUnbind
This is an internal event, caused by the receiving of a SCFUnbind operation which requires chaining to the SCSM-ChT. This event causes a transition out of this state to State 1 **Idle**,

State 3: "Wait for Bind Result"

In this state, a **SCFBind** operation and an optional **ChainedHandlingInformationRequest** operation has been sent to the SCSM-ChT. The SCSM-ChT is performing the processing associated with the SCFBind operation (e.g., access authentication). Three events are considered in this state:

(E4) SCFBind_Error
This is an external event caused by the failure of the SCFBind operation previously issued to the SCSM-ChT. This event causes a transition out of this state to State 1 **Idle**;

(E5) SCFBind_Successful
This is an external event caused by the reception of the SCFBind confirmation for the SCFBind operation previously issued to the SCSM-ChT. This event causes a transition out of this state to State 4 **SCF Bound**.

(e12) SCFUnbind
This is an internal event, caused by the receiving of a SCFUnbind operation which requires chaining to the SCSM-ChT. The scfUnbind operation is sent to the SCSM-ChT, the SCF-SCF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 **Idle**.

State 4: "SCF Bound"

In this state, the access of the SCSM-ChI to the SCSM-ChT was authorized, chained operations can be sent to the SCSM-ChT, and results of chained operations coming from the SCSM-ChT are accepted. Thus, the SCSM-ChT will not further chain outgoing chained requests (except where the chaining initiator SCF is in another network). The following events are considered in this state:

(e6) SCFUnbind
This is an internal event, caused by the receiving of a SCFUnbind operation which requires chaining to the SCSM-ChT and causes the sending of the SCFUnbind operation to the SCSM-ChT. The SCF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 **Idle**;

(E7) Response_from_terminator_-SCF
This is an external event, caused by the reception of results of the operations previously issued by the SCSM-ChI. The SCSM-ChI remains in the same state;

(e8) Request_to_terminator_-SCF
This is an internal event, caused by the sending of a chained operation to the SCSM-ChT. The SCSM-ChI remains in the same state;

(E9) Request_from_Terminator_SCF
This is an external event caused by the reception of a chained operation from the SCSM-ChT. The SCSM-ChI remains in the same state.

(e10) Response_to_terminator_SCF
This is an internal event caused by the sending of a result to a previously chained operation from the SCSM-ChT. The SCSM-ChT remains in the same state.

(E12) Referral
This is an external event caused by a the reception of a referral error to a ChainedHandlingInformationRequest. It does not cause a transition from SCF Bound.

13.5.3.2.5.2 SCF state transition models for chaining termination (SCSM-ChT)

The Finite State Machine for an SCF interacting with another SCF when acting as a chaining terminator is depicted in figure 12-27.

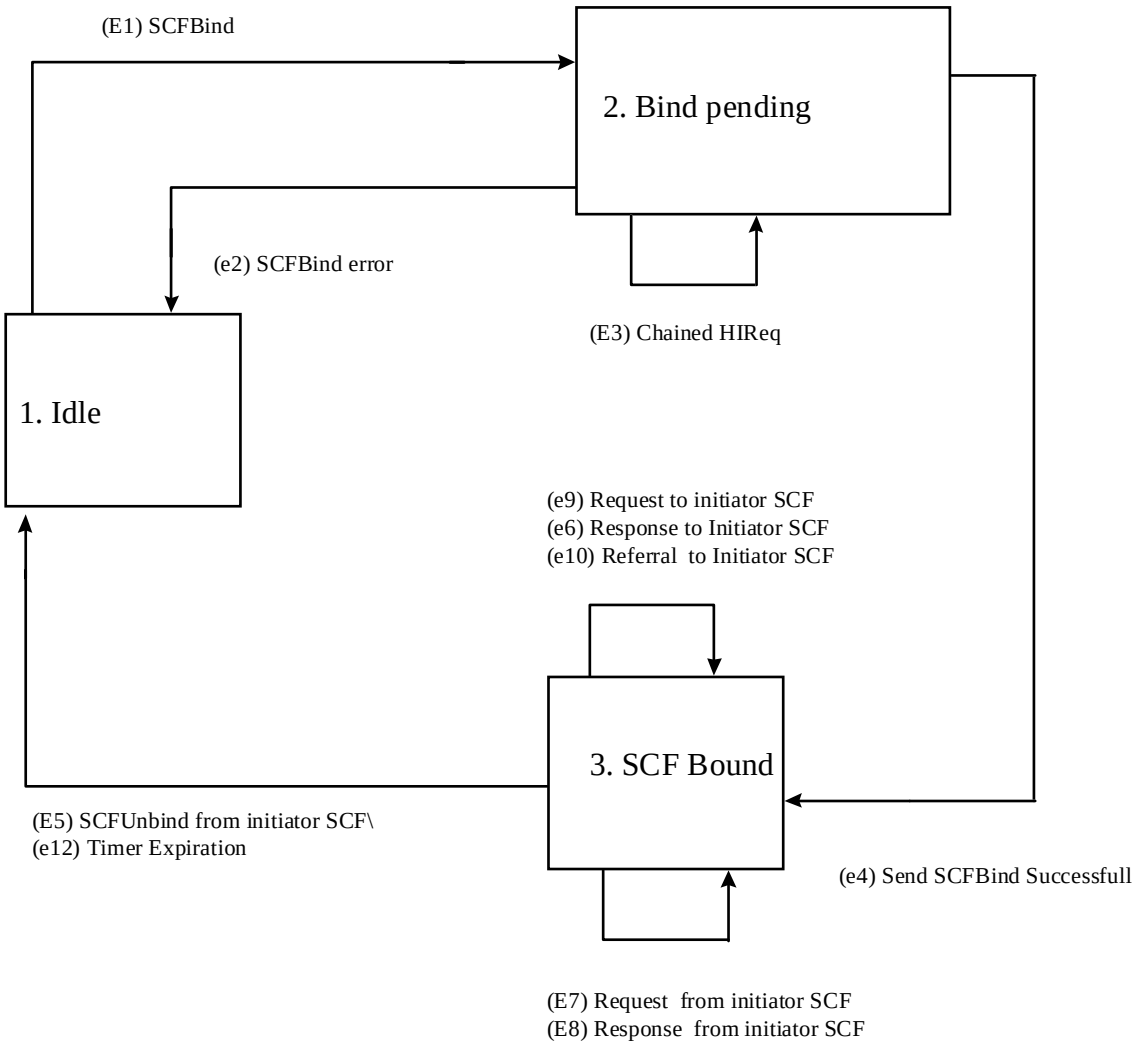


Figure 12-27: SCSM-ChT (Chaining Terminator SCF FSM)

State 1: "Idle"

The only event accepted in this state is:

(E1) SCFBind
This is an external event caused by the reception of a **SCFBind** operation from an SDSM-ChI. This event causes a transition out of this state to State 2 **Bind Pending**.

State 2: "Bind Pending"

In this state, a SCFBind request has been received from the SCSM-ChI. The SCSM-ChT is performing the SCF access control procedures associated with the SCFBind operation (e.g., access authentication). The following events are considered in this state:

(e2) SCFBind_Error
This is an internal event caused by the failure of the SCFBind operation previously issued from the SCSM-ChI. This event causes a transition out of this state to State 1 **Idle**, and a Bind error is returned to the SCSM-ChI;

(E3) ChainedHandlingInformationRequest
This is an external event caused by the reception of a ChainedHandlingInformationRequest from the chaining initiator supporting SCF. The SCSM-ChT remains in the same state;

(e4) SCFBind_Successful
This is an internal event caused by the successful completion of the SCFBind operation previously issued from the SCSM-ChI. This event causes a transition out of this state to State 3 **SCF Bound**.

State 3: "SCF Bound"

In this state, the access of the SCSM-ChI to the SCSM-ChT was authorized and chained operations coming from the SCSM-ChI are accepted. Besides waiting for requests from the SCSM-ChI, the SCSM-ChT can send operations in this state as responses to previously received operations, if any. The following events are considered in this state:

(E5) SCFUnbind_from_SCF
This is an external event, caused by the reception of the SCFUnbind operation from the SCSM-ChI. The SCF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 **Idle**;

(e6) Response_to_initiator SCF
This is an internal event, caused by the sending of an operation to SCSM-ChI. The SCSM-ChT remains in the same state;

(E7) Request_from_initiator SCF
This is an external event, caused by the reception of a request from the SCSM-ChI. The SCSM-ChT remains in the same state.

(E8) Response_from_Initiator_SCF
This is an external event caused by the reception of an operation as a result to a previously chained operation sent to the SCSM-ChI. The SCSM-ChT remains in the same state.

(e9) Request_to_initiator_SCF
This is an internal event caused by the sending of a operation to the SCSM-ChI in order to request some type of processing in the SCSM-ChI. The SCSM-ChT remains in the same state.

(e12) Timer Expiration
This is an internal event caused by a timer expiration. It cause a transition from state "SCFBound" back to state 1 'Idle

(e 10) Referral to Initiator SCF
This is a internal event caused by the sending of a Referral error as a response to the pending ChainedHandlingInformationRequest to the chaining initiator supporting SCF. SCSM-ChT stays in the same state.

13.5.4 CUSF Related states (SCSM-CUSF)

Figure 12-28 shows State Diagram of the SCSM as relevant to the procedures concerning the FSM for CUSF part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses.

An instance of FSM for CUSF is created on reception of a instruction from the CUSF . The instance is released when the state of the instance of FSM for CUSF transits to the state "Idle".

The letter "N" is added to the head of the number of each state and event in FSM for CUSF to distinguish the states and events in FSM for CUSF from those in other FSMs in the SCSM.

13.5.4.1 State N1: "Idle"

The following event is considered in this state:

(Ne1) CUSF_Initiate_Control_Requested
This is an internal event caused by the SL's need to have a new control relationship with CUSF. The FSM for CUSF requests to transmit the **initiateAssociation** operation to the CUSF. This event causes a transition to the state S2, **Preparing CUSF Instructions**. (i)

(NE2) (ii) Query_from_CUSF
This is an external event, caused by a reception of one of the following operations:

activationReceivedAndAuthorized (for TDP-R)
componentReceived (for TDP-R)
associationReleaseRequested (for TDP-R) (i)
This event causes a transition to State N2, Preparing CUSF Instructions.

(NE3) (ii) Notification_from_CUSF
This is an external event, caused by a reception of one of the following operations:

activationReceivedAndAuthorized (for Trigger Detection Point - Notification (TDP-N))
componentReceived (for TDP-N)
associationReleaseRequested (for TDP-N) (i)
This event causes a transition back to the same state.

12.5.4.2 State N2: "Preparing CUSF Instructions"

In this state, FSM for CUSF prepares appropriate instructions to the CUSF.

The following events are considered in this state:

(Ne4) (ii) Processing_completed
This is an internal event. In this case, the SCF has completed the processing of the instructions to the CUSF. This event causes the following operation to be sent to the CUSF and a transition to State N1, Idle:

releaseAssociation
To further describe the procedures relevant to this state, the state is divided into two sub-states, which are described in the following two subclauses (this subdivision is illustrated in figure 12-29).

13.5.4.1.1 State N2.1: "Preparing CUSF Instructions"

In this state, FSM for CUSF determines whether the BCUSM processing will be resumed or not, and deals with a EDP relating processing.

The following events are considered in this state:

(Ne2.1) Event_Request
This is an internal event caused by the SL when there is a need to send such an operation to the CUSF . It caused one or more of the **requestReportBCUSMEvent**(iv) operations to be issued to the CUSF:

This event causes a transition back to state N2.1 Preparing CUSF Instructions.

(Ne2.4) Request_Release_Association
This is an internal event caused by the SL when it needs to release the association between the user and the network. It caused the **releaseAssociation** operation to be issued to the CUSF.

This event maps into the FSM for CUSF in the SCSM event (Ne4).

13.5.4.1.2 State N2.2: "Waiting for Notification or Request"

In this state, FSM for CUSF waits for a notification or a request from the CUSF .

The following events are considered in this state:

(NE2.5) EDP-R
This is an external event caused by the reception of the following operation(s):

componentReceived (for EDP-R) (iii)
associationReleaseRequested (for EDP-R) (i)
This event causes a transition to state N2.1 Preparing CUSF Instructions.

(NE2.6) Not_Last_EDP-N

This is an external event caused by the reception of the following operation(s):

componentReceived (for EDP-N) (iii)
associationReleaseRequested (for EDP-N) (i)

In this case, there is still an outstanding armed EDP. This event causes a transition back to State N2.2 Waiting for Notification or Request.

(NE2. 7) Last_EDP-N

This is an external event caused by the reception of the following operation(s)

componentReceived (for EDP-N) (iii)
associationReleaseRequested (for EDP-N) (i)

This event maps into the FSM for CUSF in the SCSM event (Ne4). (ii)

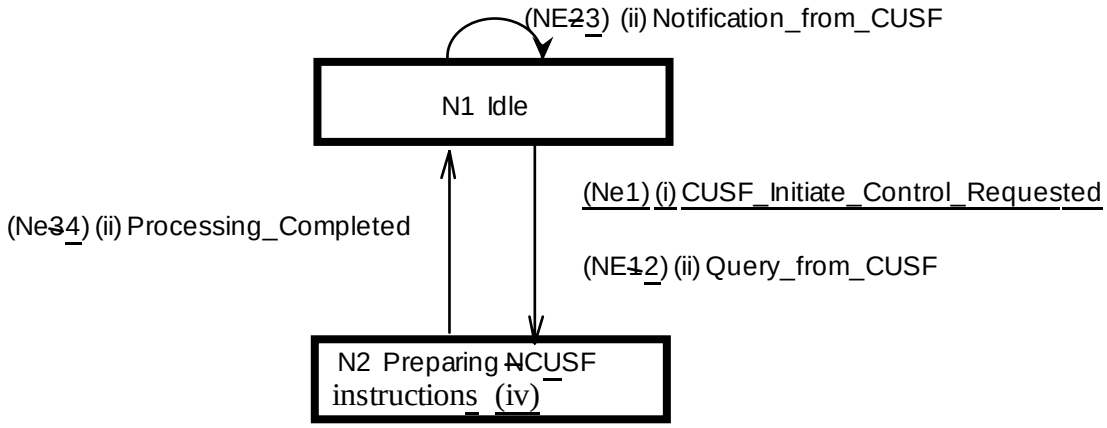
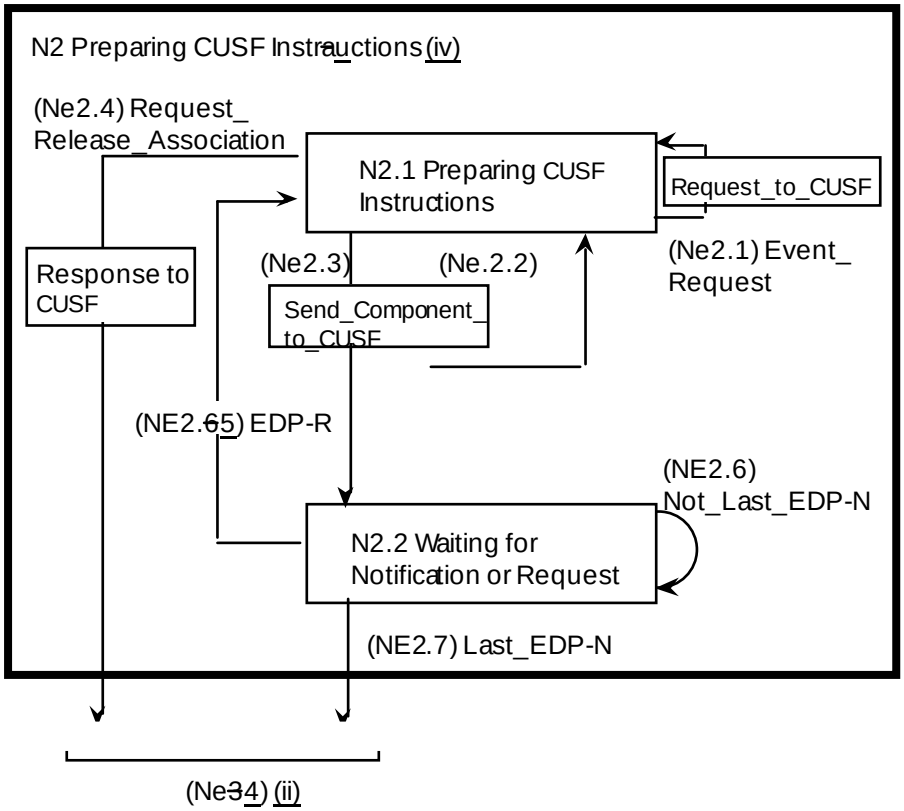


Figure 12-28: FSM for CUSF



note (Ne2.2) Request_Send_Component (monitor not required)
(Ne2.3) Request_Send_Component (monitor required)

Figure 12-29: Sub-states of state N2

13.5.5 USI_SCF FSM

The USI_SCF FSM illustrates the SCF monitoring or not the receipt of the UTSI IE. This FSM has two states which are "Monitoring UTSI information" and "Idle":

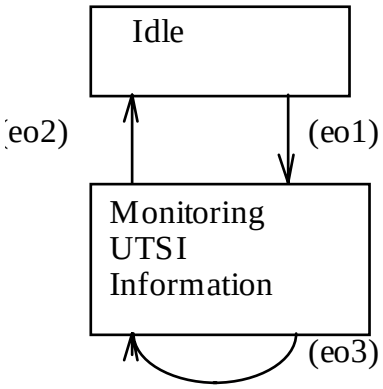


Figure 12-30: USI_SCF FSM

The USI_SCF FSM transitions are defined in the following way:

- (eo1): the SCF requests the SSF to monitor the receipt of an UTSI IE with a given *USIServiceIndicator* value.
- (eo2): The SCF is no longer interested in the receipt of an UTSI IE with the given *USIServiceIndicator* value.
- (eo3): The SCF sends an STUI IF to the User and/or receives an UTSI IE from the User with the given *USIServiceIndicator* value. This is also allowed in state "Idle" with no resulting state transition..

With the same operation, the SCF requests the SSF to monitor or to stop monitoring the receipt of an UTSI IE with a given *USIServiceIndicator* value.

NOTE: As an SCF controls or monitors the call, the SSF FSM is in any state except "Idle"; but the USI mechanism does not cause any transition in the SSF FSM.

14 SRF AE procedures

14.1 General

This subclause provides the definition of the SRF AE procedures related to the SRF-SCF interface. The procedures are based on the use of SS7; other signalling systems can be used.

Other capabilities may be supported in an implementation-dependent manner in the IP, SSP or SN.

The AE, following the architecture defined in ITU-T Recommendations Q.700 [19], Q.1400 [30] and in ETS 300 287-1 [7] includes TCAP and one or more ASEs called TC-users. The following subclauses define the TC-user ASE and SACF & MACF rules, which interface with TCAP using the primitives specified in ETS 300 287-1 [7].

The procedure may equally be used with other message-based signalling systems supporting the application layer structures defined.

In case interpretations for the AE procedures defined in the following differ from detailed procedures and the rules for using TCAP services, the statements and rules contained in the detailed clauses 17 and 18 shall be followed.

Information on the SCF - External SRF communication in the relay case can be found in subclause 3.1.2.1.

14.2 Model and interfaces

The functional model of the AE-SRF is shown in figure 13-1; the ASEs interface to TCAP (to communicate with the SCF) as well as interface to the maintenance functions. The scope of the present document is limited to the shaded area in figure 13-1.

The interfaces shown in figure 13-1 use the TC-user ASE primitives specified in ETS 300 287-1 [7] (interface (1)}and N-Primitives specified in ETS 300 009-1 [3] (interface (2)). The operations and parameters of INAP are defined in clauses 3 to 10.

14.3 Relationship between the SRF FSM and maintenance functions/bearer connection handling

The primitive interface between the SRF FSM and the maintenance functions is an internal interface and is not subject for standardization in IN CS2.

The relationship between the bearer connection handling and the SRF FSM may be described as follows for the case of a call initiated by the SSF: when a call attempt is initiated by the SSF, an instance of an SRF FSM is created.

The SRF FSM handles the interaction with the SCF FSM and the SSF FSM.

The management functions related to the execution of operations received from the SCF are executed by the SRF Management Entity (SRME). The SRME interfaces the different SRF Call State Models (SRSM) and the FEAM. Figure 13-2 shows the SRF FSM structure.

The model associates a FSM with each initial interaction request from the SCF. Thus, multiple initial requests may be executed concurrently and asynchronously by the SRF, which explains the need for a single entity that performs the tasks of creation, invocation, and maintenance of the SRSM objects. This entity is called the SRME. In addition to the above tasks, the SRME maintains the dialogues with the SCF and SSF on behalf of all instances of the SCSM. In particular, the SRME:

1 Interprets the input messages from other FEs and translates them into corresponding SRSM events; and

2 Translates the SRSM outputs into corresponding messages to other FEs.

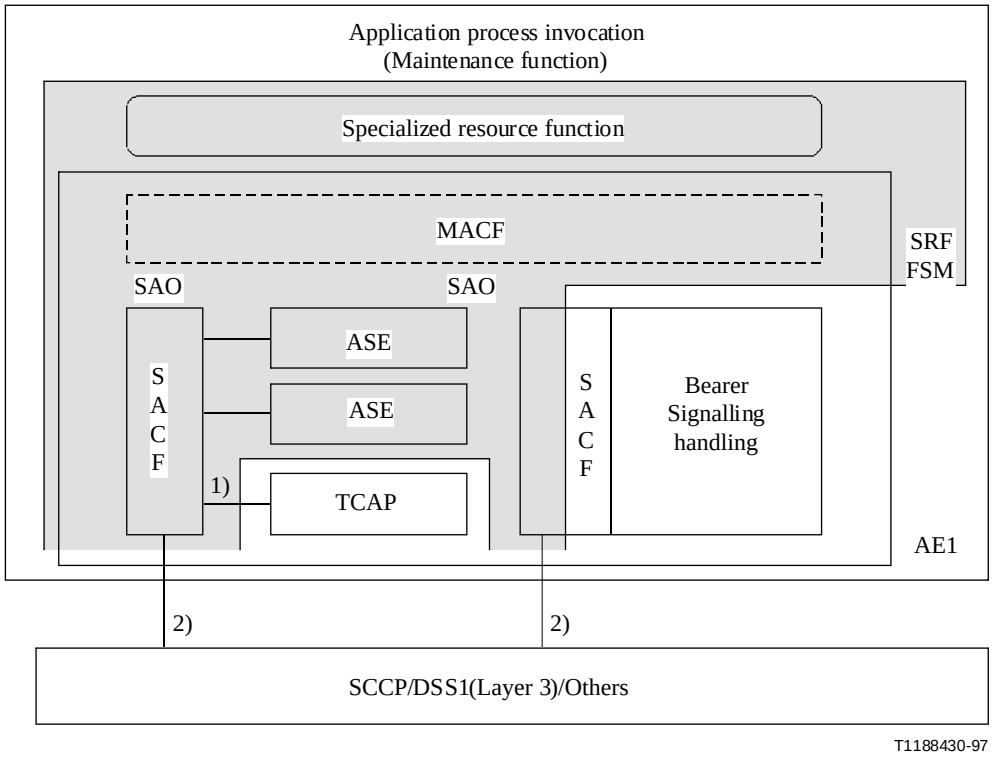
NOTE: Such a request from the SCF is executed by the SCSM when it is in its state 4.

Finally, the FEAM relieves the SRME of low-level interface functions. The FEAM functions include:

1 Establishing and maintaining the interfaces to the SSF and SCF;

2 Passing (and queueing when necessary) the messages received from the SSF and SCF to the SRME; and

3 Formatting, queueing (when necessary), and sending messages received from the SRME to the SSF and SCF.



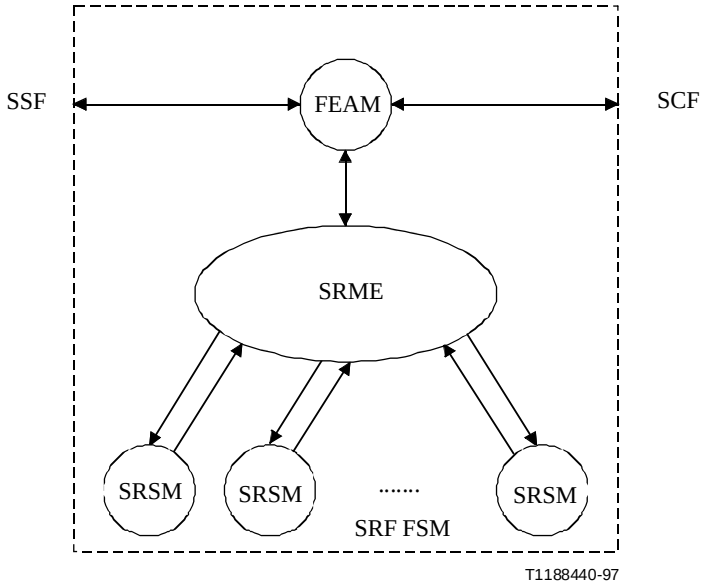
AEI Application entity invocation
SRF Specialized resource function
FSM Finite state machine
MACF Multiple association control function
SACF Single association control function
SAO Single association object

1) TC-primitives or Q.932-primitives.

2) N-primitives.

NOTE-The SRF FSM includes serveral finite state machines.

Figure 13-1: Functional model of SRF AE



FEAM Functional entity access manager
SRME SRF management entity
SRSM SRF call state model

Figure 13-2: SRF FSM structure

14.4 The SRSM

The SRSM is presented in Figure 13-3. In what follows, each state is described in a separate subclause together with the events that cause a transition out of this state. Finally, the outputs are presented within smaller rectangles than the states are; unlike the states and events, the outputs are not enumerated.

Each state is discussed in the following subclauses. General rules applicable to more than one state are addressed here.

One component or a sequence of components received in one or more TCAP messages may include a single operation or multiple operations, and it is processed as follows:

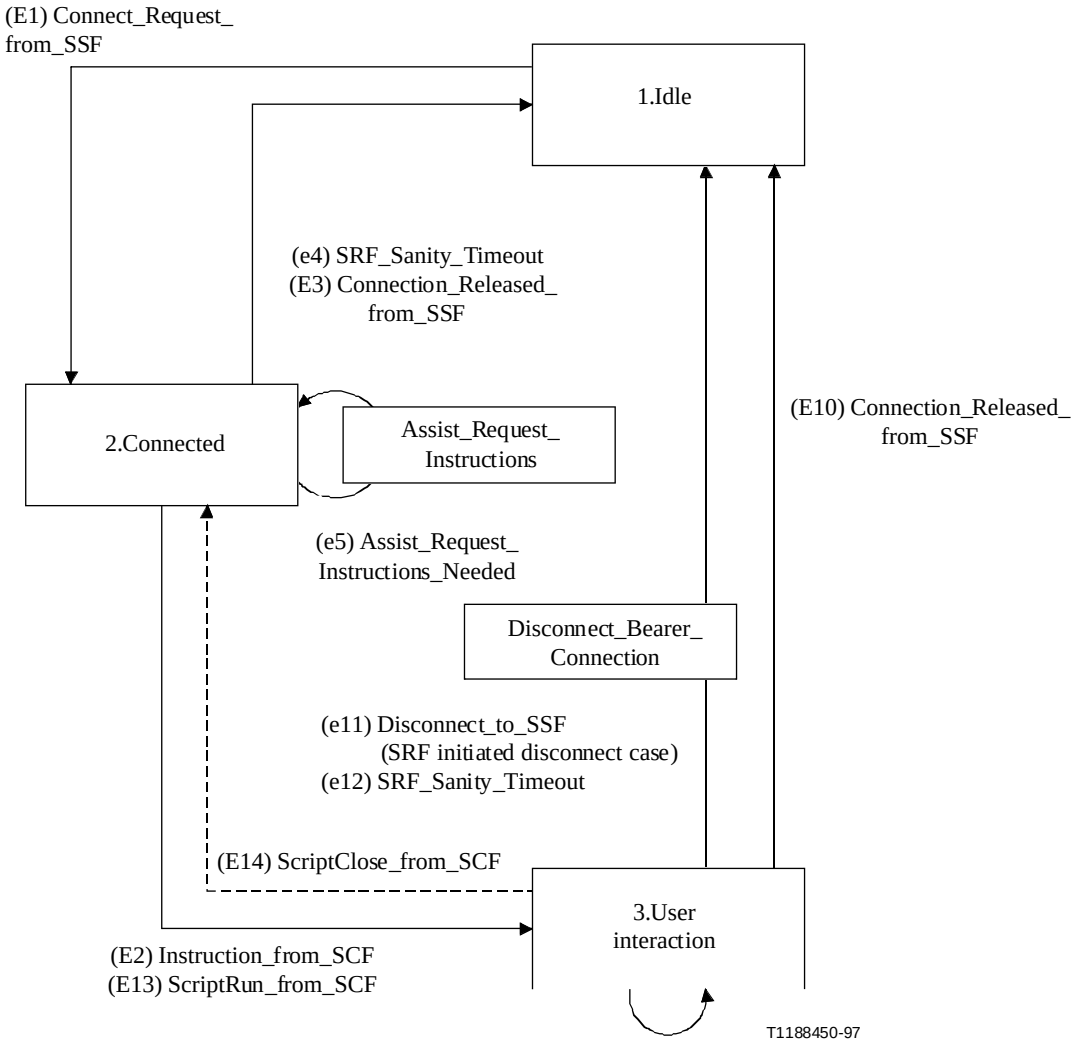
- The SRSM processes the operations in the order in which they are received.
- The SRSM examines subsequent operations in the sequence. When a Cancel (for PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage) operation is encountered in the sequence in state user interaction, it executes it immediately. In all other cases, the SRSM queues the operations and awaits an event (such an event would be the completion of the operation being executed, or reception of an external event).
- If there is an error in processing one of the operations in the sequence, the SRF FSM processes the error (see below) and discards all remaining operations in the sequence.
- If an operation is not understood or is out of context (i.e. it violates the SACF rules defined by the SRSM) as described above, the SRF FSM processes the error according to the rules given in subclause 18.1.1.2 (using TC-U-REJECT or the operation error UnexpectedComponentSequence).

In any state, if there is an error in a received operation, the maintenance functions are informed. Generally, the SRSM remains in the same state in which it received the erroneous operations, however different error treatment is possible in specific cases as described in clause 16; depending on the class of the operation, the error could be reported by the SRF to the SCF using the appropriate component (see ETS 300 287-1 [7]).

In any state, if the dialogue with the SCF (direct SCF-SRF case) is terminated, then the SRSM returns to idle state after ensuring that all resources allocated to the dialogue have been de-allocated. The SRF shall remain connected to the SSF as long as it has PlayAnnouncement operations active or buffered. The resources allocated to the call will be de-allocated when all announcements are completed or when the SSF disconnects the bearer connection (i.e. call party release).

In any state (except idle), if the SSF disconnects the bearer connection to the SRF before the SRF completes the user interaction, then the SRSM clears the call and ensures that all SRF resources allocated to the call have been de-allocated. Then it transits to the idle state.

The SRSM has an application timer, T_{SRF}, whose purpose is to prevent excessive call suspension time. This timer is set when the SRF sends Setup Response bearer message to the SSF (SSF relay case) or the AssistRequestInstructions operation (Direct SCF-SRF case). This timer is stopped when a request is received from the SCF. The SRF may reset T_{SRF} on transmission of the SpecializedResourceReport operation, the return result for the PromptAndCollectUserInformation operation or the return result for the PromptAndReceiveMessage operation when there is no queued user interaction operation. On the expiration of T_{SRF}, the SRSM transits to the idle state ensuring that all SRF resources allocated to the call have been de-allocated.



(E5) Instruction_from_SCF
(E6) Cancel_from_SCF
(e7) SRF_Report_to_SCF
(e8) PlayAnnouncement/PromptAnd CollectUserInformation/
PromptAndReceiveMessage_Cancelled_to_SCF
(e9) Cancel_Error_to_SCF
(e15) ScriptEvent_to_SCF
(E16)ScriptInformation_from_SCF
(E17)ScriptRun_from_SCF

Figure 13-3: The SRSM

14.4.1 State 1: "Idle"

The idle state represents the condition prior to, or at the completion of, an instance of user interaction. This state is entered as a result of events E3, e4, E10, e11 and e12. It is exited as a result of event E1.

(E1) Connect_Request_from_SSF
This event corresponds to a bearer signalling connection request message from the SSF. The details of the bearer signalling state machine related to establishing the connection are not of interest to the FSM. The SRSM goes to state "Connected";

(E3) Connection_Released_from_SSF
This event takes place when the SRSM receives a release message from the SSF in connected state. The SRSM goes to state "Idle";

(e4) SRF_Sanity_Timeout
This event occurs when the SRSM has been in connected state for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle";

(E10) Connection_Released_from_SSF
This event takes place when the SRSM receives a release message from the SSF in user interaction state. The SRSM goes to state "Idle";

(e11) Disconnect_to_SSF
This event occurs when the SCF has enabled SRF initiated disconnect by:

- the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message from SCF (E2) or (E5) with the parameter disconnectFromIPForbidden. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the last SpecializedResourceReport operation to the SCF (e7). The SRSM goes to state "Idle"; or

- a parameter in the ScriptRun operation.

(e12) SRF_Sanity_Timeout
This event occurs when the SRSM has been in User interaction state for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

14.4.2 State 2: "Connected"

This state represents the condition of the SRSM when a bearer channel has been established between a user and the SRF but the initial PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage has not yet been received (e.g. when EstablishTemporaryConnection procedures are used). The method used to provide this bearer channel is not of interest in the FSM.

(E1) Connect_Request_from_SSF
This event corresponds to a bearer signalling connection request message from the SSF in the Idle state. The details of the bearer signalling state machine related to establishing the connection are not of interest in the SRF FSM. The SRSM goes to state "Connected".

(E2) Instruction_from_SCF
This event takes place when the first PlayAnnouncement,_PromptAndCollectUserInformation or PromptAndReceiveMessageoperation(s) from the SCF is received. The SRSM goes to state "User interaction".

(E3) Connection_Released_from_SSF
This event takes place when the SRF receives a release message from the SSF. The SRSM goes to state "Idle".

(e4) SRF_Sanity_Timeout
This event occurs when the SRSM has been connected for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

(e5) Assist_Request_Instructions_Needed
This event occurs when the AssistRequestInstructions operation is sent from the SRSM to the SCF in the absence of a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage event (E2) initiated by the presence of a PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage operation concatenated with the setup request from SSF (E1) (Direct SCF-SRF case). No state change occurs as a result of this event.

(E13) ScriptRun_from_SCF
This event takes place when the ScriptRun operation from the SCF is received. The SRSM goes to state "User Interaction".

(E14) ScriptClose_from_SCF
This event takes place when the ScriptClose operation from the SCF is received. The SRSM goes to state "Connected".

Note that this state transition is permitted only if the SRSM received ScriptRun operation from SCF previously.

14.4.3 State 3: "User interaction"

The User interaction state indicates that communication is occurring between the user and the SRF via the bearer channel established at the Connected state. This state is entered as a result of event E2. It is exited as a result of events E10, e11 and e12. Events E5, E6, e7, e8 and e9 do not cause a state change. Event E5 also represents additional PlayAnnouncement/PromptAndCollectUser Information/PromptAndReceiveMessage operations which are buffered as discussed in the procedures.

(E2) and (E5) Instruction_from_SCF
This event takes place when an initial or subsequent PlayAnnouncement, PromptAndCollectUserInformation, or PromptAndReceiveMessage operation(s) from the SCF is received. The SRSM goes to state "User interaction" on the first (E2). The SRSM remains in state "User interaction" for subsequent (E5)s.

(E6) Cancel_from_SCF (for PlayAnnouncement/PromptAndCollectUserInformation /PromptAndReceiveMessage)
This event takes place when the corresponding PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage operation is received from the SCF. The indicated interaction is terminated if it is presently running, otherwise it is deleted from the buffer. The SRSM remains in state F"User interaction".

(e7) SRF_Report_to_SCF
This event takes place when a SpecializedResourceReport ,a return result for PromptAndCollectUserInformation or a return result for PromptAndReceiveMessage operation is sent to the SCF. The SRSM remains in state "User interaction".

(e8) PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message_Cancelled_to_SCF
This event takes place when the PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage error caused by the Cancel (for PlayAnnouncement,PromptAndCollectUserInformation or PromptAndReceiveMessage) operation is sent to the SCF. This event represents the successful cancellation of an active or buffered PlayAnnouncement/PromptAndCollect UserInformation/PromptAndReceiveMessage operation. The SRSM remains in state "User interaction".

(e9) Cancel_Error_to_SCF
This event takes place when the cancel error (for PlayAnnouncement ,-PromptAndCollectUserInformation or PromptAndReceiveMessage) is sent to the SCF. This event represents the unsuccessful cancellation of a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation. The SRSM remains in state "User interaction".

(E10) Connection_Released_from_SSF
This event takes place when the SRSM receives a release message from the SSF. The SRSM goes to state "Idle".

(e11) Disconnect_to_SSF
This event occurs when the SCF has enabled SRF initiated disconnect by:

- the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive
Message from SCF (E2) or (E5). The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the last SpecializedResourceReport operation, a return result for PromptAndCollectUser
Information or a return result for PromptAndReceiveMessage operation to the SCF. The SRSM goes to state "Idle"; or

- a parameter in the ScriptRun operation.

(e12) SRF_Sanity_Timeout
This event occurs when the SRSM has been in user interaction state for a network-operator-defined-period of time (timer T_{SRF}) without having a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

(E14) ScriptClose_from_SCF
This event takes place when the ScriptClose operation from the SCF is received. The SRSM goes to state "Connected".

Note that this state transition is permitted only if the SRSM received ScriptRun operation from SCF previously.

(e15)ScriptEvent_to_SCF
This event takes place when a partial result is sent from a SRF to the SCF in case the SRF needs additional information or when the final Result of the User Interaction script execution is sent from the SRF to the SCF.The SRSM remains in the state "User Interaction".

(E16) ScriptInformation_from_SCF
This event takes place when the ScriptInformation operation from the SCF is received. The SRSM remains in the state "User Interaction".

(E17) ScriptRun_from_SCF
This event takes place when the ScriptRun operation from the SCF is received. The SRSM remains in state "User Interaction". It follows a PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReceiveMessage which is terminated. No other User Interaction Script shall be already active for the call.

Note that a subsequent ScriptRun operation from the SCF is not permitted in this state.

In addition to these explicitly marked transitions, failure of a user-SRF bearer connection will cause the SRSM to transit to Idle from any state. These transitions are not shown on figure 13-3 for the purpose of visual clarity.

14.5 Example SRF control procedures

This subclause provides a detailed description of the SRF procedures. Arrow diagrams are used for the description of the connect, interaction with the end user, and disconnect stages.

The SRF control procedures are based on various physical allocation patterns of SRF. The various control procedures are described in this subclause in accordance with the example physical scenarios of protocol architecture in subclause 3.1.

The service assist and hand-off procedures based on the physical scenarios are also described in this subclause as examples.

Note that, throughout this subclause, bearer connection control signalling messages are used for explanatory purpose, and are not subject for standardization in the present document. The terms used for bearer connection control signalling messages only represent the functional meaning.

14.5.1 SRF connect procedures

14.5.1.1 SRF connect physical procedures

Several procedures are required for different physical scenarios. The cases to be covered are described below and illustrated in figure 13-4.

i) The IP is integrated into the SSP, or attached to the SSP, possibly via a local exchange, that is interacting with the SCP but the SCP's operations to the IP are relayed via the SSP which performs any needed protocol conversion;

ii) the IP is directly attached to the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved;

iii) the IP is integrated into another SSP, or directly attached to another SSP, than the one that is interacting with the SCP but the SCP's operations to the IP are relayed via the second SSP (called the "Assist" method), and on completion of the user interaction, control is returned to the first SSP;

iv) the IP is directly attached to a node other than the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved (called the "Assist" method, but with a variation on the physical connectivity of the entities involved), and on completion of the user interaction, control is returned to the first SSP; and

v) the IP is attached to another SSP and on completion of the user interaction, control of the call is retained at that SSP (called the "Hand-off" approach).

In each of the above cases, the operations between the SCP and the SSP may be SS7 TCAP-based; the messaging between the SSP and the IP when the SSP does relaying may be DSS1 using the facility IE (in this case, the SSP would have to do protocol conversion from SS7 TCAP to DSS1 facility IE for the operations and responses it relayed between the SCP and the IP); the direct messaging between the SCP and the IP may be SS7 TCAP based; and bearer control signalling may be any system.

Each of the scenarios will now be examined using arrow diagrams.

Case i) is illustrated in figure 13-5. Note that for the integrated IP/SSP, the internal activities of the node can still be modelled in this way, but the details of how this is achieved are left to the implementor. This approach makes it unnecessary for the SCP to distinguish between integrated and external but directly connected IPs. See also a note on the possibility of concatenating the first user interaction operation with the ConnectToResource operation discussed in the subclause on user interaction below. The establishment of the SCF-SRF relationship in this case is implicit.

Case ii) requires that the IP indicate to the SCP that it is ready to receive operations (see figure 13-6). The establishment of the SCF-SRF relationship is explicit. Note that it is necessary to convey a correlation ID to ensure that the transaction established between the SCP and the IP can be correlated to the bearer connection setup as a result of the SCP's preceding operation to the SSP.

Case iii) requires that a transaction be opened with the assisting SSP so that it may relay operations from the SCP to the IP (integrated or external). Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP. After the AssistRequestInstructions are received by the SCP, the procedures are the same as case i). Figure 13-7 illustrates the preamble involved.

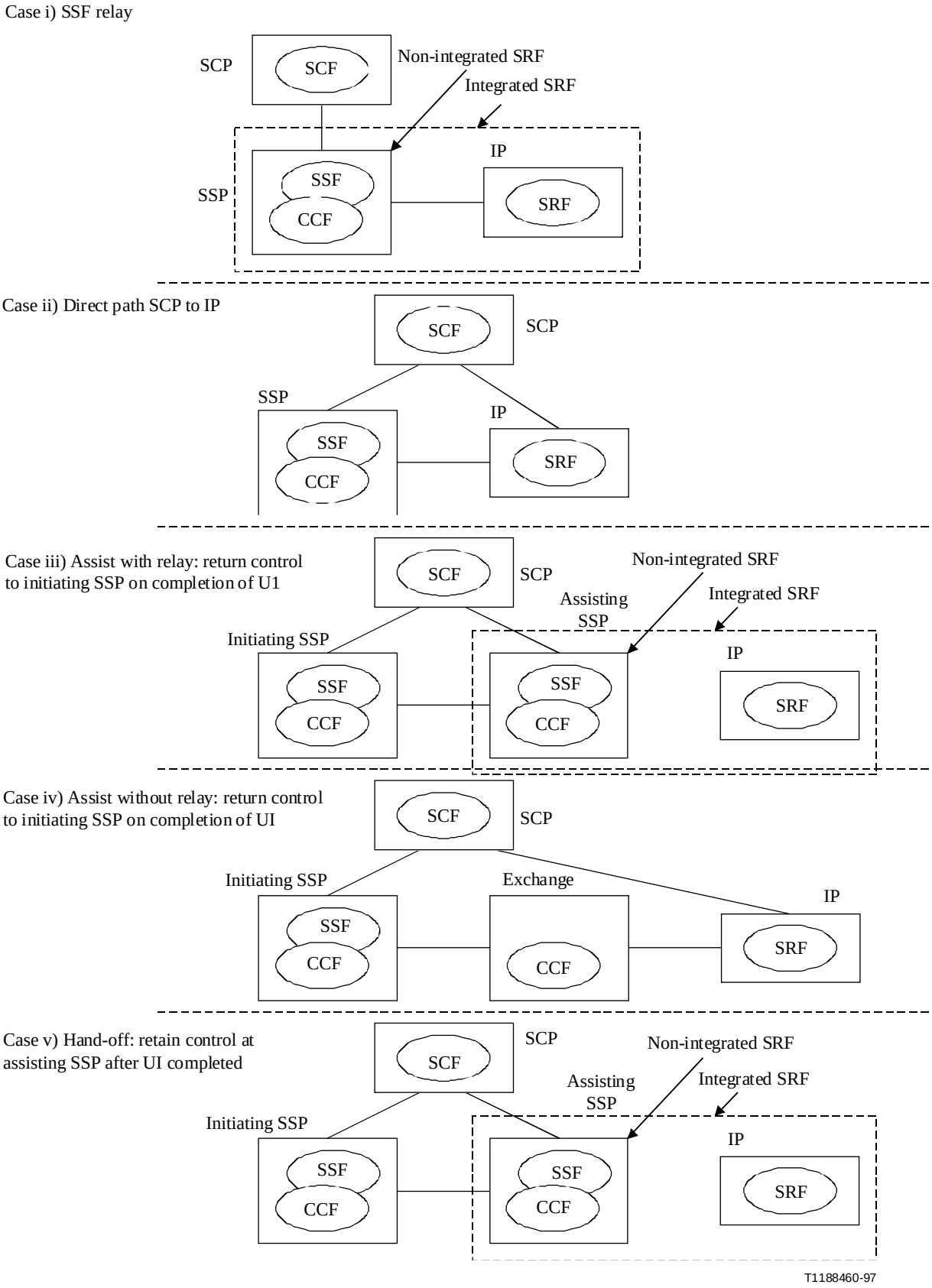

Case i) SSF relay
SCP
SCF
Non-integrated SRF
Integrated SRF
IP
SSP
SSF
CCF
SRF
Case ii) Direct path SCP to IP
SCF
SCP
SSP
IP
SSF
CCF
SRF
Case iii) Assist with relay: return control to initiating SSP on completion of U1
SCF
SCP
Non-integrated SRF
Assisting SSP
Integrated SRF
Initiating SSP
IP
SSF
CCF
SSF
CCF
SRF
Case iv) Assist without relay: return control to initiating SSP on completion of UI
SCF
SCP
Initiating SSP
Exchange
IP
SSF
CCF
CCF
SRF
Case v) Hand-off: retain control at assisting SSP after UI completed
SCF
SCP
Non-integrated SRF
Assisting SSP
Integrated SRF
Initiating SSP
IP
SSF
CCF
SSF
CCF
SRF
T1188460-97

Figure 13-4

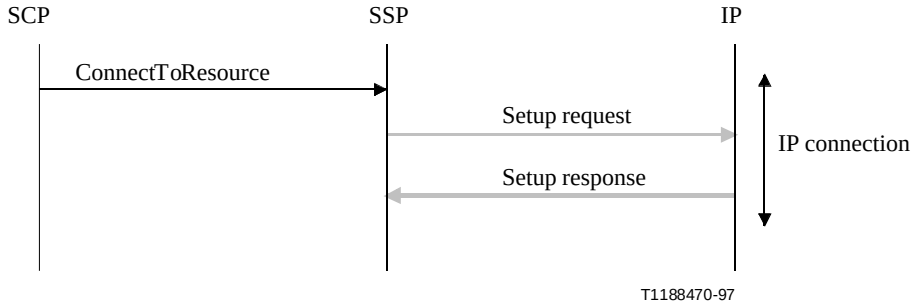
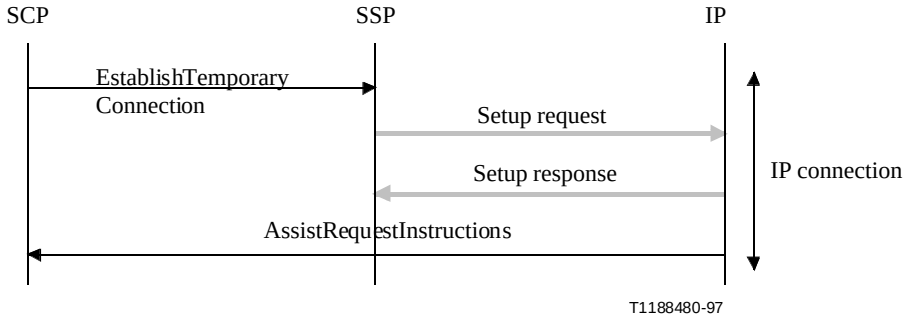


Figure 13-5: Connection to integrated or external IP with SSP relay of IP operations



NOTE: Black lines indicate INAP operations, grey lines indicate DSS1 operations.

Figure 13-6: Connection to IP with direct link to SCP, IP initiates interaction with SCP

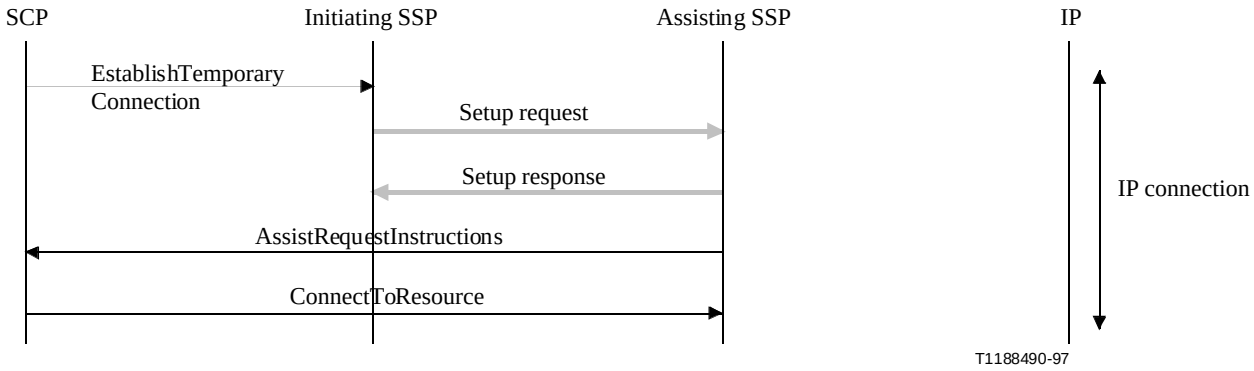


Figure 13-7: Preamble for assist case with integrated IP or external IP and SSP relay of SCP-IP messages

Case iv) does not require the establishment of a second transaction from the assisting exchange, hence it need not be an SSP. This then becomes a preamble to the procedure shown in figure 13-6 as shown in figure 13-8.

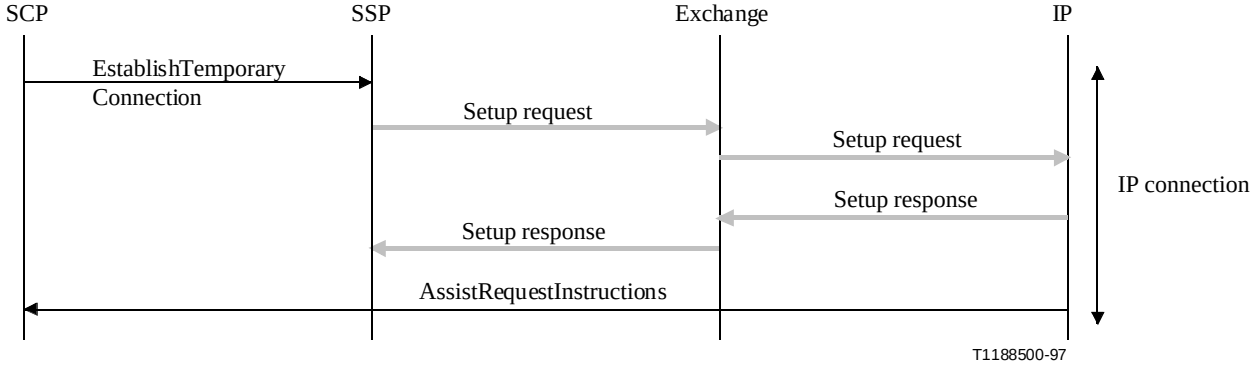


Figure 13-8: Preamble for assist case with external IP and direct SCP-IP messaging

Case v) merely requires the sending of an operation to the first SSP to route the call to the handed-off SSP, and then Figure 13-5 applies at handed-off SSP. This is shown in Figure 13-9. Note that the activity at handed-off SSP represents a new interaction with the SCP and "AssistRequestInstructions" is used. Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP.

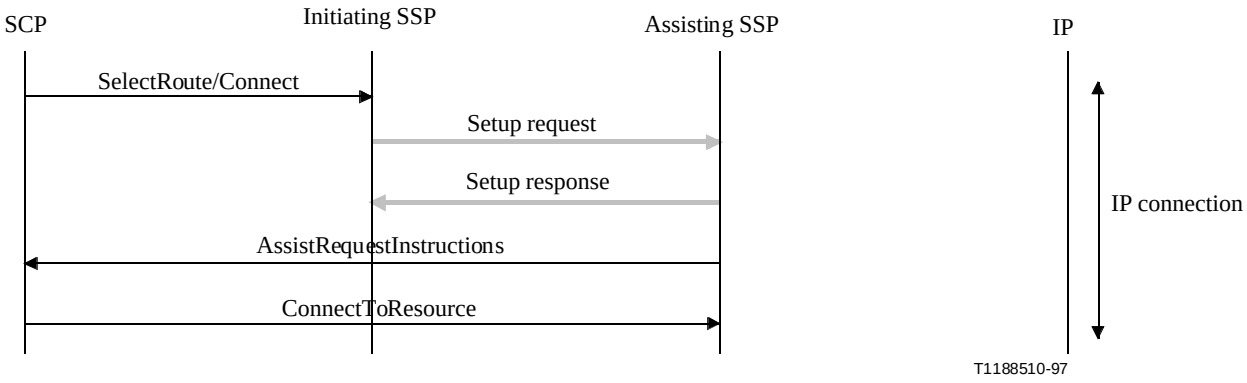


Figure 13-9: Preamble for hand-off case

14.5.2 SRF end user interaction procedures

The end user interaction procedures allow:

- the sending of one or multiple messages to the end user by using the PlayAnnouncement operations;
- a dialogue with the end user by using one or a sequence of PromptAndCollectUser Information operations;
- a dialogue with the end user by using one or a sequence of PromptAndReceiveMessage operations;
- a combination of the above; and
- cancellation of a PlayAnnouncement, PromptAndCollectUserInformation or PromptAndReciveMessage operations by using a generic cancel operation.

14.5.2.1 **PA**/prompt & collect user information/prompt & receive message (PA/P&C/P&R)

There are only two physical scenarios for user interaction:

i) the SSP relays the operations from the SCP to the IP and the responses from the IP to the SCP (SSF relay case); and

ii) The operations from the SCP to the IP and the responses from the IP are sent directly between the SCP and the IP without involving the SSP (direct SCF-SRF case).

Case i) is illustrated in figure 13-10.

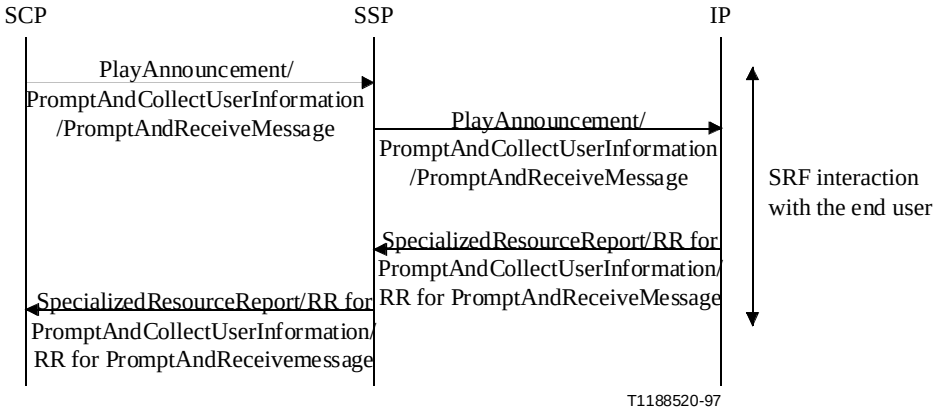


Figure 13-10: SSP relay of user interaction operations and responses

Case ii) is illustrated in figure 13-11.

It is also necessary to consider the capability of SS7 TCAP to concatenate several Invoke PDUs in one message. This capability allows, for the scenario in figure 13-5, the ConnectToResource and the first PlayAnnouncement/PromptAndCollectUserInformation/Prompt AndReceiveMessage to be carried in one message. This has some advantages in this physical scenario, such as reduced numbers of messages, and possibly better end-user perceived performance.

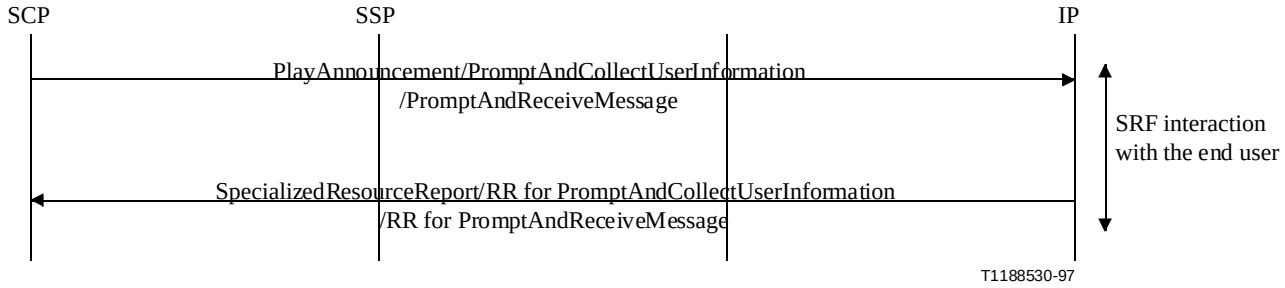


Figure 13-11: Direct SCF-SRF of user interaction operations and responses

14.5.3 SRF disconnection procedures

The disconnection procedures are controlled by the SCF and the procedure used is selected based on the needs of the service being executed. The bearer disconnection procedure selected by the SCF is to either allow the SRF to disconnect on completion of user interaction, or to have the SCF explicitly order the SSF to disconnect.

SRF disconnect does not cause disconnection by the SSF/CCF back to the end user terminal unless the transaction with the SCF has been terminated, indicating the user interaction completed the call. The SSF/CCF recognizes that a connection to an SRF is involved because the operations from the SCF for this purpose are distinct from the operations that would be used to route the call towards a destination. There is no impact on bearer signalling state machines as a result of this since incoming and outgoing bearer signalling events are not simply transferred to each other, but rather are absorbed in call processing, and regenerated as needed by call processing. Therefore, to achieve the desired functionality, call processing need simply choose not to regenerate the disconnect in the backward direction. Figure 13-12 illustrates this concept.

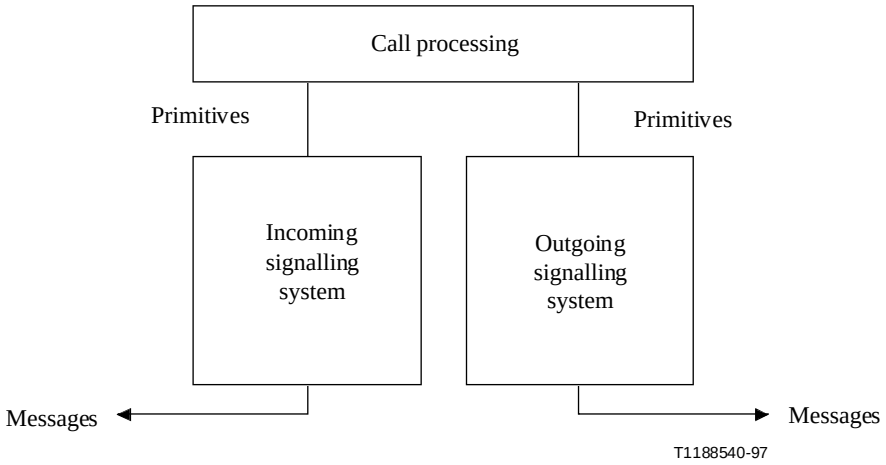


Figure 13-12: Relationship of incoming and outgoing signalling systems to call processing

As for the SRF connection procedures, the SRF disconnection is affected by the physical network configuration.

In order to simplify the interface between the SCF and the SRF, a number of assumptions are made. The assumptions, and the resulting rules, result in unambiguous procedures from both the SCF and the SRF points of view. The rules, presented below, refer to the SRF originated disconnect, or "SRF Initiated Disconnect", and to the SCF originated disconnect, or "SCF Initiated Disconnect". While other scenarios are possible, they are not included because they either duplicate the functionality presented below or they otherwise do not add value from a service perspective.

1 If a series of PlayAnnouncement/PromptAndCollectUserInformation/PromptAnd ReceiveMessage operations are to be executed by the same SRF, then SRF disconnect is inhibited for all but the last and may be inhibited on the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage. When a subsequent PlayAnnouncement/PromptAndCollectUserInformation/PromptAnd ReceiveMessage is received, it is buffered until the completion of any preceding PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage.

2 A generic cancel operation terminates the indicated PlayAnnouncement/PromptAnd CollectUserInformation/PromptAndReceiveMessage if it is being executed by the SRF, but does not disconnect the SRF. If the cancel operation is for a buffered PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage, that PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage is discarded, but the current and any buffered PlayAnnouncement/PromptAndCollectUserInformations/ PromptAndReceiveMessage are executed. An SRF interacts with one user only and therefore cancelling a PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage only affects the user to which the SRF is connected.

3 The SCF must either explicitly order "Disconnect" or enable SRF initiated disconnect at the end of the PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message. An SRF left connected without a PlayAnnouncement/PromptAndCollectUser Information/PromptAndReceiveMessage to execute may autonomously disconnect if it has not received any PlayAnnouncement/PromptAndCollectUserInformation/PromptAnd ReceiveMessage operations within a defined time limit. This could occur, for example, after an EstablishTemporaryConnection which is not followed within a reasonable time period with a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation. This sanity timing value will depend on the nature of the interaction the SRF supports and should be selected by the network operator accordingly.

4 When SRF initiated disconnect is enabled in a PlayAnnouncement/PromptAndCollectUser Information/PromptAndReceiveMessage, then the SRF must disconnect on completion of the user interaction.

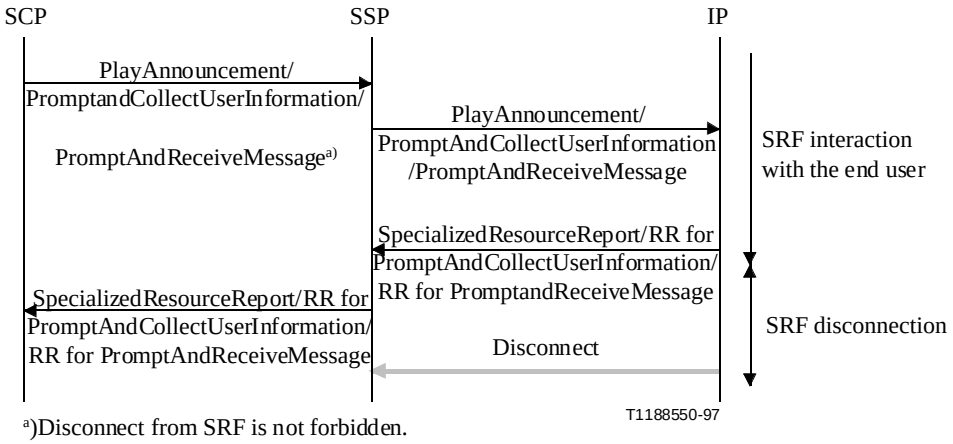
5 When SRF initiated disconnect is not enabled, the SCF must ask the SRF to inform it of the completion of the user interaction using the SpecializedResourceReport operation for "announcement complete", using the return result for the PromptAndCollectUser Information operation or using the return result for the PromptAndReceiveMessage operation.

6 If the user disconnects, the SRF is disconnected and the SSF releases resources and handles the transaction between the SSF and the SCF as specified in ITU-T Recommendation Q.1224 [26]and in the present document. The SRF discards any buffered operations and returns its resources to idle. The relationship with the SCF is terminated.

7 When the SCF explicitly orders the SSF to disconnect by "DisconnectForwardConnection" operation, the SSF releases the bearer connection to the SRF, and returns to the "waiting for instructions" state. No operation reporting SRF disconnect from the SSF to the SCF is required.

14.5.3.1 SRF initiated disconnect

The SRF disconnect procedure is illustrated in figure 13-13. The SRF disconnect is enabled by the SCF within a PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage operation. When the SRF receives a PlayAnnouncement/PromptAndCollectUserInformation/ PromptAndReceiveMessage enabling disconnection, it completes the dialogue as instructed by the PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The SSF/CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The SSF returns to the "waiting for instructions" state and executes any buffered operations. In the hand-off case, the SSP shown in Figure 13-13 is the "handed-off" SSP.



[a)]Disconnect from SRF is not forbidden.

Figure 13-13: SCF disconnect for local, embedded and hand-off scenarios

For the assisting SSF case, the SRF initiated disconnect procedures are not used because the assisting SSF remains in the "waiting for instructions" state and does not propagate the disconnection of the bearer connection to the Initiating SSF. The SCF initiated disconnect procedures described in the following subclause are used for the assisting SSF case.

For the direct SCF-SRF case, the procedures also work in the same manner. The SRF disconnect is enabled by the SCF within a PlayAnnouncement/PromptAndCollectUserInformation/Prompt AndReceiveMessage operation. When the SRF receives a PlayAnnouncement/PromptAndCollect UserInformation/PromptAndReceiveMessage enabling disconnection, it completes the dialogue as instructed by the PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The Initiating SSF/CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The Initiating SSF returns to the "waiting for instructions" state and executes any buffered operations.

14.5.3.2 SCF initiated disconnect

The SCF initiated disconnect procedure is illustrated in figure 13-14. Bearer messages are shown in gray. The figure shows only the assisting SSF case, and the direct SCF-SRF case is not shown. To initiate the SCF initiated disconnection of the SRF, the SCF must request and receive a reply to the last PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceive Message operation requested. The SpecializedResourceReport operation contains an "announcement complete" and return result for PromptAndCollectUserInformation contains "collected information."

The SCF initiated disconnect uses an operation called DisconnectForwardConnection. Once the DisconnectForwardConnection is received by the SSF, it will initiate a "release of bearer channel connection" between the PE containing the SSF and SRF, using applicable bearer control signalling. Since the SCF (which initiates the disconnect), the SSF (which instructs bearer signalling to disconnect) and the SRF (which receives disconnect notification via bearer signalling) are aware that disconnect is occurring, they are synchronized. Therefore, a "pre-arranged" end may be used to close the transaction. This does not preclude the use of explicit end messages for this purpose.

For assisting SSF case, the initiating SSP, on receipt of the DisconnectForwardConnection from the SCP, disconnects forward to the assisting SSP, and this disconnection is propagated to the IP. The initiating SSP, knowing that the forward connection was initiated as the result of an EstablishTemporaryConnection, does not disconnect back to the user but returns to the "waiting for instructions" state.

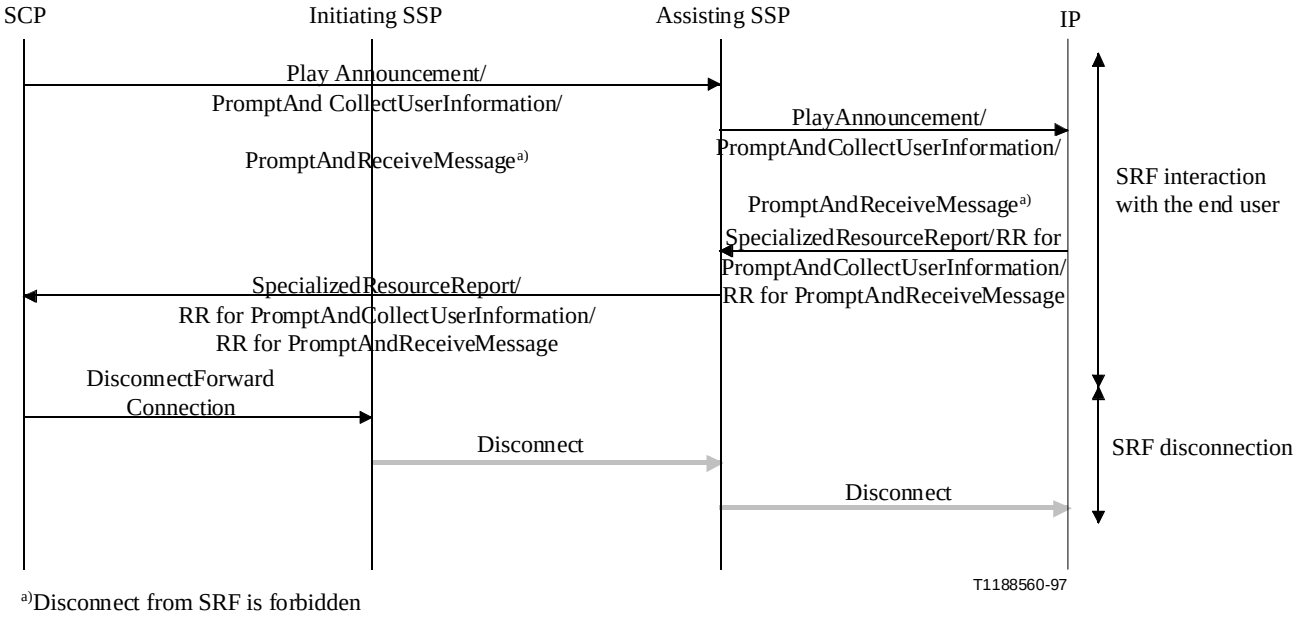


Figure 13-14: SCF initiated disconnect for assist scenario

14.5.4 Examples illustrating complete user interaction sequences

The following figures and their accompanying tables provide examples of complete sequences of user interaction operations covering the three stages:

- Connect the SRF and the end user (bearer connection) and establish the SCF-SRF relationship.
- Interact with the end user.
- Disconnect the SRF and the end user (bearer connection) and terminate the SCF-SRF relationship.

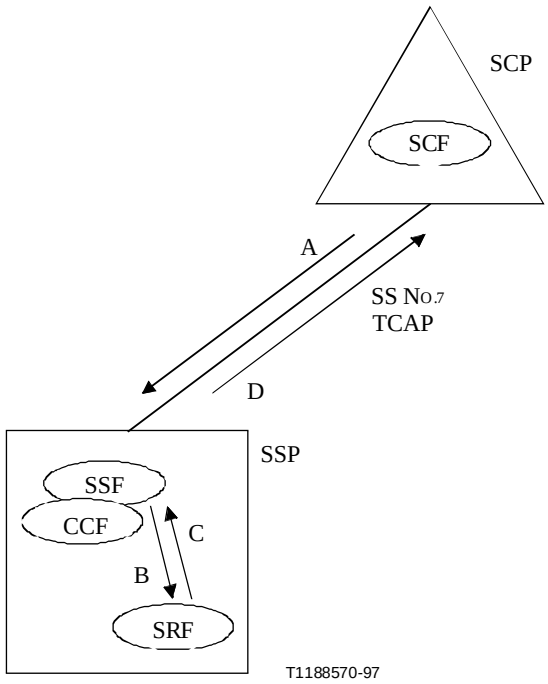


Figure 13-15: SSP with integrated SRF

In figure 13-15, the SSP with an integrated (or embedded) SRF, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA /P&C/P&R	ConnectToResource; PlayAnnouncement/ PromptAndCollectUserInformation /PromptAndReceiveMesssage Setup; PlayAnnouncement/ PromptAndCollectUserInformation/Prompt AndReceiveMessage	A B
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/Prompt AndReceiveMessage SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage	A then B C then D
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage Disconnect	C then D C (intra-SSP bearer control)
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage DisconnectForwardConnection Disconnect	C then D A B (intra-SSP bearer control)

A simple extension to this integrated case is the configuration where the SRF is located in an Intelligent Peripheral (IP) locally attached to the SSP. The SCP-IP operations are relayed via the SSF in the SSP. This is depicted in figure 13-16.

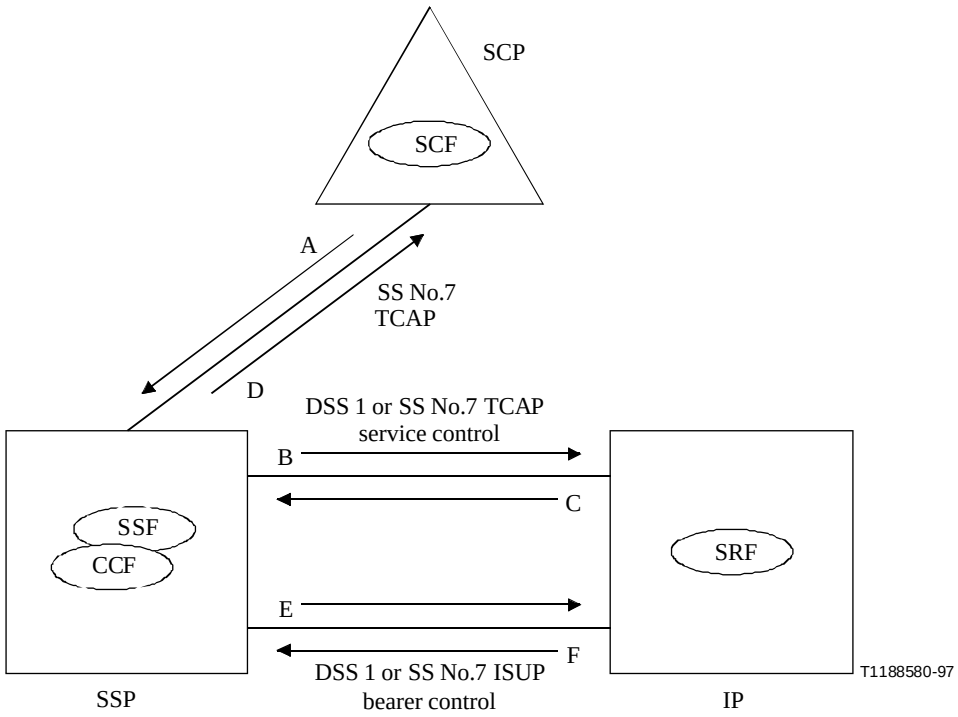


Figure 13-16: SSP relays messages between SCP and IP

The procedural scenarios for this relay SSF with an IP (Figure 13-16) can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA/P&C/P&R	ConnectToResource; PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage *If DSS 1 used:* Setup; PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage *If SS7 used:* IAM PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage	A E and B (Facility IE) E B
User interaction	PlayAnnouncement/PromptAndCollectUserInformation/PromptAndReceiveMessage SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage	A then B C then D
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage *If DSS 1 used:* Disconnect *If SS7 used:* Release	C then D F F
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS7 used:* Release	C then D A E E

In some cases, the IP may have an SS7 or other interface to the controlling SCP. This case is shown in figure 13-17. Note that the SCP must correlate two transactions to coordinate the activities.

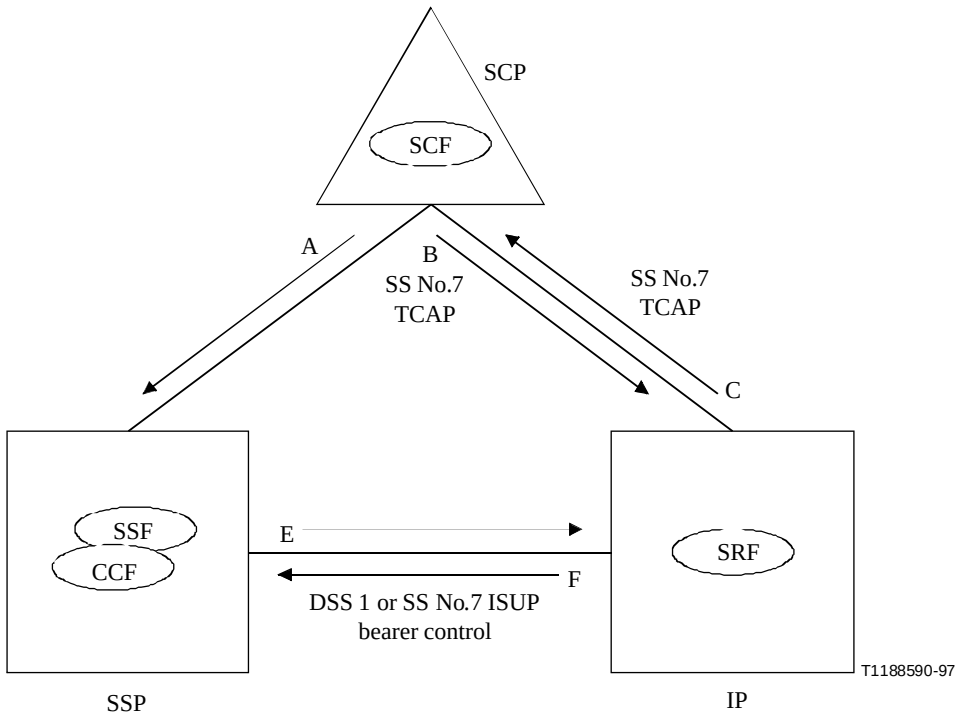


Figure 13-17: Direct SCP-IP information transfer

In figure 13-17, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource	EstablishTemporaryConnection *If DSS 1 used:* Setup AssistRequestInstructions *If SS7 used:* IAM AssistRequestInstructions	A E C E C
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage	B C
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage *If DSS 1 used:* Disconnect *If SS7 used:* Release	C F F
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS7 used:* Release	C A E E

The assisting SSF scenario involves straightforward procedural extensions to the basic cases shown above. One mapping of the assisting SSF case is shown in figure 13-18. In this case, SRF initiated disconnect cannot be used. Other physical mappings can be derived as described in the text following the figure and its accompanying table.

Note that the integrated SRF and SSF relay case requires a transaction between the SCP and the assisting SSP (Figure 13-18) but the SCP direct case does not since the transaction is directly between the SCP and the IP connected to the remote exchange. In the latter case, any transit exchanges, including the one the IP (SRF) is connected to, are transparent to the procedures.

Note also that the SCP must again correlate two transactions.

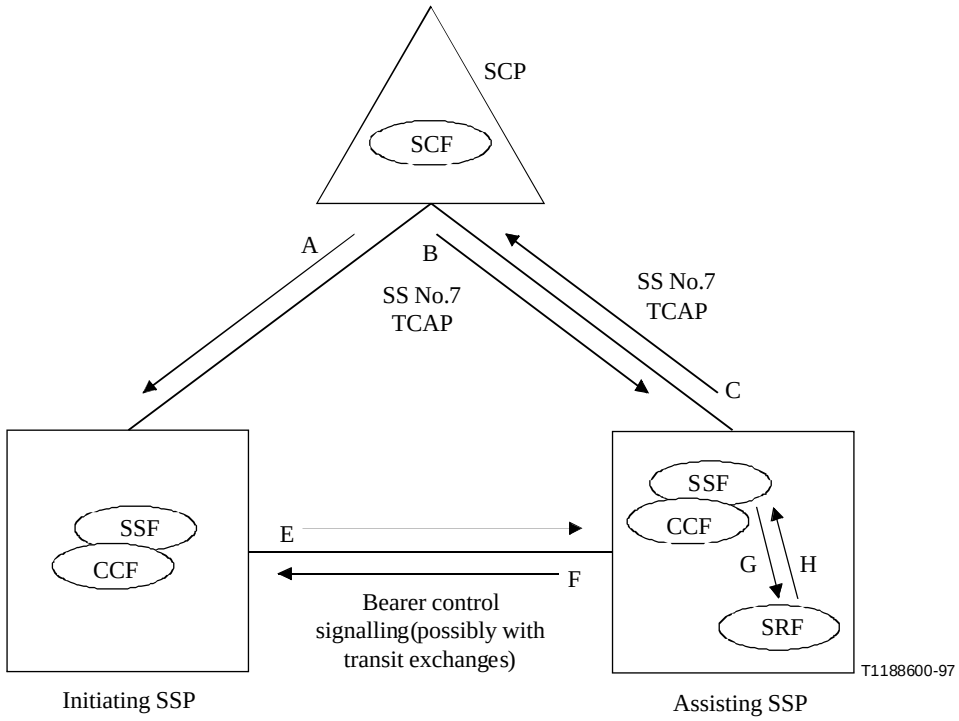


Figure 13-18: SSP assist (relay SSP)

In figure 13-18, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Assist preamble	EstablishTemporaryConnection	A
	If DSS 1 used:	
	Setup	E
	AssistRequestInstructions	C
	ConnectToResource	B
	Setup	G
	ResetTimer	A
	If SS7 used:	
	IAM AssistRequestInstructions	E
	ConnectToResource	C
	Setup	B
	ResetTimer	G
		A
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation/PromptAndReceiveMessage	B then G
	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage	H then C
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation/RR for PromptAndReceiveMessage	H then C
	DisconnectForwardConnection	A
	If DSS 1 used:	
	Disconnect	E and G (intra-SSP bearer ctrl)
	If SS7 used:	
	Release	E and G (intra-SSP bearer ctrl)

Note that the assisting SSP case shown in figure 13-18 can be generalized to cover both the case where the SRF is embedded in assisting SSP (as shown), and the case where the SRF is locally connected to assisting SSP. In this latter case, the SRF communication (protocol flows B, C, G and H) would conform to the physical scenario shown in figure 13-16.

The service hand-off scenario can similarly be viewed as a sequence consisting of an IN service to route a call from one SSP to another, followed by any one of the previously described physical user interaction scenarios. For describing this scenario ITU-T Recommendation Q.228 [] figure 13-18 can be used also.

14.5.4.1 Message sequences for service assist

The following subclause provides additional details on the message sequences for the service assist procedure in figure 13-18:

1 The SCP, during the processing of a request for instruction, determines that resources remote from the initiating SSP are required and that call processing will continue from the initiating SSP after the remote resources have been used (e.g., the call will be completed to a destination address after information is collected from the calling party). An EstablishTemporaryConnection operation containing the address of the assisting SSP (for routeing the call), the ScfID and the CorrelationID (both used for the assisting SSP to establish communication back to the SCP) is sent to the initiating SSP. The EstablishTemporaryConnection is used instead of a regular Connect operation because of the nature of the connection to the assisting SSP. The initiating SSP must be aware that the SCP will ask it to continue in the processing of the call at some point in the future.

NOTE 1: The ScfID and CorrelationID may be included in the routeing address of the assisting SSP.

Protocol Flow A

2 The initiating SSP routes the call to the assisting SSP. The ScfID and CorrelationID are sent to the assisting SSP. Existing in-band signalling and SS7 information elements (e.g. routeing number) could be used to transport this information. The transport mechanism used to send this information between SSPs is independent of the service assist control procedures between the SCF and SSF.

Protocol Flow E

3 The assisting SSP uses an AssistRequestInstructions operation to establish communication with the SCP. The CorrelationID is sent in the AssistRequestInstructions to allow the SCP to correlate two transactions.

Protocol Flow C

4 The SCP sends instructions to the assisting SSP based on SL control.

Protocol Flow B

5 The SCP may need to generate reset timer events to the initiating SSP so that it does not time out the call.

Protocol Flow A

NOTE 2: The usage of ResetTimer operation is optional.

6 When resource functions have been completed, a DisconnectForwardConnection operation is sent to the initiating SSP. This indicates, that the temporary connection to the assisting SSP has to be disconnected.

Protocol Flow A

NOTE 3: A DisconnectForwardConnection operation followed by a ConnectToResource may be sent to the assisting SSP to access several resources in the assisting case.

7 The initiating SSP sends a message via bearer control signalling to the assisting SSP to close the "assist" transaction.

Protocol Flow E

8 The call control returns to the initiating SSP.

14.5.4.2 Message sequences for hand-off

The following subclause outlines message sequences for the hand-off procedure using the protocol flows shown in figure 13-18:

1 The SCP, during the processing of a request for instruction, determines that resources remote from the initiating SSP are required and that call processing need not continue from the initiating SSP after the remote resources have been used (e.g., a terminating announcement will be played). A Connect operation containing the address of the assisting SSP (for routeing the call), the ScfID and the CorrelationID (both used for the assisting SSP to establish communication back to the SCP) is sent to the initiating SSP.

NOTE: The ScfID and CorrelationID may be included in the routeing address of the assisting SSP.

Protocol Flow A

2 The initiating SSP routes the call to the assisting SSP. The ScfID and CorrelationID are sent to the assisting SSP. Existing in-band signalling and SS7 information elements (e.g. routeing number) could be used to transport this information. The transport mechanism used to send this information between SSPs is independent of the service assist control procedures between the SCF and SSF.

Protocol Flow E

3 The assisting SSP uses an AssistRequestInstructions operation to establish communication with the SCP. The CorrelationID is sent in the AssistRequestInstructions to allow the SCP to correlate two transactions. The AssistRequestInstructions is used instead of a regular request instruction (InitialDP or DP-specific operation) because the SCP must associate the AssistRequestInstructions from the assisting SSP/IP with an already active dialogue the SCP has with another SSP.

Protocol Flow C

4 The SCP sends instructions to the assisting SSP based on SL control.

Protocol Flow B

5 The call control remains at the assisting SSP.

The same service assist and hand-off procedures can be reused for a direct link to an IP in this and future capability sets.

15 SDF AE procedures

15.1 General

This clause provides the definition of the SDF AE procedures related to the SDF-SCF and SDF-SDF interfaces. The procedures are based on the use of SS7.

Other capabilities may be supported in an implementation-dependent manner in the SCP, SDP, SSP, AD or SN.

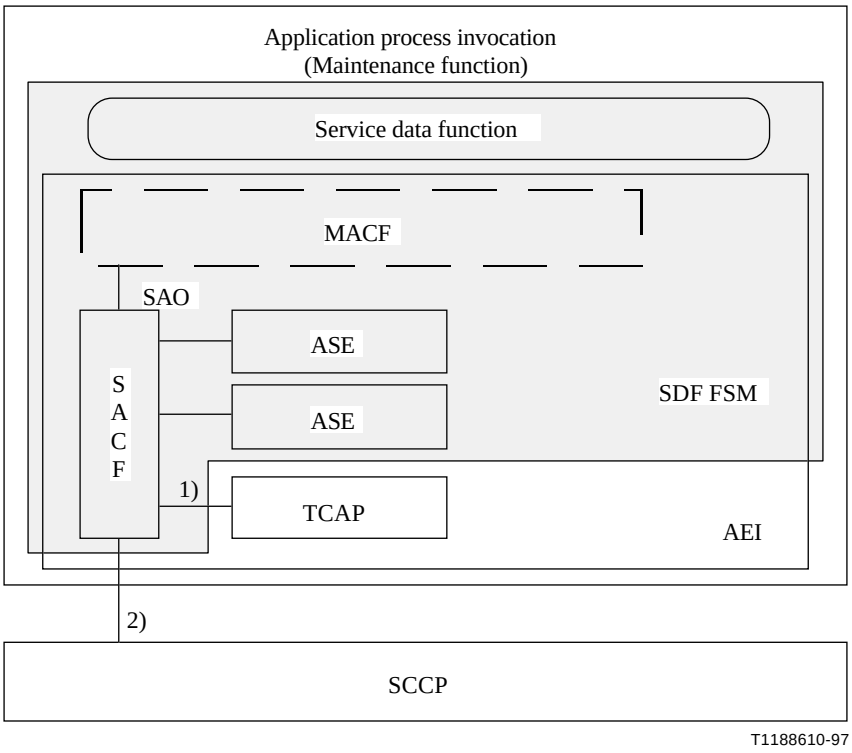
The AE, following the architecture defined in ITU-T Recommendations Q.700 [19], Q.1400 [30], and in ETS 300 287-1 [7] includes TCAP (transaction capabilities application part) and one or more ASEs called TC-users, which are based on the Directory (X.500-series of ITU-T Recommendations [36]). The following subclauses define the TC-user ASE and SACF and MACF rules, which interface with TCAP using the primitives specified in ETS 300 287-1 [7].

The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the AE procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed clause 17 shall be followed.

15.2 Model and interfaces

The functional model of the AE-SDF is shown in figure 14-1; the ASEs interface to TCAP to communicate with the SCF and other SDFs, and interface to the maintenance functions. The scope of this ITU-T Recommendation is limited to the shaded area in figure 14-1.



AEI Application entity invocation
SDF Service data function
FSM Finite state machine
SACF Single association control function
SAO Single association object
MACF Multiple association control function

1) TC-primitives.
2) N-primitives.

NOTE-The SDF FSM includes several finite state machines.

Figure 14-1: Functional model of SDF

The interfaces shown in figure 14-1 use the TC-user ASE primitives specified in ETS 300 287-1 [7] (interface 1) and N-Primitives specified in ETS 300 009-1 [3] (interface 2). The operations and parameters of INAP are defined in clauses 4 to 10 of the present document.

An instance of a specific SDF FSM may be created if IN call or call unassociated handling is received from the SCF, an SDF invokes or responds to a chaining operation to/from another SDF, or an SDF invokes or responds to a shadowing operation to/from another SDF.

The SDF FSM handles the interaction with the SCF FSM and another SDF FSM.

15.3 The SDF FSM structure

The structure of the SDF FSMs is illustrated in figure 14-2.

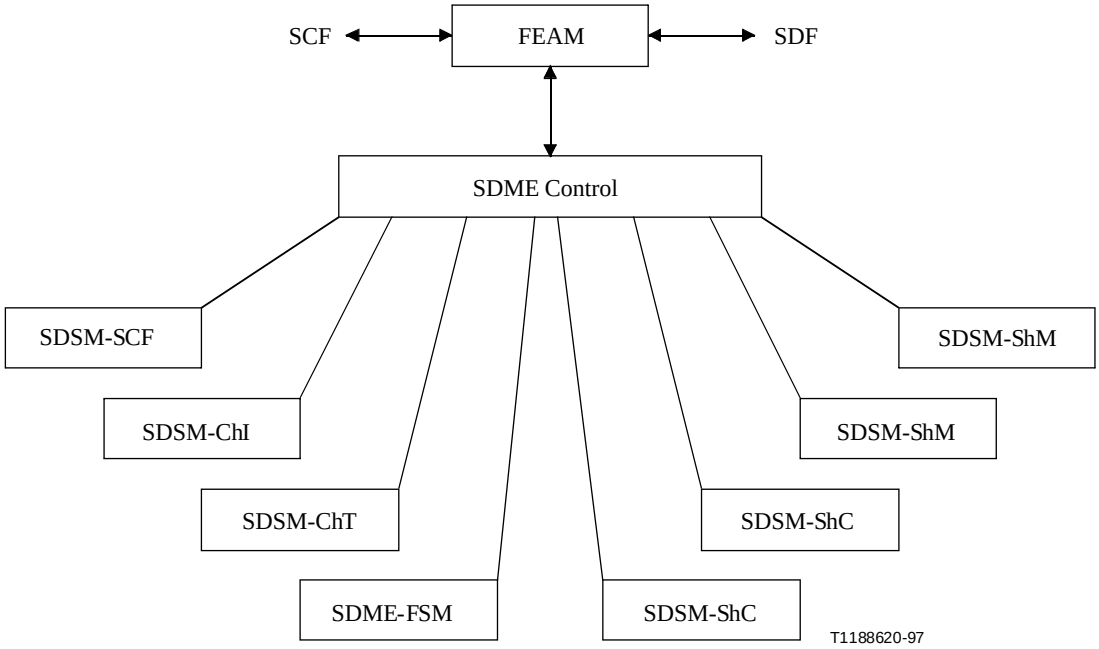


Figure 14-2: SDF Interfaces

The SDF FSM (SDSM-SCF) handles the interaction with the SCF FSM. The SDSM-ChI and SDSM-ChT handle the interactions between SDFs for chaining initiation and termination. The SDSM-ShSSi and SDSM-ShCSi handle the interactions between SDFs for shadowing supplier and consumer initiated by supplier. The SDSM-ShCSi and SDSM-ShCCi handle the same as above initiated by consumer. The SDME-FSM handles the interaction between the SDF and the SDF management functions.

The management functions related to the execution of operations received from the SCF or cooperating SDF are executed by the SDF Management Entity (SDME). The SDME is comprised of an SDME control and several instances of SDME FSMs. The SDME control interfaces the different SDF FSMs (e.g. SDSM-SCF) and SDME-FSMs respectively as well as the FEAM.

The FEAM provides the low level interface maintenance functions including the following:

1 establishing and maintaining interfaces to the SCF and cooperating SDFs;

2 passing and queuing (when necessary) the messages received from the SCF and cooperating SDF to the SDME Control;

3 formatting, queuing (when necessary), and sending the messages received from the SDME Control to the SCF and cooperating SDF.

The SDME Control maintains the associations with the SCF and cooperating SDFs on behalf of all instances of the SDF FSMs (e.g. SDSM-SCF, SDSM-ChI). These instances of the SDF FSMs occur concurrently and asynchronously as SDF related events occur, which explains the need for a single entity that performs the task of creation, invocation and maintenance of the SDF FSMs. In particular, the SDME Control performs the following tasks:

1 interprets the input messages from other FEs and translates them into corresponding SDF FSM events;

2 translates the SDF FSM outputs into corresponding messages to other FEs;

3 captures asynchronous activities related to management or supervisory functions in the SDF and creates an instance of an SDME-FSM. For example, management invocation of a shadowing procedure between network operators. In this case, the SDME Control will create an instance of the SDME-FSM to handle this management related operation.

15.4 SDF state transition models

15.4.1 SDF state transition model for SCF related states

The SDF's job relating to interactions with the SCF is to (synchronously) respond to every request from the SCF after the Bind procedure. The respective FSM is depicted in figure 14-3.

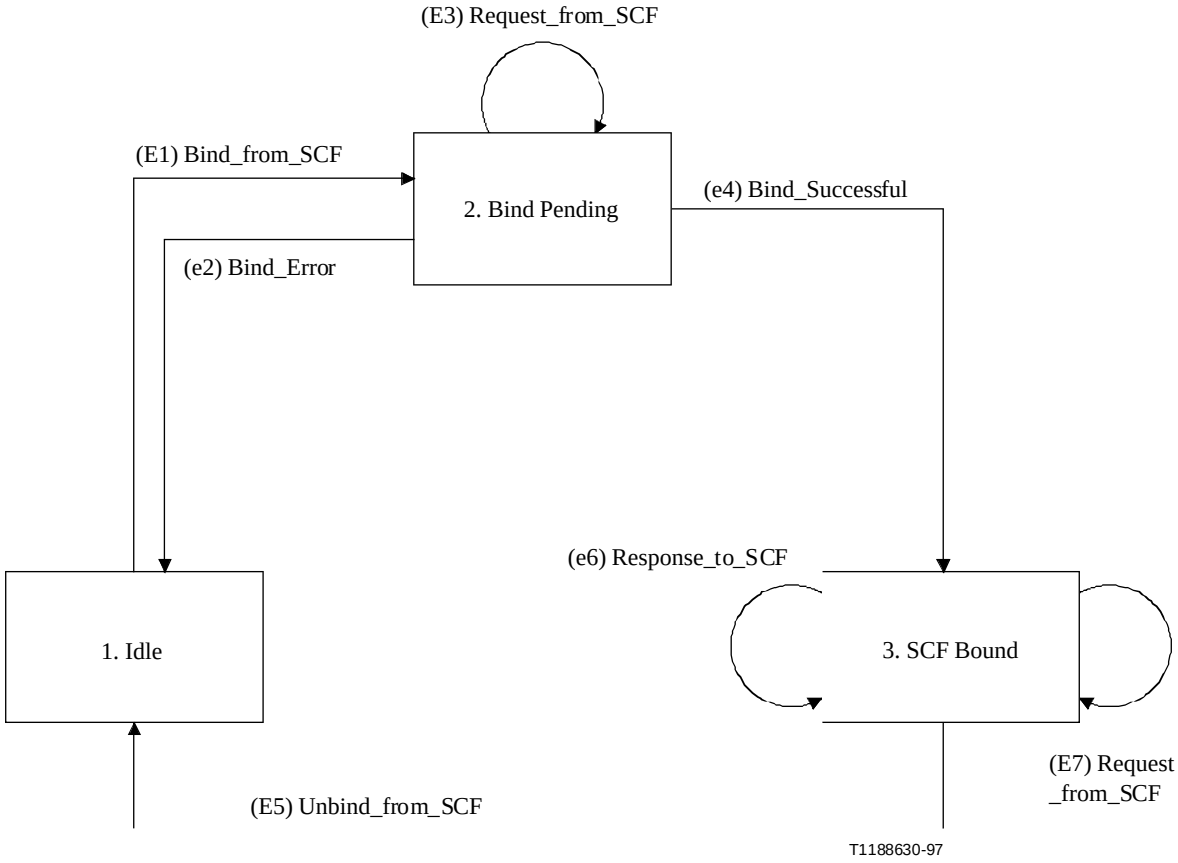


Figure 14-3: The SDF FSM

Each state is discussed in one of the following subclauses. General rules applicable to more than one state are as follows:

In any state, if the dialogue with the SCF is terminated, then the SDF FSM returns to **Idle** state after ensuring that any resources allocated to the call have been de-allocated.

15.4.1.1 State 1: “Idle”

The only event accepted in this state is:

(E1) Bind_from_SCF
This is an external event caused by the reception of directory Bind operation from the SCF. This event causes a transition out of this state to State 2, **Bind Pending**.

15.4.1.2 State 2: “Bind Pending”

In this state, a Bind request has been received from the SCF. The SDF is performing the SCF access control procedures behind the directoryBind operation (e.g. access authentication). There may also be a case such that the directoryBind operation is a dummy one. Then, the access authentication is not required. Three events are considered in this state:

(e2) Bind_Error
This is an internal event, caused by the failure of the directoryBind operation previously issued to the SDF. This event causes a transition out of this state to State 1, **Idle** and a directoryBind error is returned to the invoking SCF;

(E3) Request_from_SCF
This is an external event, caused by the reception of operations before the result from the directoryBind operation is determined.

It involves one of the following operations:

search
addEntry
removeEntry
modifyEntry
execute

The operations are stored and the SDSM remains in the same state. When a transition occurs to another state, the operations are re-examined as if they had occurred in that state.

(e4) Bind_Successful
This is an internal event, caused by the successful completion of the directoryBind operation previously issued to the SDF. This event causes a transition out of this state to State 3, **SCF Bound**.

15.4.1.3 State 3: "SCF Bound"

In this state, the access of the SCF to the SDF was authorized and operations coming from the SCF are accepted. Besides waiting for requests from the SCF, the SDF can send in that state responses to previously issued operations. Three events are considered in this state:

(E5) Unbind_from_SCF
This is an external event, caused by the reception of the in-directoryUnbind operation from the SCF. The SCF-SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1, **Idle**;

(e6) Response_to_SCF
This is an internal event, caused either by the completion of the operations previously issued by the SCF or by generation of a referral error to the SCF. Responses or referrals are sent to the SCF. The SDSM remains in the same state; and

(E7) Request_from_SCF
This is an external event, caused by the reception of a request from the SCF to the SDF.

It involves one of the following operations:

search
addEntry
removeEntry
modifyEntry
execute

The SDSM remains in the same state.

15.4.2 SDF state transition model for SDF related states

The SDF's job relating to interactions with other SDFs is to act upon shadowing and chaining operations. States related to SDF/SDF interactions for shadowing are given in subclause 14.4.2.1. States related to SDF/SDF interactions for chaining are given in subclause 14.4.2.2.

15.4.2.1 SDF state transition models for shadowing

As for the shadowing procedure, an SDF can play the role of a copy supplier and a copy consumer. Moreover, the shadowing procedure can be initiated by a copy consumer as well as a copy supplier. Therefore, there can be in total four FSMs as described below.

The four different SDF FSMs could be gathered into one more complex FSM, but to clearly show the different roles played by an SDF, it was thought better to have four separate FSMs.

In the following FSMs, the possibility of sending the DSAShadowBind operation together with other DISP operations in one TC message is taken into consideration.

Shadow supplier-initiated supplier state machine (SDSM-ShSSi)

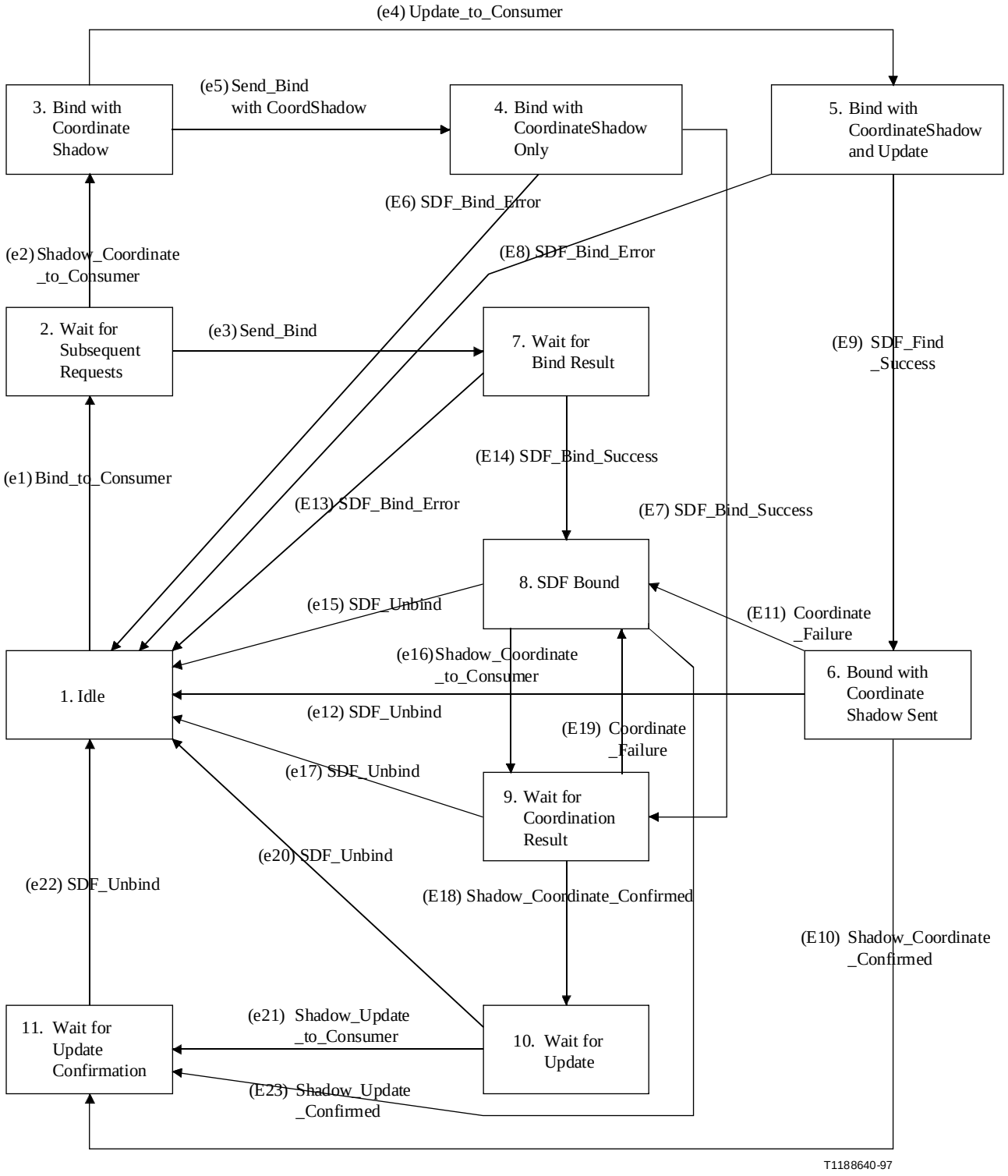


Figure 14-4: SDF FSM for a copy supplier in case of supplier-initiated (SDSM-ShSSi)

State 1: "Idle"

There is only one event accepted in this state:

(e1) Bind_to_Consumer
This is an internal event caused by the request to execute a DSAShadowBind operation. This causes a transition out of this state to State 2 **Wait for Subsequent Requests**.

State 2: "Wait for Subsequent Requests"

In this state, a CoordinateShadowUpdate operation to be sent with the DSAShadowBind operation (in the same message) to the consumer is expected. The following two events are considered in this state:

(e2) Shadow_Coordinate_to_Consumer
This is an internal event caused by the reception of a CoordinateShadowUpdate operation. The operation is buffered until the reception of a delimiter (or a timer expiration). This event causes a transition out of this state to State 3 **Bind with Coordinate Shadow**;

(e3) Send_Bind
This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind operation is sent to the consumer SDF. This event causes a transition out of this state to State 7 **Wait for Bind Result**.

State 3: "Bind with Coordinate Shadow"

In this state, an UpdateShadow operation to be sent with the DSAShadowBind and CoordinateShadowUpdate operations, or a delimiter is expected. Two events are considered in this state:

(e4) Update_to_Consumer
This is an internal event, caused by the reception of an UpdateShadow. This event causes a TCAP message containing the DSAShadowBind, CoordinateShadowUpdate and UpdateShadow operations to be sent to the consumer SDF. This event causes a transition out of this state to State 5 **Bind with Coordinate Shadow and Update**;

(e5) Send_Bind_with_CoordShadow
This is an internal event, caused by the reception of a delimiter that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind and CoordinateShadowUpdate operations is sent to the consumer SDF. This event causes a transition out of this state to State 4 **Bind with Coordinate Shadow Only**.

State 4: "Bind with Coordinate Shadow Only"

In this state, a DSAShadowBind result is expected from the consumer SDF. Two events are considered in this state:

(E6) SDF_Bind_Error
This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 **Idle**;

(E7) SDF_Bind_Success
This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to State 9 **Wait for Coordination Result**.

State 5: "Bind with Coordinate Shadow and Update"

In this state, a DSAShadowBind result is expected from the consumer SDF. Two events are considered in this state:

(E8) SDF_Bind_Error
This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 **Idle**;

(E9) SDF_Bind_Success
This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to State 6 **Bound with Coordinate Shadow Sent**.

State 6: "Bound with Coordinate Shadow Sent"

In this state, a CoordinateShadowUpdate result is expected from the consumer SDF. Three events are considered in this state:

(E10) Shadow_Coordinate_Confirmed
This is an external event, caused by the reception of a CoordinateShadowUpdate result. This indicates the successful completion of the CoordinateShadowUpdate operation previously issued to the consumer SDF. This event causes a transition out of this state to State 11 **Wait for Update Confirmation**;

(E11) Coordinate_Failure
This is an external event, caused by the reception of an error to the previously issued CoordinateShadowUpdate operation. This event causes a transition out of this state to State 8 **SDF Bound**;

(e12) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**.

State 7: "Wait for Bind Result"

In this state, a DSAShadowBind result is expected from the consumer. Two events are considered in this state:

(E13) SDF_Bind_Error
This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 **Idle**;

(E14) SDF_Bind_Success
This is an external event, caused by the successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to State 8 **SDF Bound**.

State 8: "SDF Bound"

In this state, the SDF has established an "authenticated association" to the consumer and is ready to send a CoordinateShadowUpdate operation to it. Two events are considered in this state:

(e15) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or causing the issuing of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 **Idle**;

(e16) Shadow_Coordinate_to_Consumer
This is an internal event, that causes the sending of a request to a consumer SDF to coordinate the shadow to have it later updated. This event causes a transition out of this state to State 5 **Wait for Coordination Result**.

State 9: "Wait for Coordination Result"

In this state, the supplier SDF has sent a CoordinateShadowUpdate request and waits for the answer from the consumer SDF. Three events are considered in this state:

(e17) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(E18) Shadow_Coordinate_Confirmed
This is an external event, caused by the reception of the response to the previously issued CoordinateShadowUpdate operation. This event causes a transition out of this state to State 6 **Wait for Update**;

(E19) Coordinate_Failure
This is an external event caused by the reception of an error to the previously issued CoordinateShadowUpdate request operation. This event causes a transition back to State 8 **SDF Bound**.

State 10: "Wait for Update"

In this state, the supplier SDF has received a confirmation to the previously issued Coordinate ShadowUpdate request and is ready to send an UpdateShadow request to the consumer SDF. Two events are considered in this state:

(e20) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(e21) Shadow_Update_to_Consumer
This is an internal event, that causes the sending of a request to the consumer SDF to update the shadow. This event causes a transition out of this state to State 11 **Wait for Update Confirmation**.

State 11: "Wait for Update Confirmation"

In this state, the supplier SDF has sent a UpdateShadow request and waits for the answer from the consumer SDF. Two events are considered in this state:

(e22) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(E23) Shadow_Update_Confirmed
This is an external event caused by the reception of the response to the previously issued UpdateShadow operation. This event causes a transition out of this state to State 8 **SDF Bound**.

Shadow consumer-initiated supplier state machine (SDSM-ShSCi)

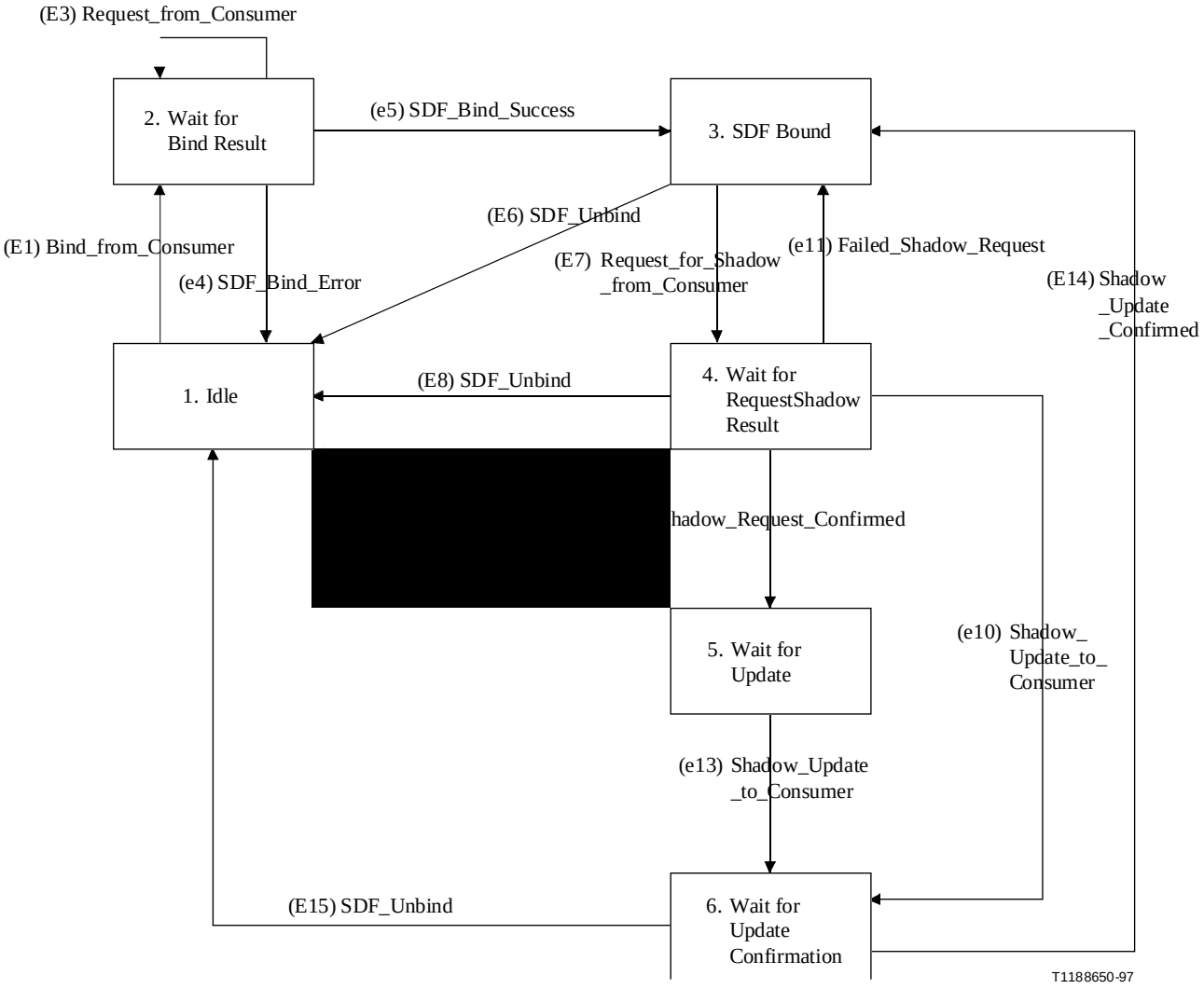


Figure 14-5: SDF FSM for a copy supplier in case of consumer-initiated (SDSM-ShSCi)

State 1: "Idle"

Two events are accepted in this state:

(E1) Bind_from_Consumer
This is an external event caused by the reception of a DSAShadowBind operation. This causes a transition out of this state to State 2 **Wait for Bind Result**;

State 2: "Wait for Bind Result"

In this state, a DSAShadowBind operation is being performed. Three events are considered in this state:

(E3) Request_from_Consumer
This is an external event, caused by the reception of a RequestShadowUpdate operation before the result from the DSAShadowBind operation is determined. This occurs when the RequestShadowUpdate is sent in the same TCAP message as the bind request. The RequestShadowUpdate message is stored and a transition occurs to the same state. When a transition occurs to the next state, the RequestShadowUpdate is re-examined as if it occurred in that state;

(e4) SDF_Bind_Error
This is an internal event, caused by the failure of the DSAShadowBind operation previously issued from the consumer. A DSAShadowBind error is returned. This event causes a transition out of this state to State 1 **Idle**;

(e5) SDF_Bind_Success
This is an internal event, caused by the successful completion of the DSAShadowBind operation previously issued from the consumer. This event causes a DSAShadowBind result to be returned, and a transition out of this state to State 3 **SDF Bound**.

State 3: "SDF Bound"

In this state, the supplier SDF is expecting a RequestShadowUpdate operation from the consumer SDF. Two events are considered in this state:

(E6) SDF_Unbind
This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or by the reception of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 **Idle**;

(E7) Request_for_Shadow_from_Consumer
This is an external event, caused by the reception of a RequestShadowUpdate operation from the consumer SDF. This event causes a transition out of this state to State 4 **Wait for RequestShadow Result**.

State 4: "Wait for RequestShadow Result"

In this state, the supplier SDF has received a RequestShadowUpdate operation from the consumer SDF. Four events are considered in this state:

(E8) SDF_Unbind
This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(e9) Shadow_Request_Confirmed
This is an internal event, that signals that the update agreement is acceptable. This causes the sending of a response to the previously received RequestShadowUpdate operation by the supplier SDF. This event causes a transition out of this state to State 5 **Wait for Update**;

(e10) Shadow_Update_to_Consumer
This is an internal event, that signals that the update agreement is acceptable, and an UpdateShadow message is ready to be sent. This causes the sending of both a RequestShadowUpdate result and an UpdateShadow operation in the same TCAP message. This event causes a transition out of this state to State 6 **Wait for Update Confirmation**;

(e11) Failed_Shadow_Request
This is an internal event, caused by the sending of an error to a previously received RequestShadowUpdate operation. This event causes a transition out of this state to State 3 **SDF Bound**.

State 5: "Wait for Update"

In this state, the supplier SDF has sent a response to the previously received RequestShadowUpdate operation and is ready to send an UpdateShadow operation to the consumer SDF. Two events are considered in this state:

(E12) SDF_Unbind
This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(e13) Shadow_Update_to_Consumer
This is an internal event that causes the sending of a request to the consumer SDF to update the shadow. This event causes a transition out of this state to State 6 **Wait for Update Confirmation**.

State 6: "Wait for Update Confirmation"

In this state, the supplier SDF has sent an UpdateShadow request and waits for the answer from the consumer SDF. Two events are considered in this state:

(E14) Shadow_Update_Confirmed
This is an external event caused by the reception of the response to the previously issued UpdateShadow operation. This event causes a transition out of this state to State 3 **SDF Bound;**

(E15) SDF_Unbind

This is an external event, caused by the cancellation of the “authenticated association” established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 Idle.

Shadow supplier-initiated consumer state machine (SDSM-ShCSi)

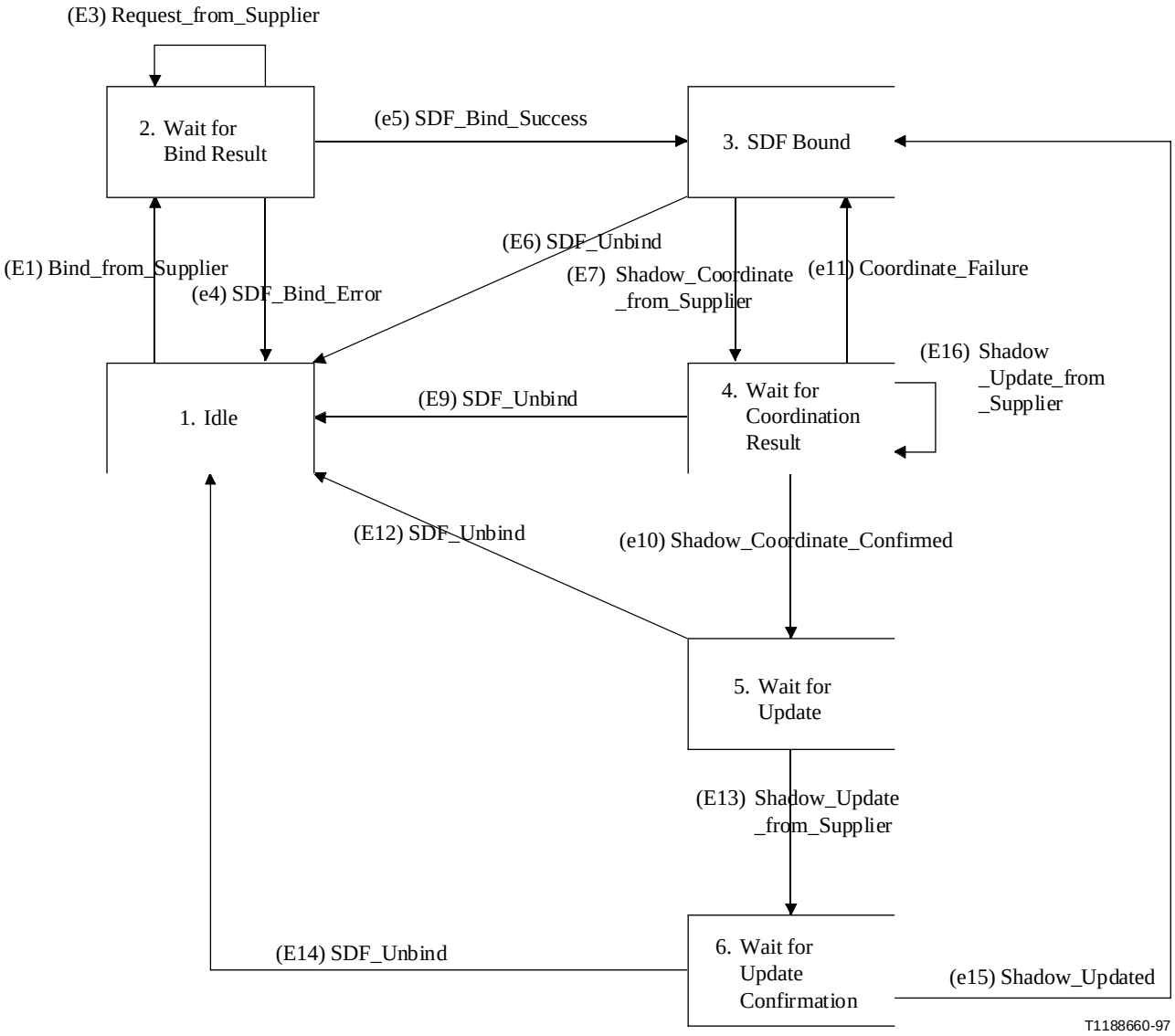


Figure 14-6: SDF FSM for a copy consumer in case of supplier-initiated (SDSM-ShCSi)

State 1: “Idle”

Two events are accepted in this state:

(E1) Bind_from_Supplier

This is an external event caused by the reception of a DSAShadowBind operation. This causes a transition out of this state to State 2 **Wait for Bind Result**;

State 2: "Wait for Bind Result"

In this state, the consumer has received a DSAShadowBind operation and is answering that operation. Three events are considered in this state:

(E3) Request_from_Supplier
This is an external event, caused by the reception of operations before the result from the DSAShadowBind operation is determined. This occurs when the CoordinateShadowUpdate or UpdateShadow are sent in the same TCAP message as the bind request. These operations are stored and a transition occurs to the same state. When a transition occurs to the following states, these operations are re-examined as if they occurred in that state;

(e4) SDF_Bind_Error
This is an internal event, caused by the failure of the DSAShadowBind operation previously issued from the supplier SDF. A DSAShadowBind error is returned. This event causes a transition out of this state to State 1 **Idle**;

(e5) SDF_Bind_Success
This is an internal event, caused by the successful completion of the DSAShadowBind operation previously issued from the supplier SDF. This event causes a transition out of this state to State 3 **SDF Bound**.

State 3: "SDF Bound"

In this state, the consumer SDF is expecting a CoordinateShadowUpdate operation from the supplier SDF. Two events are considered in this state:

(E6) SDF_Unbind
This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or by the reception of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 **Idle**;

(E7) Shadow_Coordinate_from_Supplier
This is an external event, caused by the reception of a CoordinateShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to State 4 **Wait for Coordination Result**;

State 4: "Wait for Coordination Result"

In this state, the consumer SDF has received a CoordinateShadowUpdate operation from the supplier SDF and is processing it. Three events are considered in this state:

(E9) SDF_Unbind
This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(e10) Shadow_Coordinate_Confirmed
This is an internal event, caused by the successful completion of a CoordinateShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to State 5 **Wait for Update**;

(e11) Coordinate_Failure
This is an internal event, caused by the sending of an error to a previously received CoordinateShadowUpdate operation. This event causes a transition out of this state to State 3 **SDF Bound**.

(E16) Shadow_Update_from_Supplier
This is an external event, caused by the reception of a UpdateShadow operation in the same TCAP message as a CoordinateShadowBind and CoordinateShadowUpdate. The UpdateShadow message is stored and a transition occurs to the same state. When a transition occurs to the next state, the UpdateShadow is re-examined as if it had occurred in that state.

State 5: "Wait for Update"

In this state, the SDF is expecting an UpdateShadow operation. Two events are considered in this state:

(E12) SDF_Unbind
This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(E13) Shadow_Update_from_Supplier

This is an external event, caused by the reception of the UpdateShadow operation issued from the supplier SDF. This event causes a transition out of this state to State 6 **Wait for Update Confirmation**.

State 6: "Wait for Update Confirmation"

In this state, the consumer SDF has received an UpdateShadow operation from the supplier SDF and processes to update the copy. Two events are considered in this state:

(E14) SDF_Unbind

This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(e15) Shadow_Updated

This is an internal event, caused by the completion of an UpdateShadow operation and the sending of the response to it. This event causes a transition out of this state to State 3 **SDF Bound.**

Shadow consumer-initiated consumer state machine (SDSM-ShCCi)

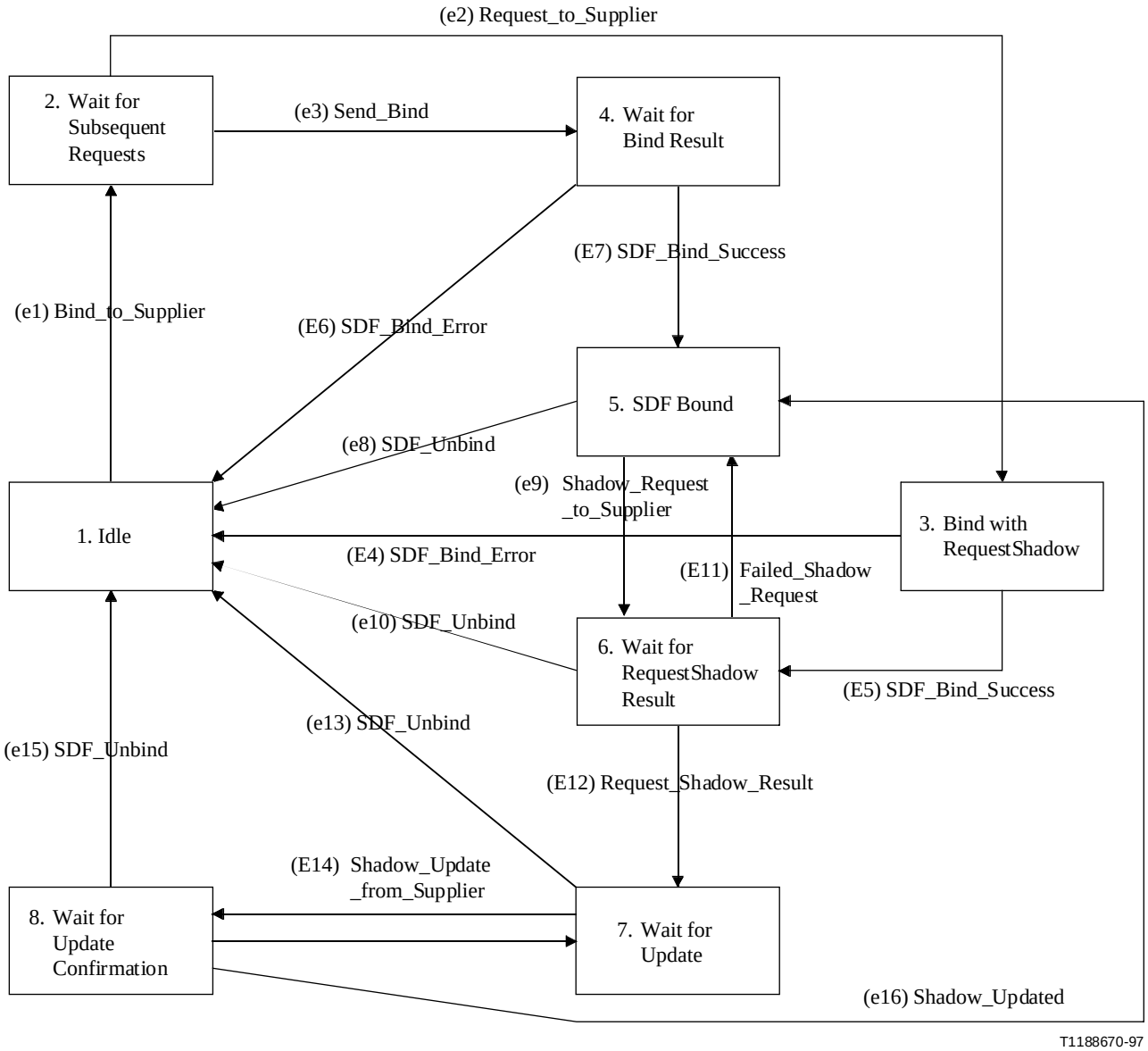


Figure 14-7: SDF FSM for a copy consumer in case of consumer-initiated (SDSM-ShCCi)

State 1: "Idle"

There is only one event accepted in this state:

(e1) Bind_to_Supplier
This is an internal event caused by the request to execute a DSAShadowBind operation. This event causes a transition out of this state to State 2 **Wait for Subsequent Requests**.

State 2: "Wait for Subsequent Requests"

In this state, a RequestShadowUpdate operation to be sent with the DSAShadowBind operation (in the same message) to the supplier is expected. The following two events are considered in this state:

(e2) Request_to_Supplier
This is an internal event caused by the reception of a RequestShadowUpdate operation. This event causes a TCAP message containing the DSAShadowBind and RequestShadowUpdate operations to be sent to the supplier SDF. This event causes a transition out of this state to State 3 **Bind with RequestShadow**;

(e3) Send_Bind
This is an internal event caused by the reception of a delimiter that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind operation is sent to the supplier SDF. This event causes a transition out of this state to State 4 **Wait for Bind Result**.

State 3: "Bind with RequestShadow"

In this state, a DSAShadowBind result is expected from the supplier SDF. Two events are considered in this state:

(E4) SDF_Bind_Error
This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the supplier SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 **Idle**;

(E5) SDF_Bind_Success
This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the supplier SDF. This event causes a transition out of this state to State 6 **Wait for RequestShadow Result**.

State 4: "Wait for Bind Result"

In this state, a DSAShadowBind result is expected from the supplier SDF. Two events are considered in this state:

(E6) SDF_Bind_Error
This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the supplier SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 **Idle**;

(E7) SDF_Bind_Success
This is an external event, caused by the successful completion of the DSAShadowBind operation previously issued to the supplier SDF. This event causes a transition out of this state to State 5 **SDF Bound**.

State 5: "SDF Bound"

In this state, the consumer SDF is ready to send a RequestShadowUpdate operation to the supplier SDF. Two events are considered in this state:

(e8) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or causing the issuing of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 **Idle**;

(e9) Shadow_Request_to_Supplier
This is an internal event, caused by the sending of a RequestShadowUpdate operation to the supplier SDF. This event causes a transition out of this state to State 6 **Wait for RequestShadow Result**.

State 6: "Wait for RequestShadow Result"

In this state, the consumer SDF has sent a RequestShadowUpdate operation and waits for the answer from the supplier SDF. Three events are considered in this state:

(e10) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(E11) Failed_Shadow_Request
This is an external event, caused by the reception of an error to the previously issued RequestShadowUpdate operation. This event causes a transition out of this state to State 5 **SDF Bound**;

(E12) Request_Shadow_Result
This is an external event, caused by the reception of the response to the previously issued RequestShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to State 7 **Wait for Update**.

State 7: "Wait for Update"

In this state, the consumer SDF has received a RequestShadowUpdate result and waits for an UpdateShadow operation from the supplier SDF. Two events are considered in this state:

(e13) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(E14) Shadow_Update_from_Supplier
This is an external event caused by the reception of an UpdateShadow operation issued from the supplier SDF. This event causes a transition out of this state to State 8 **Wait for Update Confirmation**.

State 8: "Wait for Update Confirmation"

In this state, the consumer SDF has received an UpdateShadow operation from the supplier SDF and processes to update the copy. Two events are considered in this state:

(e15) SDF_Unbind
This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 **Idle**;

(e16) Shadow_Updated
This is an internal event, caused by the completion of an UpdateShadow operation and the sending of the response to it. This event causes a transition out of this state to State 5 **SDF Bound**.

15.4.2.2 SDF state transition models for chaining

As for the chaining procedure, an SDF can act as a chaining initiator and as a chaining terminator. Therefore, there are two FSMs as described below.

In the following FSMs, the possibility of sending the DSABind operation together with other DSP operations in one TC message is taken into considerations.

SDF state transition models for chaining initiation (SDSM-ChI)

The Finite State Machine for an SDFs interaction with another SDF when acting as a chaining initiator is depicted in figure 14-8.

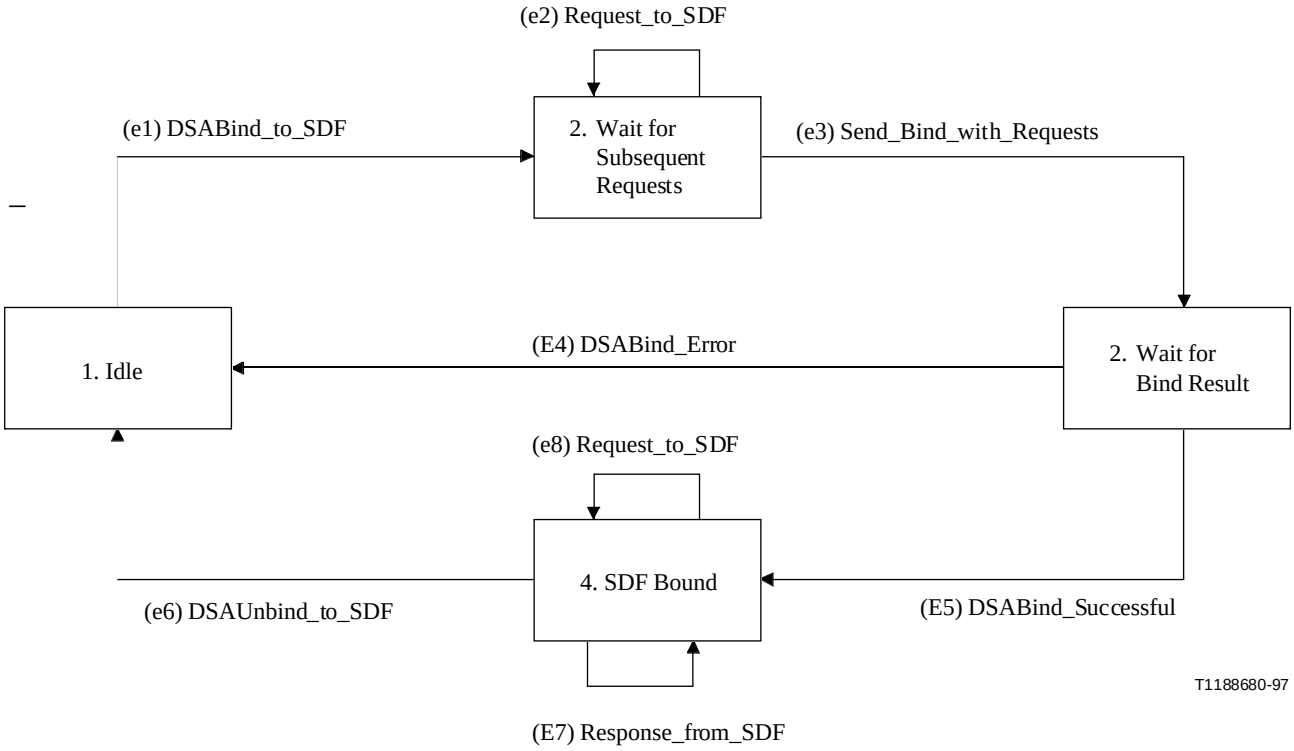


Figure 14-8: SDF/SDF chaining initiator finite state machine (SDSM-ChI)

State 1: "Idle"

The only event accepted in this state is:

(e1) DSABind_to_SDF
This is an internal event causing the sending of the DSABind operation to the SDSM-ChT. This event causes a transition out of this state to State 2 **Wait for Subsequent Requests**.

State 2: "Wait for Subsequent Requests"

In this state, subsequent operations to be sent with the DSABind operation (in the same message) to the SDSM-ChT are expected. The following two events are considered in this state:

(e2) Request_to_SDF
This is an internal event causing the sending of an operation. It involves one of the following operations:

- chainedSearch;
- chainedAddEntry;
- chainedRemoveEntry;
- chainedModifyEntry
- chainedExecute.

The operation is buffered until the reception of a delimiter (or a timer expiration). This event causes a transition to the same state;

(e3) Send_Bind_with_Requests
This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing those arguments of the DSABind operation and other operations, if any, are sent to the SDSM-ChT. This event causes a transition out of this state to State 3 **Wait for Bind Results**.

State 3: "Wait for Bind Result"

In this state, a DSABind request has been sent to the SDSM-ChT. The SDSM-ChT is performing the SDF access control procedures associated with the DSABind operation (e.g. access authentication). Two events are considered in this state:

(E4) DSABind_Error
This is an external event caused by the failure of the DSABind operation previously issued to the SDSM-ChT. This event causes a transition out of this state to State 1 **Idle**;

(E5) DSABind_Successful
This is an external event caused by the reception of the DSABind confirmation for the DSABind operation previously issued to the SDSM-ChT. This event causes a transition out of this state to State 4 **SDF Bound**.

State 4: "SDF Bound"

In this state, the access of the SDSM-ChI to the SDSM-ChT was authorized, chained operations can be sent to the SDSM-ChT, and results of chained operations coming from the SDSM-ChT are accepted. Three events are considered in this state:

(e6) DSAUnbind_to_SDF
This is an internal event, causing the sending of the in-DSAUnbind operation to the SDSM-ChT. The SDF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 **Idle**;

(E7) Response_from_SDF
This is an external event, caused either by the reception results of the operations previously issued by the SDSM-ChI or reception of a referral from the SDSM-ChT. The SDSM-ChI remains in the same state;

(e8) Request_to_SDF
This is an internal event, causing the sending of a chained operation to the SDSM-ChT.

It involves one of the following operations:

– chainedSearch;

– chainedAddEntry;

– chainedRemoveEntry;

– chainedModifyEntry

– chainedExecute.

The SDSM-ChI remains in the same state.

SDF state transition models for chaining termination (SDSM-ChT)

The Finite State Machine for an SDFs interaction with another SDF when acting as a chaining terminator is depicted in figure 14-9.

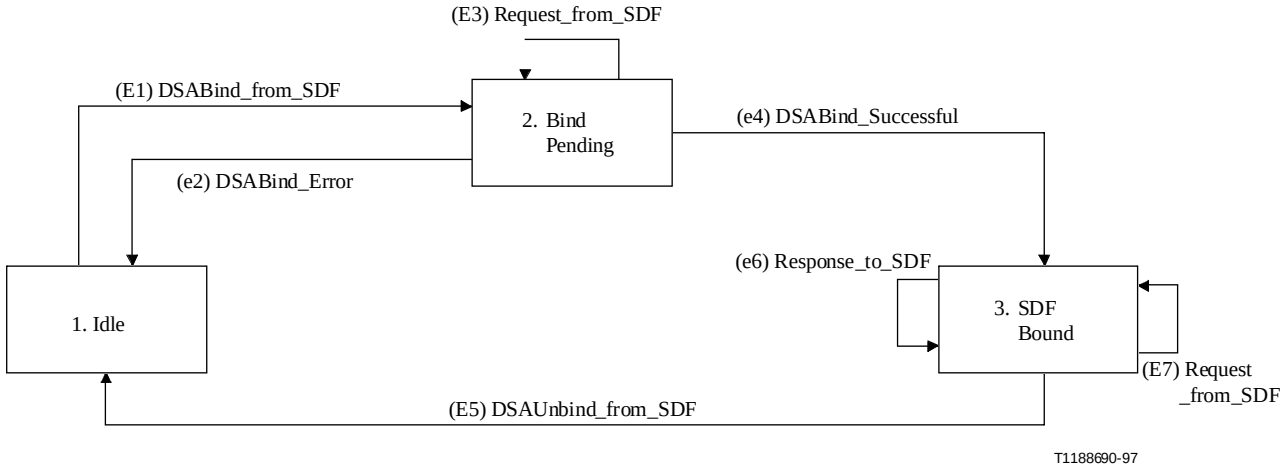


Figure 14-9: SDF/SDF chaining terminator finite state machine (SDSM-ChT)

State 1: "Idle"

The only event accepted in this state is:

(E1) DSABind_from_SDF
This is an external event caused by the reception of the DSABind operation from an SDSM-ChI. This event causes a transition out of this state to State 2 **Bind Pending**.

State 2: "Bind Pending"

In this state, a DSABind request has been received from the SDSM-ChI. The SDSM-ChT is performing the SDF access control procedures associated with the DSABind operation (e.g. access authentication). Two events are considered in this state:

(e2) DSABind_Error
This is an internal event caused by the failure of the DSABind operation previously issued from the SDSM-ChI. This event causes a transition out of this state to State 1 **Idle**, and a Bind error is returned to the SDSM-ChI;

(E3) Request_from_SDF
This is an external event, caused by the reception of operations before the result from the DSABind operation is determined.

It involves one of the following operations:

chainedSearch
chainedAddEntry
chainedRemoveEntry
chainedModifyEntry
chainedExecute

The operations are stored and the SDSM-ChT remains in the same state. When a transition occurs to another state, the operations are re-examined as if they had occurred in that state;

(e4) DSABind_Successful
This is an internal event caused by the successful completion of the DSABind operation previously issued from the SDSM-ChI. This event causes a transition out of this state to State 3 **SDF Bound**.

State 3: "SDF Bound"

In this state, the access of the SDSM-ChI to the SDSM-ChT was authorized and chained operations coming from the SDSM-ChI are accepted. Besides waiting for requests from the SDSM-ChI, the SDSM-ChT can send in this state responses to previously issued operations. Three events are considered in this state:

(E5) DSAUnbind_from_SDF
This is an external event, caused by the reception of the in-DSAUnbind operation from the SDSM-ChI. The SDF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 **Idle**;

(e6) Response_to_SDF
This is an internal event, caused either by the completion of operations previously issued by the SDSM-ChI or generation of a referral to the SDSM-ChI. Responses/referrals are sent to the SDSM-ChI. The SDSM-ChT remains in the same state;

(E7) Request_from_SDF
This is an external event, caused by the reception of a request from the SDSM-ChI.

It involves one of the following operations:

chainedSearch
chainedAddEntry
chainedRemoveEntry
chainedModifyEntry
chainedExecute

The SDSM-ChT remains in the same state.

16 CUSF AE procedures

16.1 General

This subclause provides the definition of the CUSF AE procedures related to the CUSF-SCF interface. The procedures are based on the use of Common Channel SS7; other signalling systems can be used.

Capabilities not explicitly covered by these procedures may be supported in an implementation dependent manner in the SSP, CUSP, SN, while remaining in line with clause 2.

The AE, following the architecture defined in the ITU-T Recommendations Q.700 [19], Q.1400 [30] and in ETS 300 287-1 [7] includes TC (transaction capabilities application part) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE which interfaces with TC using the primitives specified in ETS 300 287-1 [7]; other signalling systems may be used.

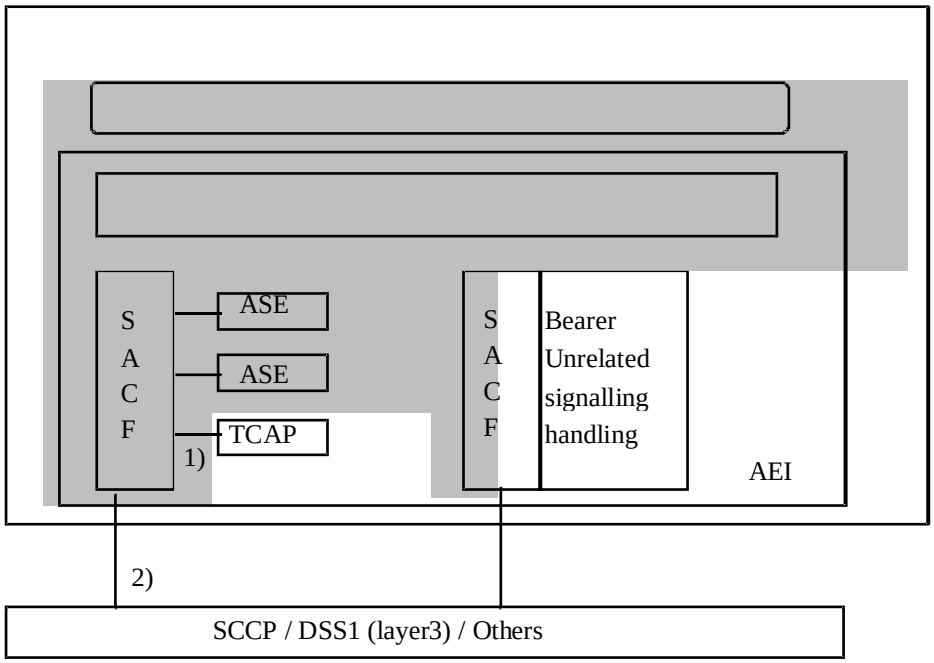
The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the AE procedures defined in the present clause differ from detailed procedures and the rules for using of TC service, the statements and rules contained in the present clause shall prevail.

16.2 Model and interfaces

The functional model of the AE-CUSF is shown in figure 15-1; the ASEs interface to TC to communicate with the SCF, and interface to the CCF, the SSF, and the maintenance functions already defined for switching systems if needed. The scope of this ITU-T Recommendation is limited to the shaded area in figure 15-1.

The interfaces shown in figure 15-1 use the TC-user ASE primitives specified in ETS 300 287-1 [7] and N-Primitives specified in ETS 300 009-1 [3]. The operations and parameters of intelligent network application protocol (INAP) are defined in clauses 3 to 10.



1) TC-primitives
2) N-primitives

Figure 15-1: Functional model of CUSF AE

16.2.1 Background for the modeling and protocol

The BCUSM depicts the handling of association for the call unrelated case.

16.2.2 Modelling and protocol

The main aspect is how to model the handling of the Call Unrelated User Interaction (CUUI) APDUs over the DSS1 or TC interface (e.g. USI IEs, TC component), because, as mentioned in the previous subclause, they are service specific and cannot be explicitly modelled except for the general parts (header handling and association handling).

It is similar to the BCSM how far the state model for the interaction depicts the details of the activities in the PE, but the interaction differs from the circuit-mode switched bearer services (represented by the BCSM) with the following points:

- the variation of the service triggering points (corresponding to TDP) is limited to the association establishment or release; additional TDP criteria such as CUUI APDU (e.g. usiServiceIndicator, cUApplicationInd, ...) can be used;
- the CUUI APDU is received over the DSS1 at the TC interface during the association establishment/release phase or within an active association;
- the modeling of the analysis of a received CUUI APDU may be not necessary, because it can be well modeled as a TDP criteria check;
- the procedures with the interaction vary from a service to one another and have many variations, so each service has different states to handle the CUUI APDUs.

Considering these points and the background, the current BCUSM only models the association handling parts over the DSS1 or TC interface. The Point In Association (PIA) shows the association handling status and each DP models one event: association establishment request/association release. (Figure 15-2 shows what part is service dependent and what part is general for this type of interaction.)

16.3 Relations between CUSF FSM and the SSF/CCF and maintenance functions

The primitive interface between the CUSF FSM and the CCF/SSF/maintenance functions is an internal interface and is not subject to standardization in CS2. Nevertheless this interface should be in line with the BCUSM defined in clause 8.

The relationship between the BCUSM and the CUSF FSM may be described as follows in case of a call unrelated association initiated by an end user or by an IN SL:

- When a call unrelated association is initiated by an end user and processed at an exchange, an instance of a BCUSM is created. As the BCUSM proceeds, it encounters detection points (see clause 8). If a DP is armed as a Trigger DP (TDP) an instance of a CUSF FSM is created;
- If an InitiateAssociation is received from the SCF, an instance of a BCUSM is created, as well as an instance of a CUSF FSM.

The CUSF logic should:

- perform the DP processing actions specified in clause 8/Q.1224 [26], including if DP criteria are met;
- check for SCF accessibility;
- handle service feature interactions in conjunction with the SSF.

The management functions related to the execution of operations received from the SCF are executed by the CUSF Management Entity (CUSME). The CUSME comprises a CUSME-Control and several instances of CUSME FSMs. The CUSME-control interfaces the different CUSF FSMs and CUSME FSMs respectively and the FEAM. Figure 15-3-1 shows the CUSF Interfaces.

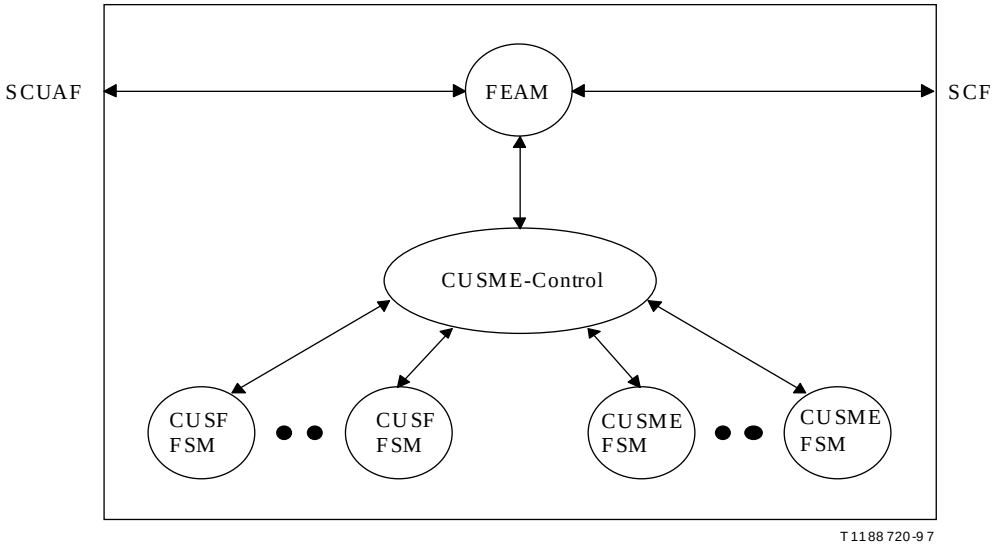


Figure 15-3-1: CUSF interfaces

The FEAM provides the low level interface maintenance functions including the following:

1) Establishing and maintaining the interfaces to the SCF;
2) Passing and queueing (when necessary) the messages received from the SCF to the CUSME-Control;
3) Formatting, queueing (when necessary), and sending the messages received from the CUSME-Control to the SCF.

The CUSME-control maintains the dialogues with the SCF on behalf of all instances of the CUSF FSM. These instances of the CUSF FSM occur concurrently and asynchronously as associations occur, which explains the need for a single entity that performs the task of creation, invocation, and maintenance of the CUSF FSMs. In particular the CUSME-control performs the following tasks:

1) Interprets the input messages from other FEs and translates them into corresponding CUSF FSM events;

2) Translates the CUSF FSM outputs into corresponding messages to other FEs;

3) Captures asynchronous (with processing association and/or operation request from the end user) activities related to management or supervisory functions in the CUSF and creates an instance of a CUSME FSM.

The CUSF FSM passes SCF instructions to the related instances of the BCUSM as needed. DPs may be dynamically armed as EDPs, requiring the CUSF FSM to remain active. At some point, further interaction with the SCF is not needed, and the CUSF FSM may be terminated while the BCUSM continues to handle the association as needed.

16.4 CUSF management FSM

The CUSME FSM only relates to the ActivityTest operation. The CUSME FSM only passes to the relevant CUSF FSM.

16.5 CUSF state transition diagram

Figure 15-2 shows the state diagram of the CUSF part of the SSP, CUSP, SN during the processing of an IN association request from the user or IN SL.

Each state is discussed in the following subclauses. General rules applicable to more than one state are addressed here.

One or a sequence of CUUI APDUs received in one or more TC messages may include a single operation or multiple operations, and is processed as follows:

- Process the operations in the order in which they are received.

- Each operation causes a state transition independent of whether or not a single operation or multiple operations are received in a message.

- The CUSF examines subsequent operations in the sequence. As long as sequential execution of these operations would leave the FSM in the same state, it will execute them.

- If there is an error in processing one of the operations in the sequence, the CUSF FSM processes the error (see below) and discards all remaining operations in the sequence.

- If an operation is not understood or is out of context (i.e. violates the SACF rules defined by the CUSF FSM) as described above, ABORT the interaction.

In any state, if there is an error in a received operation, the maintenance functions are informed and the CUSF FSM remains in the same state as when it received the erroneous operation; depending on the class of the operation, the error could be reported by the CUSF to the SCF using the appropriate component (see ETS 300 287-1 [7]).

In any state (except Idle), if the association requesting party abandons the association before it is established (i.e. before the Active PIA in the BCUSM), then the CUSF FSM should clear the association and ensure that any CUSF and CCF resources allocated to the association have been de-allocated, then moves to the Idle state.

The CUSF has an application timer, T_{CUSF}, whose purpose is to prevent excessive association processing suspension time and to guard the association between the CUSF and the SCF.

Timer T_{CUSF} is set in the following cases:

- when the CUSF sends an initialAssociationDP or an eventReportBCUSM and the CUSF FSM goes to the "Waiting For Instructions" state (see subclause 15.5.2. State b: Waiting For Instructions).

On expiration of T_{CUSF} the CUSF FSM transits to the Idle state, and aborts the interaction with the SCF, and the CUSF progresses the BCUSM if possible. This timer could be rearmed by the resetTimer operation.

The CUSF state diagram contains the following transitions (events):

er1	TDP-R encountered
er2	Idle return from waiting for instructions
er3	Monitoring instruction received.
er4	TDP-N encountered
er5	Association handling information received (if EDP(s) armed)
er6	EDP-N not last encountered
er7	EDP-N last encountered
er8	EDP-R encountered
er10	Initiate association received

The CUSF state diagram contains the following states:

State a	Idle
State b	Waiting For Instructions
State c	Monitoring

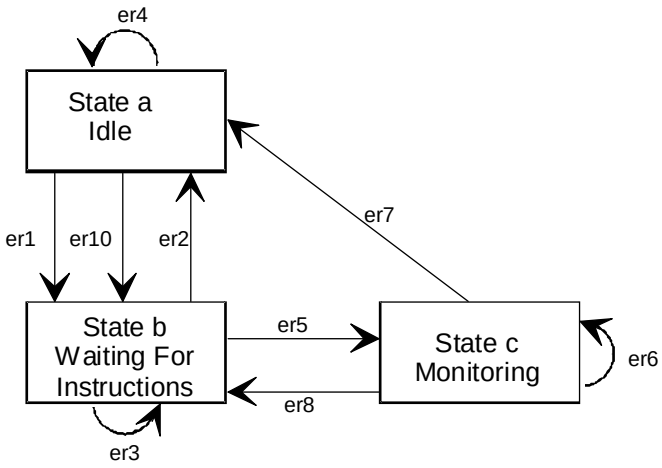


Figure 15-2: CUSF FSM

16.5.1 State a: Idle

The CUSF FSM enters the Idle state when sending or receiving an ABORT TC primitive due to abnormal conditions in any state.

The CUSF FSM enters the Idle state when one of the following occurs:

- when the association is released by the end user request in the Waiting For Instructions (transition er2) or in the Monitoring (transition er7);
- when a ReleaseAssociation operation is processed in the Waiting For Instructions (transition er2);
- when a last EDP-N is reported in the Monitoring (transition er7);
- when the application timer T_{CUSF} expires in the Waiting for Instructions state (transition er2).

During this state the following call unrelated associated event can occur:

- an armed TDP is encountered and the CUSF FSM acts as described below;
- if the DP is a TDP-N, the initialAssociationDP operation is sent to the SCF, as determined from DP processing; there is no resulting transition in the CUSF FSM to a different state (transition er4);
- if the DP is a TDP-R, the initialAssociationDP operation is sent to the SCF, as determined from DP processing, and the CUSF FSM transits to the Waiting For Instructions state (transition er1);
- a message related to a new transaction containing an InitiateAssociation operation is received from the SCF: in this case the CUSF FSM moves to the state Waiting For Instructions (transition er10).

Any other operation received from the SCF while the CUSF is in the Idle state should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TC (see ETS 300 287-1 [7]).

16.5.2 State b: Waiting For Instructions

This state is entered from the Idle state, as indicated above (transition er1) on sending an initialAssociationDP or from the "Monitoring" state on detection of a EDP-R.

In this state the CUSF FSM is waiting for an instruction from the SCF; association handling is suspended and an application timer (T_{CUSF}) should be set on entering this state.

During this state the following events can occur:

- The user releases the association. This should be processed in accordance with the general rules in subclause 15.5.
- The application Timer T_{CUSF} expires: the CUSF FSM moves to the Idle state, the T_{CUSF} expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF: The CUSF FSM acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the CUSF with no resulting transition to a different state (transition er3):

- RequestReportBCUSMEvent: this operation is sent before the "association processing" operation in order to avoid real-time message crossing.

The following operations may be received from the SCF and processed by the CUSF causing a state transition to the Idle state (transition er2) in case no EDP has been previously armed:

- ConnectAssociation;
- ContinueAssociation.

The following operation may be received from the SCF and processed by the CUSF, causing a state transition to the Monitoring state (transition er5) in case EDP(s) has(have) been previously armed:

- ConnectAssociation;
- ContinueAssociation.

ReleaseAssociation operation may be received from the SCF causing a state transition to the Idle state (transition er2). In this case, the CUSF FSM should release the association to the user and ensure that any CUSF resources allocated to the association have been de-allocated.

Any other operation received in this state should be processed in accordance with the general rules in subclause 15.5.

16.5.3 State c: "Monitoring"

The CUSF enters this state from the Waiting For Instructions state (transition er5) upon receiving a ConnectAssociation or ContinueAssociation if EDP(s) is (are) previously armed.

In this state the timer T_{CUSF} is not used; i.e., the expiration of T_{CUSF} does not have any impact on the CUSF FSM.

During this state the following events can occur:

- An EDP-N is reported to the SCF by sending an eventReportBCUSM operation; the CUSF FSM remains in the Monitoring state (transition er6) if one or more EDPs are armedor moves to the Idle state (transition er7) if there are no remaining EDPs armed.
- An EDP-R is reported to the SCF by sending an eventReportBCUSM operation; the CUSF FSM moves to the Waiting For Instructions state (transition er8).
- The receipt of an END or ABORT primitive from TC has no effect on the association; the association may continue or be completed with the information available. In this case, the CUSF FSM transits to the Idle state (transition er7), disassociating the CUSF FSM from the association.
- The user releases association. This should be processed in accordance with the general rules in subclause 15.5.

16.6 USI FSM

The CUSF-USI FSM illustrates state transitions for requesting and cancelling the monitoring of the reception of UTSI information from the User and the sending of STUI information to the User. The receipt of USI instructions from the SCFs has no effect on the CUSF FSM which remains in the same state.

The CUSF FSM has two states which are "Monitoring UTSI" and "Idle":

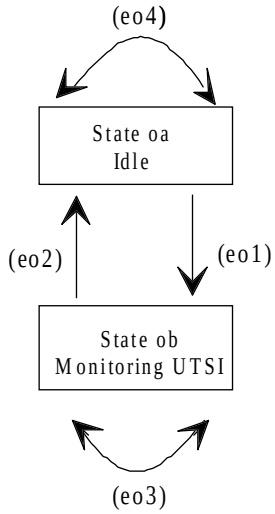


Figure 15-6: CUSF-USI FSM

The CUSF-USI FSM transitions are defined in the following way:

(eo1): The SCF requests the CUSF to monitor the receipt of an UTSI IE with a given *ServiceIndicator* value by sending a requestReportUTSI operation for a particular party indicated by the leg ID by setting the "monitorMode" value to "monitoringActive".

(eo2): The SCF requests the CUSF to stop monitoring the receipt of an UTSI IE with a given *ServiceIndicator* value by sending a requestReportUTSI operation for a particular party indicated by the leg ID by setting the "monitorMode" value to "monitoringInactive".

(eo3): The SCF either sends an STUI IE by means of the sendSTUI operation to the user indicated by the leg ID for a given USIServiceIndicator value, or receives it by means of the reportUTSI operation.

With the same operation, the SCF requests the CUSF to monitor or to stop monitoring the receipt of an UTSI IE with a given *USIServiceIndicator* value.

NOTE: As an SCF controls or monitors the association, the CUSF FSM is in any state except "Idle"; but the USI mechanism does not cause any transition in the CUSF FSM.

The CUSF FSM transition to the "Idle" state causes a USI FSM transition to the "Idle" state.

The SCF may send the sendSTUI operation when the USI FSM is in the Idle state and the CUSF FSM is not in the Idle state.

17 Error procedures

This clause defines the generic error procedures for the IN CS2 INAP. The error procedure descriptions have been divided in two subclauses, subclause 16.1 listing the errors related to INAP operations and subclause 16.2 listing the errors related to error conditions in the different FEs which are not directly related to the INAP operations.

17.1 Operation related error procedures

The following subclauses define the generic error handling for the operation related errors. The errors are defined as operation errors in subclauses 4-10. The TCAP services which are used for reporting operation errors are described in subclause 18.1.

Errors which have a specific procedure for an operation are described in subclauses 11-15 with the detailed procedure of the related operation.

All errors, which can be detected by the ASN.1 decoder, already may be detected during the decoding of the TCAP message and indicated by the TC error indication "MistypedParameter" in the TC-U-Reject.

17.1.1 AttributeError

17.1.1.1 General description

17.1.1.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report an attribute related problem. The conditions under which an attribute error is to be issued are defined in ITU-T Recommendation X.511 [39] subclause 12.4.

17.1.1.1.2 Argument description

The attribute error parameter and problem codes are specified in ITU-T Recommendation X.511 [39] subclause 12.4.

17.1.1.2 Operations SCF->SDF

AddEntry

Execute

ModifyEntry

Search

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 4 SDF Bound or

SCSM state 2 Wait for subsequent requests

Postcondition: SCSM state 4 SDF Bound or

SCSM state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SCSM state 4 SDF Bound

Postcondition: SCSM state 4 SDF Bound

Error Procedure is dependent on the SL. If the SCF is able to change the request, it can do another SDF query, otherwise the service processing should be terminated.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

B) Returning Error

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

The SDF could not perform the operation due to an attribute problem and therefore sends an Attribute error to the SCF. After returning the error, no further error treatment is performed.

17.1.1.3 Operations SDF->SDF

ChainedAddEntry

ChainedExecute
ChainedModifyEntry

ChainedSearch

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound

This error is reported to the SCF.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation due to an attribute problem and therefore sends an Attribute error to the other SDF. After returning the error, no further error treatment is performed.

17.1.2 Canceled

17.1.2.1 General Description

17.1.2.1.1 Error description

The Error "Canceled" gives an indication to the SCF that the cancellation, as it was requested by the SCF, of a specific Operation, has been successful. The SCF is only able to cancel certain predefined SCF-->SRF Operations.

17.1.2.2 Operations SCF->SRF

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

Procedures at invoking entity (SCF)

A) Sending Cancel

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

The SCF sends a Cancel after a **PA** or PromptAndCollectUserInformation has been sent. The SCF remains in the same state.

B) Receiving Canceled Error

Precondition: SCSM state any SL dependent.

Postcondition: SCSM state any SL dependent.

After sending a Cancel operation the SL may continue (e.g., sending more PlayAnnouncement or PromptAndCollectUserInformation or a DisconnectForwardConnection). The Canceled Error can therefore be received in any state. The treatment is SL dependent.

Procedures at responding entity (SRF)

A) Receiving Cancel

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

The indicated PlayAnnouncement or PromptAndCollectUserInformation is terminated if it is presently executing or deleted from the buffer. If the indicated PlayAnnouncement or PromptAndCollectUserInformation is already executed this causes a failure ("CancelFailed").

B) Sending Cancel Error

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

After returning the "Canceled" Error the SRF stays in the same state. The execution of the indicated PlayAnnouncement or PromptAndCollectUserInformation is aborted, i.e., the SRF remains connected and the next PlayAnnouncement or PromptAndCollectUserInformation is executed if available.

17.1.3 CancelFailed

17.1.3.1 General description

17.1.3.1.1 Error description

This Error is returned by Cancel if the canceling of an Operation, as requested by the SCF, was not successful. Possible failure reasons are:

0 unknownOperation, when the InvokeID of the operation to cancel is not known to SRF (this may also happen in case the operation has already been completed);

1 tooLate, when the invokeID is known but the execution of the operation is in a state that it cannot be canceled anymore. For instance the announcement is finished but the SpecializedResourceReport has not been sent to the SCF yet. The conditions for the occurrence of failure reason "tooLate" may be implementation dependent;

2 operationNotCancellable, when the invokeID points to an Operation that the SCF is not allowed to cancel.

17.1.3.1.2 Argument description

```
PARAMETER SEQUENCE {
    problem [0] ENUMERATED {
                unknownOperation (0),
                tooLate (1),
                operationNotCancellable (2)},
    operation   [1] InvokeID
}
-- The operation failed to be canceled.
```

17.1.3.2 Operations SCF->SSF

Cancel

17.1.3.3 Operations SCF->SRF

Cancel

Procedures at invoking entity (SCF)

A) Sending Cancel

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

The SCF sends a Cancel after a **PA** or PromptAndCollectUserInformation has been sent. The SCF remains in the same state.

B) Receiving CancelFailed Error

Precondition: SCSM state any SL dependent.

Postcondition: SCSM state any SL dependent.

After sending a Cancel operation the SL may continue (e.g., sending another PlayAnnouncement or PromptAndCollectUserInformation or a DisconnectForwardConnection). The CancelFailed Error can therefore be received in any state. The treatment is SL dependent.

Procedures at responding entity (SRF)

A) Receiving Cancel. However, the indicated PlayAnnouncement or PromptAndCollectUserInformation is not known, or already executed. This causes a failure, CancelFailed.

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

or SRSM state 1 Idle.

B) Sending CancelFailed Error

Precondition: SRSM state 3 User Interaction.

or SRSM state 1 Idle.

Postcondition: SRSM state 3 User Interaction.

or SRSM state 1 Idle.

After returning the CancelFailed the SRF stays in the same state.

17.1.4 DSAReferral

17.1.4.1 General description

17.1.4.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to chaining to a chained operation. The conditions under which a DSAReferral error is to be issued are defined in ITU-T Recommendation X.518 [40] subclause 8.3.

17.1.4.1.2 Argument description

The DSAReferral error parameter and problem codes are specified in ITU-T Recommendation X.518 [40] subclause 13.2.

17.1.4.2 Operations SDF->SDF

ChainedAddEntry

CheinedExecute
ChainedModifyEntry
ChainedRemoveEntry
ChainedSearch

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound

After receiving a DSAReferral error, the SDF can continue to execute the operation by accessing another SDF which is indicated by this error.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation due to a chaining condition and therefore sends a DSAReferral error to another SDF. After returning the error, no further error treatment is performed.

17.1.5 ETCFailed

17.1.5.1 General description

17.1.5.1.1 Error description

ETCFailed is an error from SSF to SCF, indicating the fact that the establishment of a temporary connection to an assisting SSF or SRF was not successful (e.g., receiving a "Backwards Release" after sending an IAM).

17.1.5.2 Operations SCF->SSF

EstablishTemporaryConnection

Procedures at invoking entity (SCF)

A) SCF sends EstablishTemporaryConnection to SSF

Precondition: SCSM state 3.1 Determine Mode.

Postcondition: SCSM state 3.2 Waiting for AssistRequestInstructions.

B) SCF receives ETCFailed Error from SSF

Precondition: SCSM state 3.2 Waiting for AssistRequestInstructions.

Postcondition: SCSM state 2.1 Preparing SSF Instructions.

Error handling depends on the SL, e.g., selecting another SRF or continue the processing of the call.

Procedures at responding entity (SSF)

A SSF receives EstablishTemporaryConnection from a SCF but the establishment of the connection fails, results in returning an ETCFailed Error to the SCF.

Precondition: SSF FSM state c Waiting for Instructions.

Postcondition: SSF FSM state c Waiting for Instructions.

No further Error treatment.

17.1.6 ExecutionError

17.1.6.1 General description

17.1.6.1.1 Error description

Execution Error is an error sent from the SDF to SCF, indicating that the request to execute an entry method has failed. The failure can be due to an incorrect input value or the failure of an internal operation or data access logic associated with the execution of the entry method.

17.1.6.1.2 Argument description

```
PARAMETER    OPTIONALLY-PROTECTED {
            SET {
                 problem [0] ExecutionProblem },
                COMPONENTS OF CommonResults},
            DIRQOP.&dirErrors-QOP{@dirqop} }
CODE        id-errcode-executionError }
ExecutionProblem:: = INTEGER {
    missingInputValues (1),
    executionFailure (2) }
```

17.1.6.2 Operations SCF->SDF

Execute

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition:

SCSM state 2 Wait for subsequent requests

or SCSM state 4 SDF Bound

Postcondition: SCSM state 2 Wait for subsequent requests

or SCSM state 4 SDF Bound

B) Receiving Error

Precondition:

SCSM state 4 SDF Bound

Postcondition:

SCSM state 4 SDF Bound

Error Procedure is independent of the SL. The service processing should be terminated.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition:

SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 2 Bind Pending

or SDF FSM state 3 SCF Bound

B) Returning Error

Precondition:

SDF FSM state 3 SCF Bound

Postcondition:

SDF FSM state 3 SCF Bound

In the case that the SDF has not received the correct input value, the SDF sends an ExecutionError to the SCF with problem set to missingInputValues. In all other cases the SDF attempts to execute the entry method but fails due to the failure of an internal operation or data access logic. The SDF therefore sends an ExecutionError to the SCF with problem set to executionFailure. After returning the error, the SDF is returned to the state it was prior to the operation, that is the execute operation is considered to be an atomic operation.

17.1.6.3 Operations SDF->SDF

ChainedExecute

Procedure at Invoking Entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests

Postcondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound

Postcondition: SDSM-ChI state 4 SDF Bound

This error is reported to the SCF in case of chained operations.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound

Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound

Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation and therefore sends an executionError to the other SDF. After returning the error, no further error treatment is performed.

17.1.7 ImproperCallerResponse

17.1.7.1 General description

17.1.7.1.1 Error description

The format of the user input has been checked by the SRF and does not correspond to the required format as it was defined in the initiating Operation.

17.1.7.2 Operations SCF->SRF

PromptAndCollectUserInformation

PromptAndReceiveMessage

Procedures at invoking entity (SCF)

A) SCF sends PromptAndCollectUserInformation to SRF

Precondition: SCSM state 3.1 Determine Mode; PromptAndCollectUserInformation will accompany the ConnectToResource.

or SCSM state 3.2 Waiting for AssistRequestInstructions; after EstablishTemporaryConnection.

or SCSM state 4.1 Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInformations are active.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

B) SCF receives ImproperCallerResponse Error from SRF

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

Error treatment depends on SL. A SCF can initiate new User Interaction or force a Disconnect (to SSF).

Procedures at responding entity (SRF)

A) SRF receives PromptAndCollectUserInformation

Precondition: SRSM state 2 Connected.

or SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

B) response from caller is not correct, SRF returns ImproperCallerResponse to SCF

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

SRF waits for a new Operation from SCF. This may be a new PromptAndCollectUserInformation or PlayAnnouncement.

17.1.7.3 Operations SCF->SCF

ChainedProvideUserInformation

ProvideUserInformation

Procedures at invoking entity (SCF)

A) Sending operation:

Precondition: SCSM-Sup in Assisting Mode (in case of ProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)

Postcondition: SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)

B) Receiving Error

Precondition: SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)

Postcondition: SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)

Once the supporting SCF or chaining terminator supporting SCF receives this type of error, it is waiting the reception of a SCFUnbind operation. A guard timer is armed.

Procedures at responding entity (SCF)

A) Receiving operation:

Precondition: SCSM-Con in Assisted Mode (in case of ProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)

Postcondition: SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)

B) Returning Error

Precondition: SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)

Postcondition: SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)

Once the controlling SCF issues this type of error, internal event (e12) SCFUnbind in SCSM-Con or event (e6) SCFUnbind in SCMS-ChI occurs.

17.1.8 MissingCustomerRecord

17.1.8.1 General description

17.1.8.1.1 Error description

This error is sent by the SSF to the SCF, in order to report the lack of the required user record within the SSP.

This error is also sent by the SCF to the SSF or the SRF, if the SLP could not be found in the SCF, because the required customer record does not exist, or the requested SLPI, indicated by the correlationID in "AssistRequestInstructions" does not exist anymore. These two cases should be distinguished as two different error situations, because the error procedure shows that the occurrence of the MissingCustomerRecord error is reported to the maintenance function, but the report to the maintenance function for the occurrence of the former case should be optional because it occurs not only in extraordinary situation but in ordinary situation. For example, the former may occur when the end user dials a missing free-phone number.

17.1.8.2 Operations SCF->SSF

Non Call Associated
ManageTriggerData

Refer to subclause 16.1.9 MissingParameter for the appropriate procedures.

17.1.8.3 Operations SSF->SCF

AssistRequestInstructions

InitialDP

Procedures at invoking entity (SSF)

A) Sending Operation

Precondition: SSF FSM state b Trigger processing.

or SSF FSM state b' Waiting for Instructions; in case of assist/hand-off.

Postcondition: SSF FSM state c Waiting for Instructions.

or SSF FSM state b' Waiting for Instructions; in case of assist/hand-off.

B) SSF receives Error "MissingCustomerRecord"

Precondition: SSF FSM state c Waiting for Instructions.

or SSF FSM state b' Waiting for Instructions; in case of assist/hand-off.

Postcondition: SSF FSM state a Idle.

or SSF FSM state a' Idle; in case of assist/hand-off.

The CCF routes the call if necessary (e.g., default routing to a terminating announcement).

Procedures at responding entity (SCF)

Precondition: (1) SCSM appropriate state.

(2) SCSM Operation received, appropriate event occurred.

Postcondition: (1) SCSM state 1 Idle; in case of all operations listed above, except AssistRequestInstructions.

(2) SCSM state 2.1 Preparing SSF instructions; for AssistRequestInstructions.

The SCSM detects that the required SLP does not exist. The SLPI may not exist anymore (e.g., in case of the operation AssistRequestInstructions), or the SLP may have never existed at all (i.e., the customer record in the SCF does not exist, e.g., in case of TDPs a SLP is attempted to be invoked). The Error parameter MissingCustomerRecord is used to inform the invoking entity of this situation. The maintenance functions are informed (however, it is optional for the TDP operation case).

17.1.8.4 Operations SRF->SCF

AssistRequestInstructions

Procedures at invoking entity (SRF)

A) Sending Operation

Precondition: SRSM state 2 Connected.

Postcondition: SRSM state 2 Connected.

B) SRF receives Error "MissingCustomerRecord"

Precondition: SRSM state 2 Connected.

Postcondition: SRSM state 1 Idle.

SRF initiated Disconnect.

Procedures at responding entity (SCF)

The SCSM detects that the required SLP does not exist (anymore). The establishment of a connection between the SSF and the SRF took too long or the correlationID was invalid. In both cases the requested SLP cannot be found. The Error parameter MissingCustomerRecord is used to inform the invoking entity of this situation. The maintenance functions are informed.

17.1.8.5 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

A) Sending operation

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting Mode (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in Assisting Mode (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChaiendProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

B) Receiving Error

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Idle (in case of first HandlingInformationRequest)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Error treatments in controlling SCF or chaining Initiator SCF depends on SL. No further error treatment is performed in supporting SCF of chaining terminator SCF.

Procedures at responding entity (SCF)

A) Receiving operation:

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted Mode (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in Assisted Mode (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in Assisted Mode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Returning Error

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Error treatments in controlling SCF or chaining Initiator SCF depends on SL. No further error treatment is performed in supporting SCF of chaining terminator SCF.

17.1.8.6 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

A) Sending Operation

Precondition: CUSF FSM state a Idle.

Postcondition: CUSF FSM state b Waiting for Instructions.

B) CUSF receives Error "MissingCustomerRecord"

Precondition: CUSF FSM state b Waiting for Instructions.

Postcondition: CUSF FSM state a Idle.

The CUSF continues to handle the association or terminate the association with default procedures (network operator specific).

Procedures at responding entity (SCF)

Precondition: (1) FSM for CUSF appropriate state (in State N1 Idle).

(2) FSM for CUSF Operation received, appropriate event occurred.

Postcondition: (1) FSM for CUSF state N1 Idle.

The FSM for CUSF detects that the required SLP does not exist. This is the same situation as for the SSF-SCF case.

17.1.9 MissingParameter

17.1.9.1 General description

17.1.9.1.1 Error description

There is an Error in the received Operation argument. The responding entity cannot start to process the requested Operation because the argument is incorrect: a mandatory parameter (the application shall always return this error in case it is not detected by the ASN.1 decoder) or an expected optional parameter which is essential for the application is not included in the Operation argument.

17.1.9.2 Operations SCF->SSF

Non Call Associated

ActivateServiceFiltering

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

Cancel

FurnishChargingInformation

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportUTSI

ResetTimer

SendChargingInformation

SendSTUI

Call Associated/Call Processing

CollectInformation

Connect

ConnectToResource

ContinueWithArgument

CreateCallSegmentAssociation

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

InitiateCallAttempt

MergeCallSegments

MoveCallSegments

MoveLeg

SplitLeg

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM any state in which the above Call associated operations can be transferred.

SCME any state in which the above Non call associated operations can transferred.

Postcondition: SCSM any state as result of the transfer of any of the above operations.

SCME any state as result of the transfer of any of the above Non call associated operations.

B) SCF receives Error "MissingParameter"

Precondition: SCSM any state as result of the transfer of any of the above Call associated operations.

SCME any state as result of the transfer of any of the above Non call associated operations.

Postcondition: SCSM transition to the initial state (i.e., before sending the erroneous operation).

SCME transition to the initial state (i.e., before sending the erroneous operation).

The SL and maintenance functions are informed. Further treatment of the call is dependent on SL.

Procedures at responding entity (SSF)

Precondition: (1) SSF FSM appropriate state.

(2) SSF FSM Call associated operation received, appropriate event occurred.

(3) SSME appropriate state.

(4) SSME Non call associated operation received, appropriate event.

Postcondition: (1) SSF FSM transition to the same state.

(2) SSME transition to the initial state (i.e., before receiving the erroneous operation).

The SSF FSM detects the error in the received operation. The Error parameter is returned to inform the SCF of this situation.

17.1.9.3 Operations SSF->SCF

ApplyChargingReport

AssistRequestInstructions

InitialDP

Procedures at invoking entity (SSF)

A) Sending Operation

Precondition: SSF FSM any state in which the above operations can be transferred.

Postcondition: SSF FSM any state as result of the transfer of any of the above operations.

B) SSF receives Error "MissingParameter"

Precondition: SSF FSM any state as result of the transfer of any of the above operations.

Postcondition: SSF FSM state a Idle.

After receiving this Error, the SSF FSM returns to the state Idle. The CCF routes the call if necessary (default routing to a terminating announcement). If the call is already established (i.e., mid-call trigger or ApplyChargingReport), the CCF may maintain the call or disconnect it. The choice between these two options is network operator specific. In case of an assisting SSF, the temporary connection is released by the assisting SSF.
Procedures at responding entity (SCF)

Precondition: (1) SCSM appropriate state.

(2) SCSM Operation received, appropriate event occurred.

Postcondition: (1) SCSM state 1 Idle; in case of any operation listed above, except AssistRequestInstructions.

or (2) SCSM state 2.1 Preparing SSF Instructions; in case of AssistRequestInstructions.

The SCSM detects the erroneous situation. The Error parameter is used to inform the SSF of this situation. The SL and maintenance functions are informed.

17.1.9.4 Operations SCF->SRF

Cancel

ScriptClose

ScriptInformation

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptRun

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 3.1 Determine Mode; PromptAndCollectUserInformation or PlayAnnouncement will accompany the ConnectToResource.

or SCSM state 3.2 Waiting for AssistRequestInstructions; after EstablishTemporaryConnection.

or SCSM state 4.1 Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInformations are outstanding.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

B) Receiving Error

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

Error treatment depends on SL. SCF can initiate new User Interaction or force Disconnect (to SSF).

Procedures at responding entity (SRF)

Precondition: SRSM state 2 Connected.

or SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

The SRSM detects that a required parameter is not present in the Operation argument. The Error parameter MissingParameter is used to inform the SCF of this situation. The SCF should take the appropriate actions to treat this error.

17.1.9.5 Operations SRF->SCF

AssistRequestInstructions

Procedures at invoking entity (SRF)

A) Sending Operation

Precondition: SRSM state 2 Connected.

Postcondition: SRSM state 2 Connected.

B) Receiving Error

Precondition: SRSM state 2 Connected.

Postcondition: SRSM state 1 Idle.

Procedures at responding entity (SCF)

Precondition: SCSM state 3.2 Waiting for AssistRequestInstructions.

Postcondition: SCSM state 2.1 Preparing SSF instructions.

The SCSM detects the error in the received operation. The Error parameter is used to inform the SRF of this situation. The SL and maintenance functions are informed. The SCF might try another SRF, route the call or release the call (SL dependent).

17.1.9.6 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

A) Sending operation:

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state Assisting Mode (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in Assisting Mode (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChaiendProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state WaitingForBindResult (in case of HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

Receiving Error

Precondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Idle (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Error treatments in controlling SCF or chaining Initiator SCF depends on SL. No further error treatment is performed in supporting SCF of chaining terminator SCF.

Procedures at responding entity (SCF)

A) Receiving operation:

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state ProcessingSCFBind (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state Assisted Mode (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in Assisted Mode (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in Assisted Mode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-Con in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Returning Error

Precondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Error treatments in controlling SCF or chaining Initiator SCF depends on SL. No further error treatment is performed in supporting SCF of chaining terminator SCF.

17.1.9.7 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

RequestReportUTSI

SendSTUI

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: FSM for CUSF state N1 Idle

or FSM for CUSF state N2.1 Preparing CUSF Instructions.

Postcondition: FSM for CUSF state N2.1 Preparing CUSF Instructions

or FSM for CUSF state N2.2 Waiting for Notification or Request.

B) SCF receives Error "MissingParameter"

Precondition: FSM for CUSF state N2.1 Preparing CUSF Instructions

or FSM for CUSF state N2.2 Waiting for Notification or Request.

Postcondition: FSM for CUSF transition to the initial state (i.e., before sending the erroneous operation).

The SL and maintenance functions are informed. Further treatment of the call is dependent on SL.

Procedures at responding entity (CUSF)

Precondition: (1) CUSF FSM any appropriate state.

(2) CUSF FSM an operation received, appropriate event occurred.

Postcondition: (1) CUSF FSM transition to the same state.

The CUSF FSM detects the error in the received operation. The Error parameter is returned to inform the SCF of this situation.

17.1.9.8 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

A) Sending Operation

Precondition: CUSF FSM state a Idle

or CUSF FSM state b Waiting for Instructions

or CUSF FSM state c Monitoring.

Postcondition: CUSF FSM state a Idle

or CUSF FSM state b Waiting for Instructions

or CUSF FSM state c Monitoring.

B) CUSF receives Error "MissingParameter"

Precondition: CUSF FSM state a Idle

or CUSF FSM state b Waiting for Instructions

or CUSF FSM state c Monitoring.

Postcondition: CUSF FSM state a Idle.

After receiving this Error, the CUSF FSM returns to the state Idle. The CUSF terminates the association if necessary. If the supplementary service is already active and ready for responding, the CUSF may maintain the association and continue service processing. The choice between these two options is network operator specific.

Procedures at responding entity (SCF)

Precondition: (1) FSM for CUSF any appropriate state.

(2) FSM for CUSF Operation received, appropriate event occurred.

Postcondition: (1) FSM for CUSF state N1 Idle; in case of any operation listed above,

The FSM for CUSF detects the erroneous situation. The Error parameter is used to inform the CUSF of this situation. The SL and maintenance functions are informed.

17.1.10 NameError

17.1.10.1 General description

17.1.10.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report an problem related to the name of the object. The conditions under which a name error is to be issued are defined in ITU-T Recommendation X.511 [39] subclause 12.5.

17.1.10.1.2 Argument description

The name error parameter and problem codes are specified in ITU-T Recommendation X.511 [39] subclause 12.5.

17.1.10.2 Operations SCF->SDF

Execute

AddEntry

ModifyEntry

RemoveEntry

Search

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 4 SDF Bound or

SCSM state 2 Wait for subsequent requests

Postcondition: SCSM state 4 SDF Bound or

SCSM state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SCSM state 4 SDF Bound

Postcondition: SCSM state 4 SDF Bound

Error Procedure is dependent on the SL. If the SCF is able to change the request, it can do another SDF query, otherwise the service processing should be terminated.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

B) Returning Error

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

The SDF could not perform the operation due to a name problem and therefore sends an Name error to the SCF. After returning the error, no further error treatment is performed.

17.1.10.3 Operations SDF->SDF

ChainedAddEntry

ChainedExecute

ChainedModifyEntry

ChainedRemoveEntry

ChainedSearch

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound

This error is reported to the SCF

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation due to a name problem and therefore sends an Name error to the other SDF. After returning the error, no further error treatment is performed.

17.1.11 ParameterOutOfRange

17.1.11.1 General description

17.1.11.1.1 Error description

The responding entity cannot start the processing of the requested Operation because an Error in a parameter of the Operation argument is detected: a parameter value is out of range. This error is applied for the following two cases (when the error is determined by the application):

(1) For the parameter which type is defined with the range of its size, such as INTEGER(x..y), SEQUENCE SIZE(x..y) OF Type. This error is applied when the parameter value is z or the parameter size is z where z<x or z>y.

(2) For the parameter which type is defined as list of ENUMERATED value, the ParameterOutOfRange error is applied when the parameter value is not equal to any of the ENUMERATED values in the list.

17.1.11.2 Operations SCF->SSF

Non Call Associated

ActivateServiceFiltering

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

SendChargingInformation

SendSTUI

Call Associated/Call Processing

Connect

InitiateCallAttempt

RequestNotificationChargingEvent

RequestReportBCSMEvent

ResetTimer

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.11.3 Operations SSF->SCF

ApplyChargingReport

InitialDP

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.11.4 Operations SCF->SRF

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.11.5 Operations SCF -> SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.11.6 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

SendSTUI

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.11.7 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.12 Referral

17.1.12.1 General description

17.1.12.1.1 Error description

This error is sent by the SDF to the SCF to report a problem related to service data location. The conditions under which a referral error is to be issued are defined in ITU-T Recommendation X.511 [39] subclause 12.6.

17.1.12.1.2 Argument description

The referral error parameter and problem codes are specified in ITU-T Recommendation X.511 [39] subclause 12.6.

17.1.12.2 Operations SCF->SDF

AddEntry

Execute

ModifyEntry

RemoveEntry

Search

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 4 SDF Bound or
SCSM state 2 Wait for subsequent requests
Postcondition: SCSM state 4 SDF Bound or
SCSM state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SCSM state 4 SDF Bound
Postcondition: SCSM state 4 SDF Bound

After receiving a referral error, the SCF can continue to execute the SL by accessing another SDF which is indicated by this error.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 3 SCF Bound
Postcondition: SDF FSM state 3 SCF Bound

B) Returning Error

Precondition: SDF FSM state 3 SCF Bound
Postcondition: SDF FSM state 3 SCF Bound

The SDF could not perform the operation due to a data location and therefore sends a Referral error to the SCF. After returning the error, no further error treatment is performed.

17.1.13 RequestedInfoError

17.1.13.1 General description

17.1.13.1.1 Error description

The RequestedInfoError is an immediate response to the CallInformationRequest operation, indicating that the requested information is not known to the SSF or is not available. RequestedInfoError is used when a specific SSF/CCF can not offer the information specified with RequestedInformationType but there exists other SSF/CCF that can offer the information.

17.1.13.1.2 Argument description

```
PARAMETER ENUMERATED {
    unknownRequestedInfo(1),
    requestedInfoNotAvailable(2)
    -- other values not specified
    }
```

17.1.13.2 Operations SCF->SSF

CallInformationRequest

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.14 ScfReferral

17.1.14.1 General description

17.1.14.1.1 Error description

This error is sent by the supporting SCF to the controlling SCF or by the chaining terminator to the chaining initiator, in order to report that it is not able to provide the assistance and to provide the address of another SCF

17.1.14.1.2 Argument description

The scfReferral error is defined through the AccessPointInformation parameter specified in ITU-T Recommendation X.511 [39].

17.1.14.2 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

HandlingInformationRequest

Procedures at invoking entity (controlling SCF or supporting chaining initiator SCF)

A) Sending operation

Precondition: SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest
or SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest

Postcondition: SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest

B) Receiving Error

Precondition: SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest

Postcondition: SCSM-Con in state Idle (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state SCFBound (in case of subsequent ChainedHandlingInformationRequest

After receiving the SCFreferral error, the SCF can request assistance to another SCF.

Procedures at responding entity (SCF)

A) Receiving operation

Precondition: SCSM-Sup in state ProcessingSCFBind (in case of first HandlingInformationRequest)
or SCSM-SUp in state Assisting Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest)

Postcondition: SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest)

B) Returning Error

Precondition: SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of subsequent ChanedHandlingInformationRequest)

Postcondition: SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-SUp in state Assisting Mode (in case of subsequent HandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of subsequent ChainedHandlingInformationRequest)

After returning the error, the assisting SCF is waiting for SCFUnbind operation. No further error treatments takes place. If a guard timer expires event (e22) in SCSM-Sup or (e6) in SCSM-ChT will occur.

17.1.15 Security

17.1.15.1 General description

17.1.15.1.1 Error description

This error is sent by the SCF/SDF to the SCF/SDF to report a problem in carrying out an operation for security reasons. The conditions under which a security error is to be issued are defined in ITU-T Recommendation X.511 [39] subclause 12.7.

17.1.15.1.2 Argument description

The security error parameter and problem codes are specified in ITU-T Recommendation X.511 [39] subclause 12.7.

17.1.15.2 Operations SCF->SDF

Execute

AddEntry

ModifyEntry

RemoveEntry

Search

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition:

SCSM state 2 Wait for subsequent

or SCSM state 4 SDF Bound

Postcondition: SCSM state 2 Wait for subsequent requests

or SCSM state 4 SDF Bound

B) Receiving Error

Precondition: SCSM state 3 Wait for Bind result

or SCSM state 4 SDF Bound

Postcondition: SCSM state 4 SDF Bound

Error Procedure is independent of the SL. The service processing should be terminated.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 1 Idle

or SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 2 Bind Pending

or SDF FSM state 3 SCF Bound

B) Returning Error

Precondition: SDF FSM state 2 Bind Pending

or SDF FSM state 3 SCF Bound

Postcondition:

SDF FSM state 3 SCF Bound

The SDF could not perform the operation for security reasons and therefore sends a Security error to the SCF. After returning the error, no further error treatment is performed.

17.1.15.3 Operations SDF->SDF

ChainedAddEntry

ChainedExecute

ChainedModifyEntry

ChainedRemoveEntry

ChainedSearch

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests
B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound

This error is reported to the SCF in case of chained operations.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound

Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation for security reasons and therefore sends a Security error to the other SDF. After returning the error, no further error treatment is performed.

17.1.15.4 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

A) Sending operation

Precondition: SCSM-Con in state Idle (in case of SCFBind)
or SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Preparing Request for Assistance (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state Assisting Mode (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in Assisting Mode (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
SCSM-ChI in state Idle (in case of SCFBind used in the chaining)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChaiendProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Con in state PreparingRequestForAssistance (in case of SCFBind)
or SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
SCSM-ChI in state PrepareChainedHIReq (in case of SCFBind used in the chaining)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

B) Receiving Error

Precondition: SCSM-Con in state PreparingRequestForAssistance (in case of SCFBind)
or SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state WaitingForBindResult (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
SCSM-ChI in state PrepareChainedHIReq (in case of SCFBind used in the chaining)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Con in state Idle (in case of SCFBind)
or SCSM-Sup in state Assisting Mode (in case of EstablishChargingRecord)
or SCSM-Con in state Idle (in case of first HandlingInformationRequest)
or SCSM-Sup in state Assisting mode (in case of HandlingInformationResult)
or SCSM-Sup in state WaitingForAdditionalInformation (in case of NetworkCapability)
or SCSM-Con in state Assisted Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Sup in WaitingForAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Con in Assisted Mode (in case of ReportChargingInformation)
or SCSM-Sup in AssistingMode (in case of RequestNotification)
SCSM-ChI in state Idle (in case of SCFBind used in the chaining)
or SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Once the security error is reported to the controlling SCF or chaining initiator internal logic, the need to send a SCFUnbind operation is determined.

Once the supporting SCF or the chaining terminator SCF receives a security error, it expects to receive an SCFUnbind operation. This event is guarded by a expiration timer.

Procedures at responding entity (SCF)

A) Receiving operation

Precondition: SCSM-Sup in state Idle (in case of SCFBind)
or SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state ProcessingSCFBind (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state Assisted Mode (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in Assisted Mode (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in Assisted Mode (in case of RequestNotification)
SCSM-ChT in state Idle (in case of SCFBind used in the chaining)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-Con in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Sup in state SCFBindRequestReceived (in case of SCFBind)
or SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state ProcessSCFBind (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
SCSM-ChT in state BindPending (in case of SCFBind used in the chaining)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

B) Returning Error

Precondition: SCSM-Sup in state SCFBindRequestReceived (in case of SCFBind)
or SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
SCSM-ChT in state BindPending (in case of SCFBind used in the chaining)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-Sup in state Idle (in case of SCFBind)
or SCSM-Con in state Assisted Mode (in case of EstablishChargingRecord)
or SCSM-Sup in state AssistingMode (in case of first HandlingInformationRequest)
or SCSM-Con in state Assisted mode (in case of HandlingInformationResult)
or SCSM-Con in state PreparingAdditionalInformation (in case of NetworkCapability)
or SCSM-Sup in state Assisting Mode (in case of NotificationProvided or subsequent HandlingInformationRequest)
or SCSM-Con in PreparingAdditionalInformation (in case of ProvideUserInformation)
or SCSM-Sup in Assisting Mode (in case of ReportChargingInformation)
or SCSM-Con in AssistedMode (in case of RequestNotification)
SCSM-ChT in state Idle (in case of SCFBind used in the chaining)
or SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

The security error is reported to the controlling SCF internal logic and the need to send a SCFUnbind operation is determined.

Once the supporting SCF or the chaining terminator SCF receives a security error, if this event does not causes a transition to idle, it expects to receive an SCFUnbind operation. This event is guarded by a expiration timer.

17.1.16 Service

17.1.16.1 General description

17.1.16.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report a problem related to the provision of the service. The conditions under which a service error is to be issued are defined in ITU-T Recommendation X.511 [39] subclause 12.8.

17.1.16.1.2 Argument description

The Service error parameter and problem codes are specified in ITU-T Recommendation X.511 [39] subclause 12.8.

17.1.16.2 Operations SCF->SDF

AddEntry

Execute

ModifyEntry

RemoveEntry

Search

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 2 Wait for subsequent requests

or SCSM state 4 SDF Bound

Postcondition: SCSM state 2 Wait for subsequent requests

or SCSM state 4 SDF Bound

B) Receiving Error

Precondition: SCSM state 4 SDF Bound

Postcondition: SCSM state 4 SDF Bound

Error Procedure is dependent on the SL. If there is an alternative SDF it may be possible to do another query, otherwise the service processing should be terminated.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

B) Returning Error

Precondition: SDF FSM state 2 Bind Pending (in case of Bind)

or SDF FSM state 3 SCF Bound (in case of operations except Bind)

Postcondition: SDF FSM state 1 Idle (in case of Bind)

or SDF FSM state 3 SCF Bound (in case of operations except Bind)

The SDF could not perform the operation due to a service related problem and sends a Service error to the SCF. After returning the error, no further error treatment is performed.

17.1.16.3 Operations SDF->SDF

ChainedAddEntry

ChainedExecute

ChainedModifyEntry

ChainedRemoveEntry

ChainedSearch

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound

Postcondition: SDSM-ChI state 4 SDF Bound

This error is reported to the SCF in case of chained operations.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound

Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation due to a service related problem and sends a Service error to the other SDF. After returning the error, no further error treatment is performed.

17.1.17 Shadow

17.1.17.1 General description

17.1.17.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to shadowing. The conditions under which a shadow error is to be issued are defined in ITU-T Recommendation X.525 [42] subclause 12.

17.1.17.1.2 Argument description

The shadow error parameter and problem codes are specified in ITU-T Recommendation X.525 [42] subclause 11.3.3.

17.1.17.2 Operations SDF->SDF

CoordinateShadowUpdate

RequestShadowUpdate

UpdateShadow

Procedures at Supplier Entity (SDF)

A-1) Sending Operation

Precondition: SDSM-ShM (in case of supplier initiated)
state 4 SDF Bound (in case of coordinateShadowUpdate) or
state 6 Wait for update (in case of UpdateShadow)
SDSM-ShM (in case of consumer initiated)
state 5 Wait for update (in case of UpdateShadow)
Postcondition: SDSM-ShM (in case of supplier initiated)
state 5 Wait for coordination result (in case of coordinateShadowUpdate) or
state 7 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShM (in case of consumer initiated)
state 6 Wait for update confirmation (in case of UpdateShadow)

B-1) Receiving Error

Precondition: SDSM-ShM (in case of supplier initiated)
state 5 Wait for coordination result (in case of coordinateShadowUpdate) or
state 7 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShM (in case of consumer initiated)
state 6 Wait for update confirmation (in case of UpdateShadow)
Postcondition: SDSM-ShM (in case of supplier initiated)
state 4 SDF Bound (in case of coordinateShadowUpdate) or
state 6 Wait for update (in case of UpdateShadow)
SDSM-ShM (in case of consumer initiated)
state 5 Wait for update (in case of UpdateShadow)

A-2) Receiving Operation

Precondition: SDSM-ShM (in case of consumer initiated)
state 3 SDF Bound (in case of requestShadowUpdate)
Postcondition: SDSM-ShM (in case of consumer initiated)
state 4 Wait for RequestShadow result (in case of requestShadowUpdate)

B-2) Returning Error

Precondition: SDSM-ShM (in case of consumer initiated)
state 4 Wait for RequestShadow result (in case of requestShadowUpdate)
Postcondition: SDSM-ShM (in case of consumer initiated)
state 3 SDF Bound (in case of requestShadowUpdate)

After receiving or returning the error, no further error treatment is performed.

Procedures at Consumer Entity (SDF)

A-1) Sending Operation

Precondition: SDSM-ShC (in case of consumer initiated)
state 4 SDF Bound (in case of requestShadowUpdate)
Postcondition: SDSM-ShC (in case of consumer initiated)
state 5 Wait for RequestShadow result (in case of requestShadowUpdate)

B-1) Receiving Error

Precondition: SDSM-ShC (in case of requestShadowUpdate)
state 5 Wait for RequestShadow result (in case of requestShadowUpdate)
Postcondition: SDSM-ShC (in case of requestShadowUpdate)
state 4 SDF Bound (in case of requestShadowUpdate)

A-2) Receiving Operation

Precondition: SDSM-ShC (in case of supplier initiated)
state 3 SDF Bound (in case of coordinateShadowUpdate) or
state 5 Wait for update (in case of UpdateShadow)
SDSM-ShC (in case of consumer initiated)
state 6 Wait for update (in case of UpdateShadow)
Postcondition: SDSM-ShC (in case of supplier initiated)
state 4 Wait for coordination result (in case of CoordinateShadowUPdate) or
state 6 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShC (in case of consumer initiated)
state 7 Wait for update confirmation (in case of UpdateShadow)

B-2) Returning Error

Precondition: SDSM-ShC (in case of supplier initiated)
state 4 Wait for coordination result (in case of CoordinateShadowUPdate) or
state 6 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShC (in case of consumer initiated)
state 7 Wait for update confirmation (in case of UpdateShadow)
Postcondition: SDSM-ShC (in case of supplier initiated)
state 3 SDF Bound (in case of coordinateShadowUpdate) or
state 5 Wait for update (in case of UpdateShadow)
SDSM-ShC (in case of consumer initiated)
state 6 Wait for update (in case of UpdateShadow)

After receiving or returning the error, no further error treatment is performed.

17.1.18 SystemFailure

17.1.18.1 General description

17.1.18.1.1 Error description

This error is returned by a PE if it was not able to fulfill a specific task as requested by an operation, and recovery is not expected to be completed within the current call instance.

17.1.18.2 Argument description

PARAMETER

UnavailableNetworkResource

UnavailableNetworkResource:: = ENUMERATED {

unavailableResources (0),

componentFailure (1),

basicCallProcessingException (2),

resourceStatusFailure (3),

endUserFailure (4)}

16.1.18.2 Operations SCF->SSF

Non Call Associated

ActivateServiceFiltering

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportUTSI

SendChargingInformation

SendSTUI

Call Associated/Call Processing

CollectInformation

Connect

ConnectToResource

CreateCallSegmentAssociation

DisconnectForwardConnection

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

InitiateCallAttempt

MergeCallSegments

MoveCallSegments

MoveLeg

SplitLeg

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.18.3 Operations SSF->SCF

ApplyChargingReport

AssistRequestInstructions

InitialDP

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.18.4 Operations SCF->SRF

ScriptClose

ScriptInformation

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptRun

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.18.5 Operations SRF->SCF

AssistRequestInstructions

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.18.6 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.18.7 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

RequestReportUTSI

SendSTUI

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.18.8 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.19 TaskRefused

17.1.19.1 General introduction

17.1.19.1.1 Error description

This Error is returned by a PE if it was not able to fulfill a specific task as requested by an operation, and recovery is expected to be completed within the current call instance.

17.1.19.1.2 Argument description

```
PARAMETER ENUMERATED {
    generic(0),
    unobtainable (1),
    congestion (2)
    }
```

17.1.19.2 Operations SCF->SSF

Non Call Associated

ActivateServiceFiltering

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

Cancel

FurnishChargingInformation

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportUTSI

ResetTimer

SendChargingInformation

SendSTUI

Call Associated/Call Processing

CollectInformation

Connect

ConnectToResource

CreateCallSegmentAssociation

DisconnectForwardConnection

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

InitiateCallAttempt

MergeCallSegments

MoveCallSegments

MoveLeg

SplitLeg

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.19.3 Operations SSF->SCF

ApplyChargingReport

AssistRequestInstructions

InitialDP

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.19.4 Operations SCF->SRF

Cancel

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptClose

ScriptInformation

ScriptRun

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.19.5 Operations SRF->SCF

AssistRequestInstructions

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.19.6 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

RequestReportUTSI

SendSTUI

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.19.7 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.20 UnavailableResource

17.1.20.1 General description

17.1.20.1.1 Error description

The SRF is not able to perform its function (i.e., play a certain announcement and/or collect specific user information), and cannot be replaced. A reattempt is not possible.

17.1.20.2 Operations SCF->SRF

ScriptClose

ScriptInformation

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptRun

Procedures at invoking entity (SCF)

A) SCF sends PlayAnnouncement or PromptAndCollectUserInformation to SRF

Precondition: SCSM state 3.1 Determine Mode; PlayAnnouncement or PromptAndCollectUserInformation will accompany the ConnectToResource.

or SCSM state 3.2 Waiting for AssistRequestInstructions; after EstablishTemporaryConnection.

or SCSM state 4.1 Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInformations are outstanding.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

B) SCF receives UnavailableResource Error from SRF

Precondition: SCSM state 4.1 Waiting for Response from the SRF.

Postcondition: SCSM state 4.1 Waiting for Response from the SRF.

If the chosen resource cannot perform its function the further treatment is service dependent.

EXAMPLES:
- request SSF to connect to alternative SRF;
- service processing without PlayAnnouncement or PromptAndCollectUserInformation (if possible);
- terminate service processing.

Procedures at responding entity (SRF)

A) SRF receiving PlayAnnouncement or PromptAndCollectUserInformation

Precondition: SRSM state 2 Connected; if initial PlayAnnouncement or PromptAndCollectUserInformation.

or SRSM state 3 User Interaction; if not initial PlayAnnouncement or

PromptAndCollectUserInformation.

B) SRF is not able to perform its function (and cannot be replaced). SRF sends UnavailableResource.

Precondition: SRSM state 3 User Interaction.

Postcondition: SRSM state 3 User Interaction.

17.1.21 UnexpectedComponentSequence

17.1.21.1 General description

17.1.21.1.1 Error description

The responding entity cannot start the processing of the requested operation because a SACF or MACF rule is violated, or the operation could not be processed in the current state of the FSM.

17.1.21.2 Operations SCF->SSF

Non Call Associated

ActivateServiceFiltering

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

FurnishChargingInformation

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportUTSI

ResetTimer

SendChargingInformation

SendSTUI

Call Associated/Call Processing

CollectInformation

Connect

ConnectToResource

ContinueWithArgument

CreateCallSegmentAssociation

DisconnectForwardConnection

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

InitiateCallAttempt

MergeCallSegments

MoveCallSegments

MoveLeg

SplitLeg

In this case the SSF detects the erroneous situation, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF the SL and maintenance functions are informed and the SL decides about error treatment.

17.1.21.3 Operations SSF->SCF

ApplyChargingReport

AssistRequestInstructions

InitialDP

In case of assisting SSF an error occurs in case an AssistRequestInstructions is sent while a relationship between SCF and assisting SSF has already been established, the SCF returns the error parameter. SL and maintenance are informed. On receiving the error the assisting SSF moves to Idle and the temporary connection is released.

In case the operation is sent by an "initiating" SSF in the context of an existing relationship, the SCF returns the error parameter. SL and maintenance are informed. On receiving the error the SSF moves to Idle.

17.1.21.4 Operations SCF->SRF (only applicable for direct SCF-SRF case)

ScriptClose

ScriptInformation

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptRun

In this case the SRF detects the erroneous situation, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF, the SL and maintenance functions are informed and the SL decides about error treatment. Possible error treatment is to send the DisconnectForwardConnection operation to the SSF.

17.1.21.5 Operations SRF->SCF

AssistRequestInstructions

In this case, an error occurs if the SRF has already an established relationship with the SCF and sends an AssistRequestInstructions. The SCF detects the erroneous situation, informs SL and maintenance functions and returns the error parameter. On receiving the parameter the SRF moves to idle and releases the temporary connection.

17.1.21.6 Operations SCF->SCF

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)
Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.21.7 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

RequestReportUTSI

SendSTUI

Procedures at invoking entity (SCF) and responding entity (CUSF)
The CUSF detects the error, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF, the SL and maintenance functions are informed and the SL determines the appropriate error treatment.

17.1.21.8 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF) and responding entity (SCF)

In case the operation is sent by CUSF, the SCF returns the error parameter. SL and maintenance functions are informed, and the SL determines the appropriate error treatment. On receiving the error the CUSF moves to Idle.

17.1.22 UnexpectedDataValue

17.1.22.1 General description

17.1.22.1.1 Error description

The responding entity cannot complete the processing of the requested Operation because a parameter has an unexpected data value.

Note that this error does not overlap with "ParameterOutOfRange"

EXAMPLE: startTime DateAndTime:: = -- value indicating January 32 1993, 12: 15: 01

The responding entity does not expect this value and responds with "UnexpectedDataValue".

17.1.22.2 Operations SCF->SSF

Non Call Associated

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

FurnishChargingInformation

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportUTSI

ResetTimer

SendSTUI

Call Associated/Call Processing

CollectInformation

Connect

ConnectToResource

ContinueWithArgument

CreateCallSegmentAssociation

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

InitiateCallAttempt

MergeCallSegments

MoveCallSegments

MoveLeg

SplitLeg

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.22.3 Operations SSF->SCF

ApplyChargingReport

AssistRequestInstructions

InitialDP

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.22.4 Operations SCF->SRF

ScriptClose

ScriptInformation

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptRun

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.22.5 Operations SRF->SCF

AssistRequestInstructions

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.22.6 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.22.7 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

RequestReportUTSI

SendSTUI

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.22.8 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23 UnexpectedParameter

17.1.23.1 General description

17.1.23.1.1 Error description

There is an error in the received Operation argument. A valid but unexpected parameter was present in the Operation argument. The presence of this parameter is not consistent with the presence of the other parameters. The responding entity cannot start to process the Operation.

17.1.23.2 Operations SCF->SSF

Non Call Associated

ActivateServiceFiltering

ManageTriggerData

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

FurnishChargingInformation

RequestNotificationChargingEvent

RequestReportBCSMEvent

RequestReportUTSI

ResetTimer

SendChargingInformation

SendSTUI

Call Associated/Call Processing

CollectInformation

Connect

ConnectToResource

ContinueWithArgument

CreateCallSegmentAssociation

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

InitiateCallAttempt

MergeCallSegments

MoveCallSegments

MoveLeg

SplitLeg

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23.3 Operations SSF->SCF

ApplyChargingReport

AssistRequestInstructions

InitialDP

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23.4 Operations SCF->SRF

ScriptClose

ScriptInformation

PlayAnnouncement

PromptAndCollectUserInformation

PromptAndReceiveMessage

ScriptRun

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23.5 Operations SRF->SCF

AssistRequestInstructions

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23.6 Operations SCF->SCF

ChainedConfirmedNotificationProvided

ChainedConfirmedReportChargingInformation

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

ConfirmedNotificationProvided

ConfirmedReportChargingInformation

EstablishChargingRecord

HandlingInformationRequest

HandlingInformationResult

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23.7 Operations SCF-> CUSF

ConnectAssociation

ContinueAssociation

InitiateAssociation

RequestReportBCUSMEvent

RequestReportUTSI

SendSTUI

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.23.8 Operations CUSF -> SCF

InitialAssociationDP

Procedures at invoking entity (CUSF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.24 UnknownLegID

17.1.24.1 General description

17.1.24.1.1 Error description

This error is used to indicate to the SCF that a specific leg, indicated by the LegID parameter value in the operation, is unknown to the SSF.

17.1.24.2 Operations SCF->SSF

Call Associated/Non Call Processing

ApplyCharging

CallInformationRequest

SendChargingInformation

SendSTUI

Call Associated/Call Processing

ConnectToResource

ContinueWithArgument

DisconnectForwardConnectionWithArgument

DisconnectLeg

EstablishTemporaryConnection

MoveCallSegments

MoveLeg

PlayAnnouncement

SplitLeg

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.24.3 Operations SCF->CUSF

Non Call Processing

SendSTUI

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.1.25 Update

17.1.25.1 General description

17.1.25.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report a problem related to attempts to add, delete, or modify information in the SDF. The conditions under which an update error is to be issued are defined in ITU-T Recommendation X.511 [39] subclause 12.9.

17.1.25.1.2 Argument description

The update error parameter and problem codes are specified in ITU-T Recommendation X.511 [39] subclause 12.9.

17.1.25.2 Operations SCF->SDF

AddEntry

Execute

ModifyEntry

RemoveEntry

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 4 SDF Bound or

SCSM state 2 Wait for subsequent requests

Postcondition: SCSM state 4 SDF Bound or

SCSM state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SCSM state 4 SDF Bound

Postcondition: SCSM state 4 SDF Bound

Error Procedure is dependent on the SL.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

B) Returning Error

Precondition: SDF FSM state 3 SCF Bound

Postcondition: SDF FSM state 3 SCF Bound

The SDF could not perform the operation due to a problem related to the addition, deletion or modification of information and sends an Update error to the SCF. After returning the error, no further error treatment is performed.

17.1.25.3 Operations SDF->SDF

ChainedAddEntry

ChainedExecute

ChainedModifyEntry

ChainedRemoveEntry

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests

B) Receiving Error

Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound

This error is reported to the SCF.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

B) Returning Error

Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

The SDF could not perform the operation due to a problem related to the addition, deletion or modification of information and sends an Update error to the other SDF. After returning the error, no further error treatment is performed.

17.1.26 ChainingRefused

17.1.26.1 General description

17.1.26.1.1 Error description

This error is sent by the chaining terminator (or initiator) supporting to the initiator (or terminator) supporting SCF to reject a chained operation.

17.1.26.1.2 Argument description

No parameters

17.1.26.2 Operations SCF->SCF

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedHandlingInformationResult

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

Procedures at invoking entity (SCF)

A) Sending operation

Precondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state PrepareChainedHIReq (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-Con in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-Sup in SCFBound (in case of ChainedRequestNotification)

B) Receiving Error

Precondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state WaitForBindResult (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-ChT in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChI in state Idle (in case of first ChainedHandlingInformationRequest)
or SCSM-ChT in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChT in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChI in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChT in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChI in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChT in SCFBound (in case of ChainedRequestNotification)

Error treatments in chaining Initiator SCF depends on SL. No further error treatment is performed in supporting SCF of chaining terminator SCF.

Procedures at responding entity (SCF)

A) Receiving operation

Precondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-Con in state SCFBound (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state SCFBound (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state BindPending (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

B) Returning Error

Precondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Postcondition: SCSM-ChI in state SCFBound (in case of ChainedEstablishChargingRecord)
or SCSM-ChT in state SCFBound (in case of first ChainedHandlingInformationRequest)
or SCSM-ChI in state Assisted mode (in case of ChainedHandlingInformationResult)
or SCSM-ChI in state PreparingAdditionalInformation (in case of ChainedNetworkCapability)
or SCSM-ChT in state SCFBound (in case of ChainedNotificationProvided or subsequent ChainedHandlingInformationRequest)
or SCSM-ChI in SCFBound (in case of ChainedProvideUserInformation)
or SCSM-ChT in SCFBound (in case of ChainedReportChargingInformation)
or SCSM-ChI in SCFBound (in case of ChainedRequestNotification)

Error treatments in the chaining Initiator SCF depends on SL. No further error treatment is performed in supporting SCF of chaining terminator SCF.

17.1.27 DirectoryBindError

17.1.27.1 General description

17.1.27.1.1 Error description

This error is sent by the SDF to the SCF/SDF to report a problem in establishing an authorized relationship. The conditions under which a directoryBindError is to be issued are defined in ITU-T Recommendation X.511 [39] (1997).

17.1.27.1.2 Argument description

The directoryBindError parameter and problem codes are specified in ITU-T Recommendation X.511 [39] (1996).

17.1.27.2 Operations SCF->SDF

directoryBind

Procedures at invoking entity (SCF)

A) Sending Operation

Precondition: SCSM state 1 Idle

Postcondition: SCSM state 3 Wait for Bind result

B) Receiving Error

Precondition: SCSM state 3 Wait for Bind result

Postcondition: SCSM state 1 Idle

Error Procedure is independent of the SL. The service processing should be terminated.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDF FSM state 1 Idle

Postcondition: SDF FSM state 2 Bind Pending

B) Returning Error

Precondition: SDF FSM state 2 Bind Pending (in case of Bind)

Postcondition: SDF FSM state 1 Idle (in case of Bind)

The SDF could not perform the operation and therefore sends a directoryBindError to the SCF. After returning the error, no further error treatment is performed.

17.1.27.3 Operations SDF->SDF

DSABind
DSAShadowBind

Procedures at invoking entity (SDF)

A) Sending Operation

Precondition: SDSM-ChI state 1 Idle (in case of DSABind)
or SDSM-ShM state 1 Idle (in case of supplier initiated DSAShadowBind)
or SDSM-ShC state 1 Idle (in case of consumer initiated DSAShadowBind)

Postcondition: SDSM-ChI state 2 Wait for subsequent requests (in case of DSABind)
or SDSM-ShM state 2 Wait for subsequent requests (in case of supplier initiated DSAShadowBind)
or SDSM-ShC state 2 Wait for subsequent requests (in case of consumer initiated DSAShadowBind)

B) Receiving Error

Precondition: SDSM-ChI state 3 Wait for Bind result (in case of DSABind)
or SDSM-ShM state 3 Wait for Bind result (in case of supplier initiated DSAShadowBind)
or SDSM-ShC state 3 Wait for Bind result (in case of consumer initiated DSAShadowBind)

Postcondition: SDSM-ChI state 1 Idle (in case of DSABind)
or SDSM-ShM state 1 Idle (in case of supplier initiated DSAShadowBind)
or SDSM-ShC state 1 Idle (in case of consumer initiated DSAShadowBind)

This error is reported to the SCF in case of DSABind operations.

Procedures at responding entity (SDF)

A) Receiving Operation

Precondition: SDSM-ChT state 1 Idle (in case of DSABind)
or SDSM-ShC state 1 Idle (in case of supplier initiated DSAShadowBind)
or SDSM-ShM state 1 Idle (in case of consumer initiated DSAShadowBind)

Postcondition: SDSM-ChT state 2 Bind pending (in case of DSABind)
or SDSM-ShC state 2 Wait for Bind result (in case of supplier initiated DSAShadowBind)
or SDSM-ShM state 2 Wait for Bind result (in case of consumer initiated DSAShadowBind)

B) Returning Error

Precondition: SDSM-ChT state 2 Bind pending (in case of DSABind)
or SDSM-ShC state 2 Wait for Bind result (in case of supplier initiated DSAShadowBind)
or SDSM-ShM state 2 Wait for Bind result (in case of consumer initiated DSAShadowBind)

Postcondition: SDSM-ChT state 1 Idle (in case of DSABind)
or SDSM-ShC state 1 Idle (in case of supplier initiated DSAShadowBind)
or SDSM-ShM state 1 Idle (in case of consumer initiated DSAShadowBind)

The SDF could not perform the operations for security reasons and therefore sends a directoryBindError to the other SDF. In the case of DSABind, the error is reported to the SCF.

17.1.28 ScfBindFailure

17.1.28.1 General description

17.1.28.1.1 Error description

This error is sent by the supporting SCF (or terminator supporting in the chaining case) to the controlling SCF (or initiator supporting SCF) to report a problem in establishing an authorized relationship.

17.1.28.1.2 Argument description

```
PARAMETER
    FailureReason
FailureReason:: = CHOICE {
    systemFailure   [0] UnavailableNetworkResource,
    scfTaskRefused  [1] ScfTaskRefusedParameter,
    securityError   [2] SET {
            problem     [0] SecurityProblem,
            spkmInfo    [1] SPKM-ERROR
            }
    }
```

17.1.28.2 Operations Controlling SCF-> Supporting SCF

scfBind

Procedure at Invoking Entity (Controlling SCF)

A) Sending Operation

Precondition: SCSM-Con state 1 Idle

Postcondition: SCSM-Con state 3 Wait for Bind result

B) Receiving Error

Precondition: SCSM-Con state 3 Wait for Bind result

Postcondition: SCSM-Con state 1 Idle

Error Procedure is independent of the SL. The service processing should be terminated.

Procedure at Responding Entity (Supporting SCF)

A) Receiving Operation

Precondition: SCSM-Sup state 1 Idle

Postcondition: SCSM-Sup state 2 Bind Pending

B) Returning Error

Precondition: SCF FSM state 2 Bind Pending (in case of Bind)

Postcondition: SCF FSM state 1 Idle (in case of Bind)

The SCF could not perform the operation and therefore sends a Security error to the SCF. After returning the error, no further error treatment is performed.

17.1.28.3 Operations Chaining Initiator Supporting SCF-> Terminator Supporting SCF

scfBind

Procedure at Invoking Entity (Initiator Supporting SCF)

A) Sending Operation

Precondition: SCSM-ChI state 1 Idle

Postcondition: SCSM-ChI state 3 Wait for Bind result

B) Receiving Error

Precondition: SCSM-ChI state 3 Wait for Bind result

Postcondition: SCSM-ChI state 1 Idle

Error Procedure is independent of the SL. The service processing should be terminated.

Procedure at Responding Entity (Terminator Supporting SCF)

A) Receiving Operation

Precondition: SCSM-ChT state 1 Idle

Postcondition: SCSM-ChT state 2 Bind Pending

B) Returning Error

Precondition: SCSM-ChT state 2 Bind Pending (in case of Bind)

Postcondition: SCSM-ChT state 1 Idle (in case of Bind)

17.1.29 ScfTaskRefused

17.1.29.1 General introduction

17.1.29.1.1 Error description

This Error is returned by a PE if it was not able to fulfill a specific task as requested by an operation. It is identical to taskRefused when security protection is not activated.

17.1.29.1.2 Argument description

```
PARAMETER
    ScfTaskRefusedParameter
ScfTaskRefusedParameter:: = OPTIONALLY-PROTECTED { SEQUENCE {
        reason  ENUMERATED {
            generic(0),
            unobtainable (1),
            congestion(2)
            },
        securityParameters [1] SecurityParameters OPTIONAL
    },
    SCFQOP.&scfErrorsQOP{@scfqop}
    }
```

17.1.29.2 Operations SCF->SCF

EstablishChargingRecord

HandlingInformationRequest

NetworkCapability

NotificationProvided

ProvideUserInformation

ReportChargingInformation

RequestNotification

ChainedEstablishChargingRecord

ChainedHandlingInformationRequest

ChainedNetworkCapability

ChainedNotificationProvided

ChainedProvideUserInformation

ChainedReportChargingInformation

ChainedRequestNotification

Procedures at invoking entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

Procedures at responding entity (SCF)

Refer to subclause 16.1.9 MissingParameter for the appropriate error procedures.

17.2 Entity related error procedures

The following subclauses define the error handling for the entity related errors. Since the error situations are not originated by the reception of an operation, the invoking entity is denoted here as the entity at which the error situation is detected. The responding entity is the entity which receives the error report.

The TCAP services used for reporting errors are described in clause 18.

17.2.1 Expiration of T_{SSF}

17.2.1.1 General description

17.2.1.1.1 Error description

A timeout occurred in the SSF on the response from the SCF.

17.2.1.2 Procedures SSF->SCF

Procedure at the invoking entity (SSF)

Timeout occurs in SSF on T_{SSF}

Precondition: SSF FSM state c Waiting for instructions.

or SSF FSM state d Waiting for end of User Interaction.

or SSF FSM state e Waiting for end of Temporary connection.

Postcondition: SSF FSM state a Idle.

The SSF FSM aborts the dialogue and moves to the Idle state, the CCF routes the call if necessary (e.g., defaultrouting to a terminating announcement). The abort is reported to the maintenance functions.

Procedure at the responding entity (SCF)

SCF receives a dialogue abort

Precondition: Any state.

Postcondition: SCSM state 1 Idle; if the abort is related to a SSF dialogue.

or SCSM state 2 Preparing SSF instructions; if the abort is related to an assisting SSF dialogue.

The SCF releases all allocated resources and reports the abort to the maintenance functions, if the abort is received on a SSF dialogue. The SCF releases all resources related to the dialogue, reports the abort to the maintenance functions and returns to state preparing SSF instructions, if the abort is received on an assisting SSF dialogue.

17.2.2 Expiration of T_{SRF}

17.2.2.1 General Description

17.2.2.1.1 Error description

A timeout occurred in the SRF on the response from the SCF. This procedure concerns only the direct SCF-SRF case.

17.2.2.2 Procedures SRF->SCF

Procedure at the invoking entity (SRF)

Timeout occurs in SRF on T_{SRF}

Precondition: SRSM state 2 Connected.

or SRSM state 3 User Interaction.

Postcondition: SRSM state 1 Idle.

The SRF aborts the dialogue and moves to the Idle state, all allocated resources are de-allocated. The abort is reported to the maintenance functions.

Procedure at the responding entity (SCF)

SCF receives a dialogue abort

Precondition: SCSM state 4 User Interaction.

Postcondition: SCSM state 2 Preparing SSF instructions.

The SCF releases all resources related to the dialogue, reports the abort to the maintenance functions and returns to state preparing SSF instructions.

17.2.3 Expiration of T_{cusf}

17.2.3.1 General description

17.2.3.1.1 Error description

A timeout occurred in the CUSF on the response from the SCF.

17.2.3.2 Procedures CUSF->SCF

Procedure at the invoking entity (CUSF)

Timeout occurs in CUSF for T_{cusf}

Precondition: CUSF FSM state b Waiting for instructions.

Postcondition: CUSF FSM state a Idle.

The CUSF FSM aborts the dialogue and moves to the Idle state, the CUSF terminates the association if necessary (e.g., default exception handling). The abort is reported to the maintenance functions.

Procedure at the responding entity (SCF)

SCF receives a dialogue abort

Precondition: Any state.

Postcondition: FSM for CUSF state N1 Idle.

The SCF releases all allocated resources and reports the abort to the maintenance functions.

18 Detailed operation procedures

NOTE: The detailed operation procedures in this subclause which cross reference the SCF FSMs for the pre- and post-conditions are for information only; refer to the note at the beginning of clause 12.

18.1 ActivateServiceFiltering procedure

18.1.1 General description

When receiving this operation, the SSF handles calls to destinations in a specified manner without request for instructions to the SCF. In the case of service filtering the SSF executes a specific service filtering algorithm. For the transfer of service filtering results refer to the operation "ServiceFilteringResponse".

18.1.1.1 Parameters

- filteredCallTreatment:
  This parameter specifies how filtered calls are treated. It includes information about the announcement to be played, the charging approach, the number of counters used and the release cause to be applied to filtered calls.
  - sFBillingChargingCharacteristics:
    This parameter determines the charging to be applied for service filtering. Its content is network specific.

    NOTE: Actual format and encoding is for further study.
  - informationToSend:
    This parameter indicates an announcement, a tone or display information to be sent to the calling party. At the end of information sending, the call shall be released.
    - inbandInfo:
      This parameter specifies the inband information to be sent.
      - messageID:
        This parameter indicates the message(s) to be sent, it can be one of the following:
        - elementaryMessageID:
          This parameter indicates a single announcement.
        - text:
          This parameter indicates a text to be sent. The text shall be transformed to inband information (speech). This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.
        - elementaryMessageIDs:
          This parameter specifies a sequence of announcements.
        - variableMessage:
          This parameter specifies an announcement with one or more variable parts.
      - numberOfRepetitions:
        This parameter indicates the maximum number of times the message shall be sent to the end-user.
      - duration:
        This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.
      - interval:
        This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end

of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

- tone:
  This parameter specifies a tone to be sent to the end-user.
  - toneID:
    This parameter indicates the tone to be sent.
  - duration:
    This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.
- displayInformation:
  This parameter indicates a text string to be sent to the end-user. This information can not be received by a Public Switched Telecommunication Network (PSTN) end-user.

- maximumNumberOfCounters:
  This parameter provides the number of counters to be allocated as well as the number of destinations included in the service filtering, i.e. "maximumNumberOfCounters" subsequent destination addresses beginning with the destination address provided in "filteringCriteria" are used for service filtering. One counter is assigned to each of these destination addresses.

  The number of counters may only be >1 if the "filteringCriteria" are of the type "addressAndService".
- releaseCause:
  This parameter provides the cause value used for call release after the "informationToSend" (for example announcement) has been sent to the calling party. If "releaseCause" is not present, the default value is the same as the ISUP value decimal 31 (normal unspecified).
- sFTariffMessage:
  This sub-parameter determines the tariff to be applied to the filtered call. It may include the ChargingTariffInformation sub-parameter.

- filteringCharacteristics:
  This parameter indicates the severity of the filtering and the point in time when the "ServiceFilteringResponse" shall be sent. It determines whether the "interval" or the "numberOfCalls" are used.
  - interval:
    After expiration of the interval timer the next call to arrive causes following actions:
    - sending of an "InitialDP" operation,
    - sending of an "ServiceFilteringResponse",
    - starting again the interval timer.

    When filtering is started the first interval is started.

    An interval of 0 indicates that all calls matching the filtering criteria will result in sending of an "InitialDP" operation and no filtering will be applied (i.e., no "ServiceFilteringResponse" will be sent).

    An interval of -1 indicates that none of the calls matching the filtering criteria will either result in sending of an "InitialDP" operation or a "ServiceFilteringResponse" operation.

    Other values indicate duration in seconds.
  - numberOfCalls:
    The n'th call causes an "InitialDP" operation and an "ServiceFilteringResponse" operation sent to the SCF. This threshold value is met if the sum of all counters assigned to one service filtering entity is equal to "numberOfCalls".

    A number of calls of 0 indicates that none of the calls matching the filtering criteria will result in sending of an "InitialDP" operation and a "ServiceFilteringResponse" operation.

- filteringTimeOut:
  This parameter indicates the duration of the filtering. When the time expires, a "ServiceFilteringResponse" is sent to the SCF and service filtering is stopped. Two approaches are supported (duration or stopTime):

  - duration:

    If the duration time expires, then service filtering is stopped and the final report is sent to the SCF.

    A duration of 0 indicates that service filtering is to be removed.

    A duration of -1 indicates an infinite duration.

    A duration of -2 indicates a network specific duration.

    Other values indicate duration in seconds.

  - stopTime:
    When the "stopTime" is met then service filtering is stopped and the final report is sent to the SCF. If "stopTime" was already met, i.e. the value of the stopTime is less than the value of the actual time but the difference does not exceed the value equivalent to 50 years, then service filtering is immediately stopped and the actual counter values are reported to the SCF. This occurs in cases where the SCF wishes to explicitly stop a running service filtering.

- filteringCriteria:

This parameter specifies which calls are filtered based on "serviceKey", "callingAddressValue", "calledAddressValue" or "locationNumber". It is a choice of "servicekey" or "addressAndService".

  - serviceKey:
    This parameter identifies unambiguously the requested IN service for which filtering should be applied.

  - addressAndService:
    This parameter identifies the IN service and dialled number for which filtering should be applied. The geographical area may also be identified ("callingAddressValue" and/or "locationNumber").

    - calledAddressValue:
      This parameter contains the dialled number towards which filtering shall be applied. The complete called party number shall be specified.

    - serviceKey:
      This parameter identifies unambiguously the requested IN service for which filtering should be applied.

    - callingAddressValue:
      This parameter contains the calling party number which identifies the calling party or geographical origin of the call for which filtering shall be applied.

    - locationNumber:
      This parameter identifies the geographical area from which the call to be filtered originates. It is used when "callingAddressValue" does not contain any information about the geographical location of the calling party.

- startTime:
  This parameter defines when filtering is started. If "startTime" is not provided or was already met, the SSF starts filtering immediately.

18.1.2 Invoking entity (SCF)

18.1.2.1 Normal procedure

SCF Precondition:

(1) SLPI detects that service filtering has to be initiated at the SSF.

SCF Postconditions:

(1) SLPI starts an application timer to monitor the expected end of service filtering.

(2) The SCME is in the state "Waiting For ServiceFilteringResponse".

Sending the "ActivateServiceFiltering" operation causes a transition of the SCME from the state "Service Filtering Idle" to the state "Waiting For SSF Service Filtering Response". The SCME remains in this state until the application timer in the SLPI expires. The SCME is informed by the SLPI about timer expiration. Then it moves to the state "Service Filtering Idle".

If no errors occurred after receiving an "ActivateServiceFiltering" at the SSF an empty Return Result is sent to the SCF. That causes no state transition in the SCME.

To change the parameters of an existing service filtering entity the SCF has to send an "ActivateServiceFiltering" operation with the same "filtering Criteria". The second parameter set replaces the first one.

18.1.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.1.3 Responding entity (SSF)

18.1.3.1 Normal procedure

SSF Precondition:

None.

SSF Postcondition:

(1) The SSME-FSM is in the state "Non Call Associated Treatment".

If there is no already existing SSME-FSM for the "filteringCriteria" provided then a new SSME-FSM is created. This SSME-FSM enters the state "Non-Call Associated Treatment" and initializes the service filtering for the specified IN calls. The parameters "filteredCallTreatment", "filteringCharacteristics", "filteringCriteria", "filteringTimeOut" and "startTime" are set as provided in the operation. A number of counters will be allocated and reset. In the case of the "startTime" that has not been met yet, the service filtering will be started at the specified point in time.

If the operation "ActivateServiceFiltering" addresses an already existing service filtering entity the parameters "filteredCallTreatment", "filteringCharacteristics" "filteringTimeOut" and "startTime" are modified as provided in the operation. In the case that the addressed service filtering entity is active the SSF reports the counter values to the SCF via the operation "ServiceFilteringResponse". The service filtering process is stopped if an already expired "stopTime" or "duration" equal to ZERO or a new not yet met "startTime" is provided. The SSF then proceeds as described for "ServiceFilteringResponse". In the case of the "startTime" that has not been met yet, the service filtering will be continued at the specified point in time.

If the service filtering proceeds then the SSME-FSM remains in the state "Non-Call Associated Treatment". Otherwise the SSME-FSM moves to state "Idle Management".

When a call matches several active "filteringCriteria" it should be subject to filtering on the most specific criteria, i.e. the criteria with the longest "callingAddressValue" or "locationNumber", or alternatively the criteria with the largest number of parameters specified.

When performing service filtering with the "filteringCriteria" - "addressAndService" the first parameters checked will always be the "serviceKey" and "calledAddressValue".

If an "ActivateServiceFiltering" operation is passed to the SSF with the "filteringCriteria" "addressAndService" with both callingAddressValue and "locationNumber" present, the following is applicable:

- When the SSF receives a call that matches "serviceKey" and "calledAddressValue" (in the active "filteringCriteria"), it investigates whether or not the "locationNumber" is present in the initial address message. If it is present and matches the active "filteringCriteria" the call is filtered. If the SSF finds that the "locationNumber" is absent, then it will check the "callingAddressValue" and perform filtering depending on that parameter.

If no errors occurred after receiving an "ActivateServiceFiltering" on the SSF an empty Return Result is sent to the SCF. That causes no state transition in the SSME-FSM.

Following application timers are used:

- detect moment to start service filtering (start time)
- duration time for service filtering
- interval time for service filtering (for timer controlled approach)

18.1.3.2 Error handling

If the SSF detects an error with any of the defined error values then this error is reported to the SCF.

The event is recorded in the SSF and an error condition indicated.

In case a new SSME FSM should be created, the relationship is ended and all concerned resources (e.g. counters) are released. The SSME FSM remains in the state "Idle Management".

In case there is already an existing SSME FSM, the service filtering data remains unchanged. The SSME FSM remains in the state "Non-Call Associated Treatment".

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.2 ActivityTest procedure

18.2.1 General description

This operation is used to check for the continued existence of a relationship between the SCF and SSF, between the SCF and the CUSF or between SCFs. If the relationship is still in existence, then the receiving entity will respond. If no reply is received within a given time period, then the SCF which sent this operation will assume that the receiving entity has failed in some way and will take the appropriate action.

18.2.1.1 Parameters

None.

18.2.2 Invoking entity (SCF)

18.2.2.1 Normal procedure

SCF Preconditions:

(1) A relationship exists between the SCF and the SSF, between the SCF and the CUSF or between SCFs.

(2) The activity test timer(Tati) expires, after which the "ActivityTest" operation is sent to the remote entity.

(3) The SCME is in state "Activity Test Idle".

SCF Postcondition:

(1) The SCME is in the state "Waiting for Activity Test Response". If a Return Result "ActivityTest" is received, the SCME resets the activity test timer, returns to state "Activity Test Idle", and takes no further action.

18.2.2.2 Error handling

If a time out on the "ActivityTest" operation or a P-Abort is received from TC, this is an indication that the relationship with the remote entity was somehow lost. If a time-out is received, SCF aborts the dialogue.

The SLPI that was the user of this dialogue will be informed, the corresponding SCSM-FSM will move to the state "idle".

18.2.3 Responding entity (SSF)

18.2.3.1 Normal procedure

SSF Precondition:

(1) A relationship exists between the SCF and the SSF .

(2) The SSME FSM is is the state "Idle Management".

SSF Postconditions:

(1) The SSME-FSM stays in the state " Idle Management".

(2) If the Dialogue ID is active and if there is a SSF-FSM using the dialogue, the SSME sends a Return Result "ActivityTest" to the SCF. The SSME-FSM returns to the state "Idle Management".

If the Dialogue ID is not active, the TC in the SSF will issue a P-Abort, the SSME will in that case never receive the "ActivityTest" req.ind and thus will not be able to reply.

18.2.3.2 Error handling

Operation related error handling is not applicable, due to class 3 operation.

18.2.4 Responding entity (CUSF)

18.2.4.1 Normal procedure

CUSF Precondition:

(1) A relationship exists between the SCF and the CUSF .

CUSF Postconditions:

(1) If the Dialogue ID is active and if there is a CUSF-FSM using the dialogue, the CUSME sends a Return Result "ActivityTest" to the SCF.

If the Dialogue ID is not active, the TC in the CUSF will issue a P-Abort, the CUSME will in that case never receive the "ActivityTest" req.ind and thus will not be able to reply.

18.2.4.2 Error handling

Operation related error handling is not applicable, due to class 3 operation.

18.2.5 Responding entity (controlling SCF or supporting SCF)

18.2.5.1 Normal procedure

SCF Precondition:

(1) A dialogue between the two SCFs has been established.

SCF Postconditions:

(1) The SCME-FSM stays in the same state .

(2) If the Dialogue ID is active and if there is a SCF-FSM using the dialogue, the SCME sends a Return Result "ActivityTests" to the other SCF.

If the Dialogue ID is not active, the TC in the other SCF will issue a P-Abort, the SCME will in that case never receive the "ActivityTest" req.ind and thus will not be able to reply.

18.2.5.2 Error handling

Operation related error handling is not applicable, due to class 3 operation.

18.3 AddEntry procedure

18.3.1 General description

The ITU-T Recommendation X.500 [36] "AddEntry" operation is used to request the SDF to add a leaf entry (either an object entry or an alias entry) in the DIT. For a full description of the AddEntry operation, see ITU-T Recommendation X.511 [39] subclause 11.1.

18.3.1.1 Parameters

See ITU-T Recommendation X.511 [39] subclauses 11.1.1 and 11.1.2.

18.3.2 Invoking entity (SCF)

18.3.2.1 Normal procedure

SCF Precondition:

(1) SCSM: "SDF Bound" or "Wait for Subsequent Requests".

SCF Postcondition:

(1) SCSM: "SDF Bound"

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the SL to add an entry in the SDF exists, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response ((E5) Response_from_SDF_with_Bind or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle".

When the SCSM has moved to state "Idle", the AddEntry operation was discarded. In the State "SDF Bound", the response of the AddEntry operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the AddEntry operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the SL to add an entry in the SDF exists an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response ((E7) Response_from_SDF) to the AddEntry operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the AddEntry operation or an error.

18.3.2.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 11.1.4 and 11.1.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.3.3 Responding entity (SDF)

18.3.3.1 Normal procedure

SDF Precondition:

(1) SDSM: "SCF Bound" or "Bind Pending"

SDF Postcondition:

(1) SDSM "SCF Bound"

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a "AddEntry" operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "AddEntry" operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before adding the new entry item, the SDF takes the following actions:

- verify that the superior object to which the entry should be added exists in the SDF;
- verify that the entry does not already exist in the SDF;
- verify that the access rights to add the entry and each of its components (attributes and values) are sufficient;
- verify that the entry conforms to the Directory schema;

After the specified actions indicated above are successfully executed, the entry is added into the SDF database. A null result is returned to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

18.3.3.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 11.1.4 and 11.1.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.4 ApplyCharging procedure

18.4.1 General description

This operation is used for interacting from the SCF with the SSF charging mechanisms. The "ApplyChargingReport" operation provides the feedback from the SSF to the SCF.

As several connection configurations may be established during a call, a possibility exists for the "ApplyCharging" to be invoked at the beginning of each connection configuration, for each party.

The charging scenarios supported by this operation are 4.1 and 4.2 (refer to ETS 300 374-1 [10], annex B "Charging scenarios").

18.4.1.1 Parameters

- aChBillingChargingCharacteristics:
  This parameter specifies the charging related information to be provided by the SSF and the conditions on which this information has to be reported back to the SCF via the "ApplyChargingReport" operation. Its contents is network operator specific.

- sendCalculationToSCPIndication:
  This parameter indicates that ApplyChargingReport operations (at least one at the end of the connection configuration charging process) are expected from the SSF. This parameter is always set to TRUE.

- partyToCharge:
  This parameter indicates the party in the call to which the "ApplyCharging" operation should be applied. If it is not present, then it is applied to the calling party (A-party).

18.4.2 Invoking entity (SCF)

18.4.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) The SLPI has determined that an "ApplyCharging" operation has to be sent.

(3) The FSM for CS is in any state except "CS Control Idle".

SCF Postconditions:

(1) No FSM state transition.

(2) The SLPI is expecting "ApplyChargingReport" operations from the SSF.

This operation is invoked by the SCF if a SLPI results in the request of interacting with the charging mechanisms within the SSF to get back information about the charging.

18.4.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services used for reporting operation errors are described in clause 18.

18.4.3 Responding entity (SSF)

18.4.3.1 Normal procedure

SSF Preconditions:

(1) The FSM for CS is in one of the following states:
"Waiting for Instructions"
"Waiting for End of User Interaction(WFI)" ,
"Waiting for End of User Interaction(MON)" ,
"Waiting for End of Temporary Connection(WFI)" ,
"Waiting for End of Temporary Connection(MON)" ,
"Monitoring" ,
or the assisting/hand-off SSF-FSM is in state:
"Waiting for Instructions"

SSF Postconditions:

(1) No FSM state transition

On receipt of this operation, the SSF sets the charging data using the information elements included in the operation and acts accordingly. In addition, the SSF will start the monitoring of the end of the connection configuration and other charging events, if requested.

18.4.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services used for reporting operation errors are described in clause 18.

18.5 ApplyChargingReport procedure

18.5.1 General description

This operation is used by the SSF to report charging related information to the SCF as requested by the SCF using the "ApplyCharging" operation.

During a connection configuration the "ApplyChargingReport" operation may be invoked on multiple occasions. For each call party and each connection configuration, the "ApplyChargingReport" operation may be used several times. Note that at least one "ApplyChargingReport" operation is to be sent at the end of the connection configuration charging process.

The charging scenarios supported by this operation are 4.1 and 4.2 (refer to ETS 300 374-1 [10], annex B "Charging scenarios").

18.5.1.1 Parameters

- callResult:
  This parameter provides the SCF with the charging related information previously requested using the "ApplyCharging" operation. The "CallResult" will include the "partyToCharge" parameter as received in the related "ApplyCharging" operation to correlate the result to the request. The remaining content of "CallResult" is network operator specific. Examples of these contents may be: bulk counter values, costs, tariff change and time of change, time stamps, durations, etc.

18.5.2 Invoking entity (SSF)

18.5.2.1 Normal procedure

SSF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) A charging event has been detected that was requested by the SCF via an "ApplyCharging" operation.

SSF Postconditions:

(1) If the connection configuration does not change then no FSM state transition shall occur.
If the connection configuration changes then the FSM shall move to:

- "Idle" state if there is no other EDP armed and no report requests are pending, or otherwise;
- shall remain in the same state.

This operation is invoked if a charging event has been detected that was requested by the SCF. The "ApplyChargingReport" operation only deals with charging events within the SSF itself. Examples of charging events may be: threshold value reached, timer expiration, tariff change, end of connection configuration initiated by a release, etc.

18.5.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services used for reporting operation errors are described in clause 18.

18.5.3 Responding entity (SCF)

18.5.3.1 Normal procedure

SCF preconditions:

(1) An "ApplyCharging" operation has been sent at the request of an SLPI and the SLPI is expecting an "ApplyChargingReport" from the SSF.

SCF postconditions:

(1) No FSM state transition if further reports, including "EventReportBCSM" and "CallInformationReport", are expected, or Transition to the state "Idle" if the report is the last one and no "EventReportBCSM" or "CallInformationReport" is expected.

On receipt of this operation the SLPI which is expecting this operation will continue.

18.5.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 the TC services used for reporting operation errors are described in clause 18.

18.6 AssistRequestInstructions procedure

18.6.1 General description

This operation is sent to the SCF by an SSF, which is acting as the assisting SSF in an assist or hand-off procedure, or by a SRF. The operation is sent when the assisting SSF or SRF receives an indication from an initiating SSF containing information indicating an assist or hand-off procedure.

18.6.1.1 Parameters

- correlationID:
  This parameter is used by the SCF to associate the "AssistRequestInstructions" from the assisting SSF or by a SRF with the request from the initiating SSF. The value of the "correlationID" may be extracted from the digits received from the initiating SSF or be all of the digits.

- iPAvailable:
  See ITU-T Recommendation Q.1290 [29]. This parameter is applicable to this operation only in the physical scenarios corresponding to assist with relay or hand-off. The use of this parameter is network operator dependent.

- iPSSPCapabilities:
  See ITU-T Recommendation Q.1290 [29]. This parameter is applicable to this operation only in the physical scenarios corresponding to assist with relay or hand-off. The use of this parameter is network operator dependent.

18.6.2 Invoking entity (SSF/SRF)

18.6.2.1 Normal procedure

SSF Precondition:

(1) An assist indication is detected by the assisting or Hand-off SSF.

SSF Postcondition:

(1) The assisting or Hand-off SSF waits for instructions.

On receipt of an assist indication from the initiating SSF, the SSF or SRF shall assure that the required resources are available to invoke an "AssistRequestInstructions" operation in the SSF/SRF and indicate to the initiating SSF that the call is accepted (refer to ITU-T Recommendation Q.71 [17]). The "AssistRequestInstructions" operation is invoked by the SSF or SRF after the call, which initiated the assist indication, is accepted. The assisting SSF FSM transitions to state "Waiting For Instructions".

18.6.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.6.3 Responding entity (SCF)

18.6.3.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the initiating SSF in case of assist procedure.

(2) The SCF waits for "AssistRequestInstructions".

SCF Postcondition:

(1) An SSF or SRF instruction is being prepared.

On receipt of this operation in the SCSM state "Waiting for Assist Request", the SCP has to perform the following actions:

If the "AssistRequestInstructions" operation was received from an assisting SSF, and the resource is available, the SCSM prepares the "ConnectToResource" and the user interaction operations like "PlayAnnouncement" or "PromptAndCollectUserInformation" to be sent to the assisting SSF.

The SCF determines SSF/SRF by means of "correlationID", "destinationNumber" or network knowledge.

If the "AssistRequestInstructions" operation was received from a SRF, and the resource is available, the SCSM prepares the user interaction operations e.g. "PlayAnnouncement" or "PromptAndCollectUserInformation" to be sent to the SRF.

On receipt of this operation from the Hand-off SSF, the SCSM associated with the Hand-off SSF transits from the "Idle" state to the "Preparing SSF Instructions" state.

18.6.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.7 CallGap procedure

18.7.1 General description

This operation is used to request the SSF to reduce the rate at which specific service requests are sent to the SCF.

18.7.1.1 Parameters

- gapCriteria:
  This parameter identifies the criteria for a call to be subject to call gapping.
  - calledAddressValue:
    This parameter indicates that call gapping will be applied when the leading digits of the dialled number of a call attempt match those specified in "gapCriteria".
  - gapOnService:
    This parameter indicates that call gapping will be applied when the "servicekey" of a call attempt match those specified in "gapCriteria".
  - calledAddressAndService:
    This parameter indicates that call gapping will be applied when the "serviceKey" and the leading digits of the dialled number of a call attempt match those specified in "gapCriteria".
  - callingAddressAndService:
    This parameter indicates that call gapping will be applied when the "serviceKey" and the leading digits of the calling party number or the location number of a call attempt match those specified in "gapCriteria".
  - gapAllINTraffic:
    This parameter indicates that call gapping will be applied for every IN call.
- gapIndicators:
  This parameter indicates the gapping characteristics.
  - duration:
    Duration specifies the total time interval during which call gapping for the specified gap criteria will be active.

    A duration of 0 indicates that gapping is to be removed.

    A duration of -1 indicates an infinite duration.

A duration of -2 indicates a network specific duration.

Other values indicate duration in seconds.

- gapInterval:

  This parameter specifies the minimum time between calls being allowed through.

  An interval of 0 indicates that calls meeting the gap criteria are not to be rejected.

  An interval of -1 indicates that all calls meeting the gap criteria are to be rejected.

  Other values indicate interval in milliseconds.

- controlType:
  This parameter indicates the reason for activating call gapping.

  The "controlType" value "sCPOverloaded" indicates that an automatic congestion detection and control mechanism in the SCP has detected a congestion situation.

  The "controlType" value "manuallyInitiated" indicates that the service and or network/service management centre has detected a congestion situation, or any other situation that requires manually initiated controls.[1]

- gapTreatment:
  This parameter indicates how calls that were stopped by the call gapping mechanism shall be treated.

  - informationToSend:
    This parameter indicates an announcement, a tone or display information to be sent to the calling party. At the end of information sending, the call shall be released.

    - inbandInfo:
      This parameter specifies the inband information to be sent.

      - messageID:
        This parameter indicates the message(s) to be sent, it can be one of the following:

        - elementaryMessageID:
          This parameter indicates a single announcement.

        - text:
          This parameter indicates a text to be sent. The text shall be transformed to inband information (speech). This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

        - elementaryMessageIDs:
          This parameter specifies a sequence of announcements.

        - variableMessage:
          This parameter specifies an announcement with one or more variable parts.

      - numberOfRepetitions:
        This parameter indicates the maximum number of times the message shall be sent to the end-user.

---

[1] The controlType 'manuallyInitiated' will have priority over 'sCPOverloaded' call gap.

It should be noted that also non-IN controlled traffic control mechanism can apply to an exchange with the SSF functionality. The non-IN controlled traffic control may also have some influence to the IN call. Therefore it is recommended to take measures to co-ordinate several traffic control mechanisms. The non-IN controlled traffic control and co-ordination of several traffic control mechanisms are out of the scope of INAP.

- duration:
  This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

- interval:
  This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

- tone:
  This parameter specifies a tone to be sent to the end-user.

  - toneID:
    This parameter indicates the tone to be sent.

  - duration:
    This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

- displayInformation:
  This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

- releaseCause:
  This parameter indicates that the call shall be released using the given release cause. See EN 300 356-1 [9].

- both:
  This parameter indicates inband info, a tone or display information to be sent to the calling party. At the end of information sending, the call shall be released, using the given release cause.

18.7.2 Invoking entity (SCF)

18.7.2.1 Normal procedure

SCF Preconditions:

(1) The SCF detects an overload condition persists and call gapping has to be initiated at the SSF, or
The SCF receives a manually initiated call gapping request.

SCF Postcondition:

(1) The SCME FSM remains in the same state upon issuing the "CallGap" operation.

A congestion detection and control algorithm monitors the load of SCP resources. After detection of a congestion situation the parameters for the "CallGap" operation are provided.

If the congestion level changes new "CallGap" operations may be sent for active gap criteria but with new gap interval. If no congestion is detected gapping may be removed.

A manual initiated call gap will take prevail over an automatic initiated call gap.

18.7.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.7.3 Responding entity (SSF)

18.7.3.1 Normal procedure

SSF Preconditions:

(1) Call gapping for gapCriteria is not active, or
Call gapping for gapCriteria is active.

SSF Postconditions:

(1) The SSME-FSM is in the state "Non call associated treatment".

(2) Call gapping for gapCriteria is activated, or
Call gapping for gapCriteria is renewed, or
Call gapping for gapCriteria is removed.

If there is no already existing SSME-FSM for the gap criteria provided, a new SSME-FSM is created. This SSME-FSM enters the state "Non call associated treatment" and initializes call gapping for the specified IN calls. The parameters "gapIndicators", "controlType" and "gapTreatment" for the indicated gap criteria will be set as provided by the "CallGap" operation.

In general, the manuallyInitiated call gapping will prevail over automatically initiated ("sCPOverloaded"). More specifically, the following rules will be applied in the SSF to manage the priority of different control Types associated with the same "gapCriteria":

- If an SSME-FSM already exists for the "gapCriteria" provided, then:

  1) if the (new) "controlType" equals an existing "controlType", then the new parameters (i.e., "gapIndicators" and "gapTreatment") will overwrite the existing parameter values.

  2) if the (new) "controlType" is different than the existing "controlType", then the new parameters (i.e., "controlType", "gapIndicators", and "gapTreatment") will be appended to the appropriate SSME-FSM (in addition to the existing parameters). The SSME-FSM remains in the state "Non-Call Associated Treatment".

If the SSF meets a TDP, it will check if call gapping was initiated either for the "serviceKey" or for the "calledAddressValue" assigned to this TDP. If not, an "InitialDP" operation can be sent. In case call gapping was initiated for "calledAddressAndService" or "callingAddressAndService" and the "serviceKey" matches, a check on the "calledAddressValue" and "callingAddressValue" - and optionally "locationNumber" - for active call gapping is performed. If not, an "InitialDP" operation can be sent.

In case of gapping on "callingAddressAndService" and the parameter "locationNumber" is present, gapping will be performed on "locationNumber" instead of "callingAddressValue".

If a call to a controlled number matches only one "gapCriteria", then the corresponding control is applied. If both "manuallyInitiated" and "sCPOverload" controls are active, then only the manually initiated control will be applied.

If a call to a controlled called number matches several active "gapCriteria", then only the "gapCriteria" associated with the longest called party number should be used, and the corresponding control should be applied. For example, the codes 1234 and 12345 are under control. Then the call with 123456 is subject to the control on 12345. Furthermore, if both "manuallyInitiated" and "sCPOverloaded" "controlTypes" are active for this "gapCriteria", then the "manuallyInitiated" control will be applied.

If "gapAllINTraffic" is active, then the checks for other criteria will be applied as described above. After these checks, control according to "gapAllINTraffic" will be applied for every IN call not blocked by other active criteria.

If call gapping shall be applied and there is no gap interval active, an "InitialDP" operation can be sent including the "cGEncountered" parameter according to the specified controlType. A new gap interval will be initiated as indicated by "gapInterval".

If a gap interval is active, no "InitialDP" operation is sent and the call is treated as indicated by "gapTreatment".

The call gap process is stopped if the indicated duration equals ZERO.

If call gapping proceeds then the SSME-FSM remains in the state "Non call associated treatment". Otherwise, the SSME-FSM moves to state "idle management".

18.7.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.8 CallInformationReport procedure

18.8.1 General description

This operation is used to send specific call information for a single call/call party to the SCF as requested by the SCF in previous "CallInformationRequest" operation. The report is sent at the end of a call/call party connection which is indicated by one of the events specified below.

18.8.1.1 Parameters

- requestedInformationList:
  According to the requested information the SSF sends the appropriate types and values to the SCF.

- legID:
  This parameter indicates the party in the call for which the information shall be collected and at the end of connection of which the report shall be sent. When absent, it shall apply to the first "outgoing" leg (i.e. the passive leg in an O-BCSM or the controlling leg in an T-BCSM) within the initial Call Segment, this can be a leg created by InitiateCallAttempt; or Connect/Continue/ContinueWithArgument.

18.8.2 Invoking entity (SSF)

18.8.2.1 Normal procedure

SSF Preconditions:

(1) The indicated or default party is released from the call or call setup towards the indicated or default party is not completed.

(2) Requested call information has been collected.

(3) "CallInformationReport" is pending due to a previously received "CallInformationRequest" operation.

(4) A control relationship exists between the SCF and the SSF.

SSF Postcondition:

(1) The FSM for CS in the SSF shall move to the "Idle" state in the case where no other report requests are pending and no EDPs are armed otherwise the SSF FSM shall remain in the same state.

(2) When the CallInformationReport is sent due to the receipt of the operation DisconnectLeg, which was followed by a state change to Waiting for Instructions, the SSF-FSM remains in the same state independent of other pending reports or armed EDPs.

If the SSF FSM executes a state transition caused by one of the following events:

- A party release for the indicated party or where A-party release causes a release of the default party.

- A party abandon for the indicated party or where A-party release causes a release of the default party.

- B party release for the indicated or default party.

- B party busy for the indicated or default party.

- SSF no answer timer expiration for the indicated or default party.

- route select failure indicated by the network for the indicated or default party.
- release of the indicated or default party by the SCF (DisconnectLeg),
- release call initiated by the SCF,

and "CallInformationRequest" is pending then one "CallInformationReport" operation is sent to the SCF containing all information requested.

If a "CallInformationReport" has been sent to the SCF then no "CallInformationReport" is pending, i.e. a further "CallInformationReport", for example in the case of follow-on, has to be explicitly requested by the SCF.

If an event causing the "CallInformationReport" is also detected by an armed EDP-R then immediately after "CallInformationReport" the corresponding "EventReportBCSM" has to be sent.

If an event causing the "CallInformationReport" is also detected by an armed EDP-N then immediately before "CallInformationReport" the corresponding "EventReportBCSM" has to be sent.

18.8.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.8.3 Responding entity (SCF)

18.8.3.1 Normal procedure

SCF Preconditions:

(1) An SLPI is expecting "CallInformationReport".

(2) A control relationship exists between the SCF and the SSF.

SCF Postcondition:

(1) The SLPI may be further executed.

In any state (except "Idle") the SCSM may receive "CallInformationReport" from the SSF, when the "CallInformationReport" is outstanding.

If "CallInformationReport" is outstanding and the SL program indicates that the processing has been completed, the SCSM remains in the same state until it receives the "CallInformationReport" operation.

When the SCF receives the "CallInformationReport" operation and the SL processing has been completed, then the FSM for CS in the SCSM moves to the "Idle" state.

When the SCF receives the "CallInformationReport" operation and the SL processing has not been completed yet, then the SCSM remains in the same state (EventReportBCSM and/or ApplyChargingReport and/or CallInformationReport pending).

18.8.4 Error handling

If requested information is not available, a "CallInformationReport" will be sent, indicating the requested information type, but with "RequestedInformationValue" filled in with an appropriate default value as specified by the network operator.

Operation related error handling is not applicable, due to class 4 operation.

18.9 CallInformationRequest procedure

18.9.1 General description

This operation is used to request the SSF to record specific information about a single call and report it to the SCF using the "CallInformationReport" operation.

18.9.1.1 Parameters

- requestedInformationTypeList:
  This parameter specifies a list of specific items of information which is requested.
  The list may contain:

  - callAttemptElapsedTime:
    This parameter indicates the duration between the end of INAP processing of operations initiating call setup ("Connect", "Continue") and the received answer indication from the indicated or default called party side. For a calling party leg this parameter has to be set to 0.

    In case of unsuccessful call setup the network event indicating the unsuccessful call setup stops the measurement of "callAttemptElapsedTime".

  - callStopTime:
    This parameter indicates the time stamp when the connection to the indicated or default party is released.

  - callConnectedElapsedTime:
    This parameter indicates the duration between the received answer indication from the indicated or defualt called party side and the release of that connection or party. For a calling party it indicates the duration between the sending of IDP and the release of that party.

  - calledAddress
    This parameter indicates the incoming called party address that was received by the SSF (i.e., before translation by the SCF) and is as available on the UNI or NNI and interpreted as per the numbering plan.

  - releaseCause:
    See EN 300 356-1 [9]. The release cause that applied to the indicated or default party.

- legID:
  This parameter indicates the party in the call for which the information has been collected. When absent, it indicates the first "outgoing" leg (i.e. the passive leg in an O-BCSM or the controlling leg in an T-BCSM) within the initial Call Segmen, this can be a leg created by InitiateCallAttempt; or Connect/Continue/ContinueWithArgument/AnalyseInformation/SelectRoute.

Any set of these values can be requested.

18.9.2 Invoking entity (SCF)

18.9.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) The SLPI has determined that a "CallInformationRequest" operation has to be sent by the SCF.

SCF Postcondition:

(1) The SLPI is expecting a "CallInformationReport" from SSF.

When the SLP requests call information, the SCF sends the "CallInformationRequest" operation to the SSF to request the SSF to provide call related information.

The "CallInformationRequest" operation specifies the information items to be provided by the SSF.

18.9.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.9.3 Responding entity (SSF)

18.9.3.1 Normal procedure

SSF Preconditions:

(1) Call origination attempt has been initiated.

(2) A control relationship exists between SSF and SCF.

SSF Postconditions:

(1) Requested call information is retained by the SSF.

(2) The SSF is waiting for further instructions.

The SSF may receive the "CallInformationRequest" operation within an existing call associated (CA) dialogue only.

The "CallInformationRequest" operation is accepted by the SSF Finite State Machine (SSF-FSM) only in the state "Waiting for Instructions". The operation does not lead to any transition to another state.

The SSF allocates a record for the indicated or default party and stores the requested information if already available and prepares the recording of information items, that will become available later like for example "callStopTimeValue".

Call informtion may be requested for any call party connection (identified by a legID). The indicated leg may be any controlling leg or passive leg.

18.9.3.2 Error handling

In any other than the "Waiting for Instruction" state the "CallInformationRequest" operation will be handled as an error with the error code "UnexpectedComponentSequence".

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.10 Cancel procedure

18.10.1 General description

The SCF uses this class 2 operation to request the SRF/SSF to cancel a correlated previous operation.

The SRF operation to be deleted can be either a "PlayAnnouncement" operation, a "PromptAndCollectUserInformation" operation or a PromptAndReceiveMessage operation.

The cancellation of an operation is indicated via a respective error indication, "Canceled", to the invoking entity of the cancelled "PlayAnnouncement" or "PromptAndCollectUserInformation" or "PromptAndReceiveMessage" operation. The "Cancel" operation can also be used to cancel all outstanding requests and enable the state machine (SSF) to go to idle. In this case the "Cancel operation does not specify any specific operation to be cancelled.

18.10.1.1 Parameters

- invokeID:
  This parameter specifies which operation invokation is to be cancelled, i.e. PromptAndCollectUserInformation, PromptAndReceiveMessage or PlayAnnouncement.

- callSegmentToCancel:
  This parameter specifies to which call segment the cancellation of a user interaction operation shall apply as well as the InvoleID to be cancelled.

- allRequests:
  This parameter indicates that all active requests for EDP reports, "ApplyChargingReport" and "CallInformationReport" should be cancelled.

NOTE: This cancellation is different from the invokeID based cancel mechanism described above.

18.10.2 Invoking entity (SCF)

18.10.2.1 Normal procedure

The SCF may either invoke this operation to the SSF or to the SRF, different conditions will previal in each case.

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF/SRF.

(2) An SLPI in the "Waiting for response from SRF" state has determined that a previously requested operation is to be cancelled; or
An SLPI has determined that it is no longer interested in any reports or notifications from the SSF and that the control relationship should be ended.

SCF Postcondition:

(1) The SLPI remains in the "Waiting for Response from SRF" state, or
In case all request are cancelled, the control relationship with the concerned FE (SSF) is ended and the SCSM FSM returns to "Idle" state.

18.10.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.10.3 Responding entity (SRF)

18.10.3.1 Normal procedure

SRF Precondition:

(1) A PlayAnnouncement or PromptAndCollectUserInformation operation has been received and the SRF is in the "User Interaction" state.

SRF Postcondition:

(1) The execution of the PlayAnnouncement or PromptAndCollectUserInformation operation has been aborted and the SRF remains in the "User Interaction" state.

18.10.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.10.4 Responding entity (SSF)

18.10.4.1 Normal procedure

SSF Precondition:

(1) The SSF-FSM is in the state "Waiting for Instructions" or "Monitoring".

SSF Postcondition:

(1) All active requests for reports and notifications have been cancelled.

(2) In case the SSF-FSM was in state "Monitoring" it shall return to idle, or
In case the SSF-FSM was in state "Waiting for Instructions" it will remain in that state.
A subsequent call-processing operation will move the SSF-FSM state to "Idle". The call, if in active state, is further treated by SSF autonomously as a normal (non-IN-) call.

18.10.4.2 Error handling

Sending of return error on cancel is not applicable in the cancel "allRequests" case. Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.11 ChainedAddEntry procedure

18.11.1 General description

The ITU-T Recommendation X.500 [36] "chainedAddEntry" operation is used to request remote processing of an AddEntry operation on behalf of an end user. For a full description of the chainedAddEntry operation, see ITU-T Recommendation X.511 [39] subclause 12.1.

18.11.1.1 Parameters

See ITU-T Recommendation X.518 [40]subclause 12.1.

18.11.2 Invoking entity (SDF)

18.11.2.1 Normal procedure

SDF Precondition:

1) The invoking SDF has received a request to perform an "AddEntry" operation for an end user which requires the operation to be chained to a responding SDF for processing.

2) SDSM-CHI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

1) SDSM-CHI: "SDF Bound"

2) A response to the request to perform an "AddEntry" operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to add an entry in the service data which requires processing in the responding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the "AddEntry" operation was discarded. In the State "SDF Bound", the response of the chainedAddEntry operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the chainedAddEntry operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to add an entry in the service data which requires processing in the resonding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedAddEntry operation previously issued to the responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedAddEntry operation or an error. This response will be returned to the originating end user.

18.11.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.11.3 Responding entity (SDF)

18.11.3.1 Normal procedure

SDF Precondition:

1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a "chainedAddEntry" operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the "dSABind" operation has been successfully executed. It remains in the same state. If the "dSABind" fails then the operation is discarded. If the "dSABind" operation succeeds then the chainedAddEntry operation is processed by the responding SDF.

When the responding SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "chainedAddEntry" operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the "chainedAddEntry" operation in one of two ways:

1) if the invoking SDF is located in another network then the responding SDF may "chain" the operation to another SDF within the same network as the responding SDF.

2) the operation is processed according to the actions described in the "AddEntry" procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

18.11.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and clause and the TC services that are used for reporting operating errors are described in clause 18.

18.12 ChainedConfirmedNotificationProvided procedure

18.12.1 General description

The operation reports to chaining terminator supporting SCF the outcome of the call in terms of charging information. . This is achieved by chaining the confirmedReportChargingInformation operation received from the controlling SCF to the chaining terminator supporting SCF'

18.12.1.1 Parameters

The parameters of this operation are the ConfirmedNotificationProvided parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2.

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.12.2 Invoking entity (chaining initiator supporting SCF)

18.12.2.1 Normal procedure

SCF Preconditions:

(1) the chaining initiator supporting SCF has received a "chainedEstablishChargingRecord" operation from the chaining terminator supporting SCF

(2) the chaining initiator supporting SCF has received a "confirmedReportChargingInformation" operation from the controlling SCF

(3) the SCSM-ChI his in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.12.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.12.3 Responding entity (chaining terminator supporting SCF)

18.12.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChT remains in the state SCF Bound

18.12.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.13 ChainedConfirmedReportChargingInformation procedure

18.13.1 General description

The operation is used by chaining initiator supporting SCF to report to the chaining terminator supporting SCF the outcome of the call in terms of charging information sent by the controlling SCF and at the same time requests for confirmation. This is achieved applying the SCF CHAINED syntax to ConfirmedReportChargingInformation operation.

It may be a response to a previously issued "ChainedEstablishChargingRecord" operation or the first operation relating to the charge to the supporting SCF. In the latter case, the call by call charging related information exchange by ChainedEstablishChargingRecord operation is not necessary, because the charging rate, etc. is pre-defined and is properly applied to the call in the controlling SCF.

18.13.1.1 Parameters

The parameters of this operation are the ConfirmedReportChargingInformation parameters and the SCF chaining parameters (see chainedConfirmedNotificationProvided parameters).

18.13.2 Invoking entity (chaining initiator supporting SCF)

18.13.2.1 Normal procedure

SCF Precondition:

(1) The SCSM-ChI is in state SCF bound

SCF Postcondition

(1) The SCSM-ChI remains in the state SCF bound

18.13.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TCAP services which are used for reporting operation errors are described in clause 18.

18.13.3 Responding entity (chaining terminator supporting SCF)

18.13.3.1 Normal procedure

SCF Precondition:

(1) The SCSM-ChT is in the state SCF bound

SCF Postcondition:

(1) The SCSM-ChT is in the state SCF bound

18.13.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16. and the TCAP services which are used for reporting operation errors are described in clause 18.

18.14 ChainedEstablishChargingRecord procedure

18.14.1 General Description

This operation is used by chaining terminator supporting SCF to give charging information to the chaining initiator supporting SCF. This operation is used by the chaining initiator supporting SCF to prepare and to send a establishChargingRecord operation to the controlling SCF

18.14.1.1 Parameters

The parameters of this operation are the EstablishChargingRecord parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.14.2 Invoking entity (chaining terminator supporting SCF)

18.14.2.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChT remains in the state SCF Bound

18.14.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.14.3 Responding entity (chaining initiator supporting SCF)

18.14.3.1 Normal procedure

SCF Preconditions:

The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.14.3.2 Error Handling

If chaining initiator supporting SCF receives "chainedEstablishedChargingInformation" operation withpout "charging paramiters and without "user credit" parameters it replies with "missing parameter" error

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.15 ChainedExecute procedure

18.15.1 General description

The "chainedExecute" operation is used to request remote processing of an Execute operation on behalf of an end user. For a full description of the operation chaining mechanism, see ITU-T Recommendation X.518 [40] clause 12.1.

18.15.1.1 Parameters

See ITU-T Recommendation X.518 [40] subclause 12.1 and Execute operation parameters

18.15.2 Invoking entity (SDF)

18.15.2.1 Normal procedure

SDF Precondition:

1) The invoking SDF has received a request to perform an "Execute" operation for an end user which requires the operation to be chained to a responding SDF for processing.

2) SDSM-CHI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

1) SDSM-CHI: "SDF Bound"

2) A response to the request to perform an "Execute" operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to add an entry in the service data which requires processing in the responding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the "Execute" operation was discarded. In the State "SDF Bound", the response of the chainedExecute operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the chainedExecute operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to add an entry in the service data which requires processing in the responding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedExecute operation previously issued to the responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedExecute operation or an error. This response will be returned to the originating end user.

18.15.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.15.3 Responding entity (SDF)

18.15.3.1 Normal procedure

SDF Precondition:

1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a "chainedExecute" operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the "dSABind" operation has been successfully executed. It remains in the same state. If the "dSABind" fails then the operation is discarded. If the "dSABind" operation succeeds then the chainedExecute operation is processed by the responding SDF.

When the responding SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "chainedExecute" operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the "chainedExecute" operation in one of two ways:

1) if the invoking SDF is located in another network then the responding SDF may "chain" the operation to another SDF within the same network as the responding SDF.

2) the operation is processed according to the actions described in the "Execute" procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

18.15.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.16 ChainedHandlingInformationRequest procedure

18.16.1 General description

This operation requests to chaining terminator supporting SCF assistance on how to proceed. This is achieved by chaining the HandlingInformationRequest Information operation received from the controlling SCF to the chaining terminator supporting SCF". The chainedHandlingInformationRequset operation can be sent to the chaining terminator SCF in the same message as the SCFBind operation.

18.16.1.1 Parameters

The parameters of this operation are the HandlingInformationRequest parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.16.2 Invoking entity (chaining initiator supporting SCF)

18.16.2.1 Normal procedure

SCF Preconditions:

(1) The chaining initiator supporting SCF has received a "HandlingInformationRequest "operation from the controlling SCF

(2) The SCSM-ChI is in the state "Prepare Chained HandlingInformationRequest"

SCF Post conditions:

(1) The SCSM-ChI moves to the state Wait for Bound Result

18.16.2.2 Error Handling

If the chaining terminator supporting SCF is not accessible, the supporting SCF FSM (SCSM-sup) is informed.

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.16.3 Responding entity (chaining terminator supporting SCF)

18.16.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state BindPending

SCF Post conditions:

(1) The SCSM-ChT remains in the state BindPending

18.16.3.2 Error Handling

If the chaining terminator supporting SCF receives a chainedHandlingInformationRequest operation while a ChainedHandlingInformationResult operation is still pending, "taskrefused error is returned to chaining initiator supporting SCF

If the chaining terminator supporting SCF receives a chainedHandlingInformationRequest operation that it is not able to handle, it can return a "referral" error to chaining initiator supporting SCF

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.17 ChainedHandlingInformationResult procedure

18.17.1 General description

This operation is used by chaining terminator supporting SCF to give the response to the chainedHandlingInformationRequest operation, received from the chaining initiator supporting SCF.

18.17.1.1 Parameters

The parameters of this operation are the HandlingInformationResult parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this

information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.17.2 Invoking entity (chaining terminator supporting SCF)

18.17.2.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.17.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.17.3 Responding entity (chaining initiator supporting SCF)

18.17.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.17.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.18 ChainedModifyEntry procedure

18.18.1 General description

The ITU-T Recommendation X.500 [36] "chainedModifyEntry" operation is used to request remote processing of an ModifyEntry operation on behalf of an end user. For a full description of the chainedModifyEntry operation, see ITU-T Recommendation X.518 [40] subclause 12.1.

18.18.1.1 Parameters

See ITU-T Recommendation X.518 [40] subclause 12.1.

18.18.2 Invoking entity (SDF)

18.18.2.1 Normal procedure

SDF Precondition:

1) The invoking SDF has received a request to perform an "ModifyEntry" operation for an end user which requires the operation to be chained to a responding SDF for processing.

2) SDSM-CHI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

1) SDSM-CHI: "SDF Bound"

2) A response to the request to perform an "ModifyEntry" operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to modify an entry in the service data which requires processing in the responding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the "ModifyEntry" operation was discarded. In the State "SDF Bound", the response of the chainedModifyEntry operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the chainedModifyEntry operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to modify an entry in the service data which requires processing in the responding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedModifyEntry operation previously issued to the responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedModifyEntry operation or an error. This response will be returned to the originating end user.

18.18.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.18.3 Responding entity (SDF)

18.18.3.1 Normal procedure

SDF Precondition:

1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a "chainedModifyEntry" operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the "dSABind" operation has been successfully executed. It remains in the same state.

If the "dSABind" fails then the operation is discarded. If the "dSABind" operation succeeds then the chainedModifyEntry operation is processed by the responding SDF.

When the responding SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "chainedModifyEntry" operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the "chainedModifyEntry" operation in one of two ways:

1) if the invoking SDF is located in another network then the responding SDF may "chain" the operation to another SDF within the same network as the responding SDF.

2) the operation is processed according to the actions described in the "ModifyEntry" procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

18.18.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.19 ChainedNetworkCapability procedure

18.19.1 General Description

This operation is used by chaining terminator supporting SCF to request the chaining initiator supporting SCF the type of services that are supported for the user and in the contextt of the call, if not already specified in the agreement and to agree with.

18.19.1.1 Parameters

The parameters of this operation are the NetworkCapability parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.19.2 Invoking entity (chaining terminator supporting SCF)

18.19.2.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChT remains in the state SCF Bound

18.19.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.19.3 Responding entity (chaining initiator supporting SCF)

18.19.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.19.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.20 ChainedNotificationProvided procedure

18.20.1 General description

This operation is used to report that a call condition previously specified by the" chaining terminator supporting SCF was met. This is achieved by chaining the notification Provided operation received from the controlling SCF to the chaining terminator supporting SCF"

18.20.1.1 Parameters

The parameters of this operation are the NotificationProvided parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.20.2 Invoking entity (chaining initiator supporting SCF)

18.20.2.1 Normal procedure

SCF Preconditions:

(1) The chaining initiator supporting SCF has received a "chainedRequestNotification" operation from the chaining terminator supporting SCF

(2) The chaining initiator supporting SCF has received a "notificationProvided" operation from the controlling SCF

(3) The SCSM-ChI is in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.20.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.20.3 Responding entity (chaining terminator supporting SCF)

18.20.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChT remains in the state SCF Bound

18.20.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.21 ChainedProvideUserInformation procedure

18.21.1 General description

This operation is used by chaining terminator supporting SCF to request the chaining initiator supporting SCF to send a provideUserInformation operation to the controlling SCF, in order to request the user information. The received user information will be used in the cahining terminator supporting SCF to determine the way the call should be treated.

18.21.1.1 Parameters

The parameters of this operation are the ProvideUserInformation parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.21.2 Invoking entity (chaining termintaor supporting SCF)

18.21.2.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChT remains in the state SCF Bound

18.21.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.21.3 Responding entity (chaining initiator supporting SCF)

18.21.3.1 Normal procedure

(1) The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.21.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.22 ChainedRemoveEntry procedure

18.22.1 General description

The ITU-T Recommendation X.500 [36] "chainedRemoveEntry" operation is used to request remote processing of an RemoveEntry operation on behalf of an end user. For a full description of the chainedRemoveEntry operation, see ITU-T Recommendation X.518 [40] subclause 12.1.

18.22.1.1 Parameters

See ITU-T Recommendation X.518 [40] subclause 12.1.

18.22.2 Invoking entity (SDF)

18.22.2.1 Normal procedure

SDF Precondition:

1) The invoking SDF has received a request to perform an "RemoveEntry" operation for an end user which requires the operation to be chained to a responding SDF for processing.

2) SDSM-CHI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

1) SDSM-CHI: "SDF Bound"

2) A response to the request to perform an "RemoveEntry" operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to remove an entry from the service data which reqires processing in the resonding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the "RemoveEntry" operation was discarded. In the State "SDF Bound", the response of the chainedRemoveEntry operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the chainedRemoveEntry operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to remove an entry from the service data which reqires processing in the resonding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from theresponding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedRemoveEntry operation previously issued to the

responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedRemoveEntry operation or an error. This response will be returned to the originating end user.

18.22.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.22.3 Responding entity (SDF)

18.22.3.1 Normal procedure

SDF Precondition:

1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a "chainedRemoveEntry" operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the "dSABind" operation has been successfully executed. It remains in the same state. If the "dSABind" fails then the operation is discarded. If the "dSABind" operation succeeds then the chainedRemoveEntry operation is processed by the responding SDF.

When the responding SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "chainedRemoveEntry" operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the "chainedRemoveEntry" operation in one of two ways:

1) if the invoking SDF is located in another network then the responding SDF may "chain" the operation to another SDF within the same network as the responding SDF.

2) the operation is processed according to the actions described in the "RemoveEntry" procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

18.22.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and clause and the TC services that are used for reporting operating errors are described in clause 18.

18.23 ChainedReportChargingInformation procedure

18.23.1 General description

The operation reports to chaining terminator supporting SCF the outcome of the call in terms of charging information. . This is achieved by chaining the reportChargingInformation operation received from the controlling SCF to the chaining terminator supporting SCF"

18.23.1.1 Parameters

The parameters of this operation are the ReportChargingInformation parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.23.2 Invoking entity (chaining initiator supporting SCF)

18.23.2.1 Normal procedure

SCF Preconditions:

(1) The chaining initiator supporting SCF has received a "chainedEstablishChargingRecord" operation from the chaining terminator supporting SCF

(2) The chaining initiator supporting SCF has received a "reportChargingInformation" operation from the controlling SCF

(3) The SCSM-ChI is in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.23.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16, and the TC services which are used for reporting operation errors are described in clause18.

18.23.3 Responding entity (chaining terminator supporting SCF)

18.23.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChT remains in the state SCF Bound

18.23.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause18.

18.24 ChainedRequestNotification procedure

18.24.1 General description

This operation is used by chaining terminator supporting SCF to request an event notification to the controlling SCF through the chaining initiator SCF; in fact it causes the chaining initiator supporting SCF to send a requestNotification operation to the controlling SCF.

18.24.1.1 Parameters

The parameters of this operation are the RequestNotification parameters and the following chaining parameters. It is to be noted that these parameters are common to all SCF chained operations except ultimateResponder present only in operations with result:

- originatingSCF:
  This parameter is used to convey the SCF identity (the encoding of this parameter required bilateral agreement between the involved operators) of the (ultimate) originator of the request when similar information is not already conveyed in the security parameter.

- target:
  This parameter indicates the identity of the subscriber involved in the operation.

- traceInformation:
  This parameter contains information to prevent looping among SCFs when chaining is in operation, this information is similar to ITU-T Recommendation X.518 [40] chaining mechanism (See ITU-T Recommendation X.518 [40] subclause 10.6). This parameter is present in the operation argument and in the result parameter (when existed).

- scfAuthenticationLevel:
  This parameter is optionally supplied when it is required to indicate the manner in which authentication has been carried out on during the bind operation. The type of the parameter is an **AuthenticationLevel** described in ITU-T Recommendation X.501 | ISO/IEC 9594-2 [37].

- timelimit:
  This parameter indicates the time by which the operation is to be completed

- ultimateResponder:
  This result parameter is used to convey back the ISDN address of the (ultimate) SCF responder.

- securityParameters:
  This is an optional parameter that conveys security related in the operation and result operation.

18.24.2 Invoking entity (chaining terminator supporting SCF)

18.24.2.1 Normal Procedure

SCF Preconditions:

(1) The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.24.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.24.3 Responding entity (chaining initiator supporting SCF)

18.24.3.1 Normal Procedure

SCF Preconditions:

(1) The SCSM-ChI is in the state SCF Bound

SCF Post conditions:

(1) The SCSM-ChI remains in the state SCF Bound

18.24.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.25 ChainedSearch procedure

18.25.1 General description

The ITU-T Recommendation X.500 [36] "chainedSearch" operation is used to request remote processing of an Search operation on behalf of an end user. For a full description of the chainedSearch operation, see ITU-T Recommendation X.518 [40] subclause 12.1.

18.25.1.1 Parameters

See ITU-T Recommendation X.518 [40] subclause 12.1.

18.25.2 Invoking entity (SDF)

18.25.2.1 Normal procedure

SDF Precondition:

1) The invoking SDF has received a request to perform an "Search" operation for an end user which requires the operation to be chained to a responding SDF for processing.

2) SDSM-CHI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

1) SDSM-CHI: "SDF Bound"

2) A response to the request to perform an "Search" operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to search for an entry in the service data which reqires processing in the resonding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the "Search" operation was discarded. In the State "SDF Bound", the response of the chainedSearch operation

((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the chainedSearch operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to search for an entry in the service data which reqires processing in the resonding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from theresponding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedSearch operation previously issued to the responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedSearch operation or an error. This response will be returned to the originating end user.

18.25.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.25.3 Responding entity (SDF)

18.25.3.1 Normal procedure

SDF Precondition:

1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a "chainedSearch" operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the "dSABind" operation has been successfully executed. It remains in the same state. If the "dSABind" fails then the operation is discarded. If the "dSABind" operation succeeds then the chainedSearch operation is processed by the responding SDF.

When the responding SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "chainedSearch" operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the "chainedSearch" operation in one of two ways:

1) if the invoking SDF is located in another network then the responding SDF may "chain" the operation to another SDF within the same network as the responding SDF.

2) the operation is processed according to the actions described in the "Search" procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

18.25.3.2 Error handling

Generic error handling for the operation related errors is described in in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.26 CollectInformation procedure

18.26.1 General description

The CollectInformation is a class 2 operation which is used by the SCF to request the call to return to the Collect_Information PIC, and then perform the basic originating call processing actions associated with this PIC (e.g., the checking of information in the CalledPartyNumber parameter with the supported dialing plan). This operation uses only the resources of the SSF/CCF to collect the information. The use of this operation is only appropriate for a call which had not yet left the setup phase.

When the user provides calledPartyNumber, Collect_Information PIC processing includes collecting of destination information from a calling party.

Refer to Subclause 11.5 General rules and procedure principles for inclusion of CPH capabilities.

18.26.1.1 Parameters

-

18.26.2 Invoking entity (SCF)

18.26.2.1 Normal procedure

SCF Precondition:

(1) An SLPI has determined that more information from the calling party is required to enable processing to proceed.

SCF Postcondition:

(1) SLPI execution is suspended pending receipt of dialled digits.

This operation is invoked in the SCSM FSM state "Preparing SSF Instructions" if the SLP requires additional information to progress the call. It causes a transition of the FSM to the state "Waiting for Notification or Report".

18.26.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.26.3 Responding entity (SSF)

18.26.3.1 Normal procedure

SSF Precondition:

(1) A TDP or EDP Request has been invoked.

(2) FSM for CS is in state "Waiting For Instructions"

SSF Postconditions:

(1) The SSF has executed a transition to the state "Monitoring".

(2) The SSF performs the call processing actions to collect destination information from the calling party. This may include prompting the party with in-band or out-band signals.

(3) Basic call processing is resumed at PIC Collect_Information.

The operation is only valid in the state "Waiting for Instruction" and after having received an operation "RequestReportBCSMEvent" for DPCollected_Information. The SSP has to perform the following actions:

- The SSF cancels T_{SSF}.
- When DPCollected_Information will be encountered, an "EventReportBCSM" operation will be invoked, and the SSF FSM will return to the state "Waiting for Instructions".

18.26.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18 .

18.27 ConfirmedNotificationProvided procedure

18.27.1 General description

This operation is used to report that a call condition previously specified by the supporting SCF or pre-arranged between two network operators was met and at the same time to request for confirmation. This is achieved applying the MAKE CONFIRM syntaxt to the Notification Provided operation.

18.27.1.1 Parameters

- notifications:
  This parameter contains an indication that a call condition previously expressed by the supporting SCF or pre-arranged between the two operators has been met. It links together call condition met and some information related to call conditions (if any).
- notificationInformation:
  This parameter contains any other kind of information that might be needed to be notified by a specific kind of SL. Information that can be conveyed has been agreed between network operators when defining the SL object.
- securityParameter:
  This is an optional parameter that conveys security related information.

18.27.2 Invoking entity (controlling SCF)

18.27.2.1 Normal procedure

SCF Precondition:

(1) The controlling SCF has received a "Request Notification" operation if call conditions at which this operation is sent has not been pre-arranged between two network operators.

(2) A call condition specified earlier by the supporting SCF or pre-arranged between two network operators has been met.

(3) The SCF FSM is in the state "Assisted Mode".

(4) he need to request for confirmation has been realized from the SL

SCF Postcondition:

(1) SCF FSM remains in Assisted mode state.

If any call resource has been engaged before the "confirmedNotificationProvided" operation is sent (e.g. as a result of the "requested Notification " operation), it remains as it is.

If several call conditions as specified by the supporting SCF or as pred-arranged between two network operators have been met, they are reported in sequence to the supporting SCF, which takes the appropriate actions.

18.27.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.27.3 Responding entity (supporting SCF)

18.27.3.1 Normal procedure

SCF Precondition:

(1) A dialogue between the two SCFs has been previously established

(2) The SCF FSM is in the state "Assisting Mode".

SCF Postcondition:

(1) SCF FSM remains in the state "Assisting Mode".

(2) A Return Result is sent for confirmation.

On receipt of the "ConfirmedNotificationProvided" operation the SLPI determines whether the call configuration should be modified as a consequence of the received notification information. If the call configuration in the invoking network needs to be modified, the SCF prepares instructions to assist the controlling SCF and sends them with a "handlingInformationResult" operation. Otherwise the SCF does not undertake any actions towards the controlling SCF, but the notification can be used in the SLPI (e.g. for charging purposes).

18.27.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.28 ConfirmedReportChargingInformation procedure

18.28.1 General description

The operation reports to an supporting SCF the outcome of the call in terms of charging information, the call being controlled by the controlling SCF and at the same tim requests for confirmation. This is achieved applying the MAKE CONFIRM syntaxt to ReportChargingInformation operation.

It may be a response to a previously issued "EstablishChargingRecord" operation or the first operation relating to the charge to the supporting SCF. In the latter case, the call by call charging related information exchange by EstablishChargingRecord operation is not necessary, because the charging rate, etc. is pre-defined and is properly applied to the call in the controlling SCF.

18.28.1.1 Parameters

- uniqueCallIdentity:
  This parameter is used to indicate the identity of the call that has motivated the use of the operation. There is a one-to-one relationship between this identity, the identity of the SLPI that treats the call and the identity of the assisting SLPI. This information can be further used to address a specific logic program instance.

- remainingUserCredit:
  This parameter contains the user's credit after the call. It is expressed in terms of telecommunication units.

- callRecord:
  This parameter contains the call record related to the call. This information consists (at least) of call duration, calling party and called party number. is network specific and defined by bilateral agreements between network operators.

- accountNumber:
  This parameter provides the unique identification of the account to which the cost of the call is registered

- securityParameter:
  This is an optional parameter that conveys security related information.

18.28.2 Invoking entity (controlling SCF)

18.28.2.1 Normal procedure

SCF Precondition:

(1) An "establishChargingRecord" operation has been received with the "expectedReport" parameter positionned to TRUE or it has been prearranged.

(2) A call has taken place.

(3) SCF FSM is in the "Assisted Mode" state

(4) SLP realizes the need to request for confirmation

SCF Postcondition:

(1) SCF FSM moves to the state "Waiting for Response from Supporting SCF"

After a call has taken place and either if a "EstablishChargingRecord" operation has been sent by the supporting SCF with the "reportExpected" parameter set to TRUE, requesting the charging information to be reported, or if it has been prearranged, the controlling SCF sends a "confirmedReportChargingInformation" operation in order to report the outcome of the charging procedure that took place for the call and in order to request for further confirmation from the supporting SCF. It contains the call identity to know the call it refers to and the user credit after the call.

18.28.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.28.3 Responding entity (supporting SCF)

18.28.3.1 Normal procedure

SCF Precondition:

(1) The SCF has sent an "establishChargingRecord" operation with the "reportExpected" parameter set to TRUE, or it has been prearranged.

(2) SCF FSM is in the "Assisting Mode" state

SCF Postcondition:

(1) SCF FSM moves to "Preparing a Response to Controlling SCF"

(2) A Return Result is sent for confirmation

On receipt of the "confirmedChargingInformation" operation, the SCF uses the information to update the user's data and it could also report the information to the management functions for security and billing purposes.

18.28.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16. and the TC services which are used for reporting operation errors are described in clause 18.

18.29 Connect procedure

18.29.1 General description

This operation is used to request the SSF to perform the call processing actions to route a call to a specific destination. To do so, the SSF may use destination information from the calling party (e.g. dialled digits) and existing call set-up information (e.g. route index to a list of trunk groups) depending on the information provided by the SCF.

In general all parameters which are provided in a Connect operation to the SSF shall replace the corresponding signalling parameter in the CCF, if the relevant parameter has already been received in the CCF, and shall be used for subsequent call processing. Parameters which are not provided by the Connect operation shall retain their value (if already assigned) in the CCF for subsequent call processing.

Refer to Subclause 11.5 General rules and procedure principles for inclusion of CPH capabilities.

18.29.1.1 Parameters

- destinationRoutingAddress:
  This parameter contains the called party numbers (see EN 300 356-1 [9]) towards which the call is to be routed. The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22]. The "destinationRoutingAddress" may include the "correlationID" and "scfID" if used in the context of a hand-off pocedure, but only if "correlationID" and "scfID" are not specified separately.

- alertingPattern:
  See ITU-T Recommendation Q.1290 [29] It only applies if the network signalling support this parameter or if SSF is the terminating local exchange for the subscriber.

- correlationID:
  This parameter is used by the SCF to associate the "AssistRequestInstructions" operation from the assisting SSF with the Request from the initiating SSF. The "correlationID" is used in the context of a hand-off procedure and only if the correlation id is not embedded in the "destinationRoutingAddress". The network operators has to decide about the actual mapping of this parameter on the used signalling system.

- cutAndPaste:
  This parameter is used by the SCF to instruct the SSF to delete (cut) a specified number of leading digits that it has received from the calling party and to paste the remaining dialled digits on to the end of the digits supplied by the SCF in the "destinationRoutingAddress".

- iSDNAccessRelatedInformation:
  Carries the same information as the protocol element ISUP Access Transport parameter in EN 300 356-1 [9]. See IN CS2 Signalling Interworking Requirements [].

- originalCalledPartyID:
  See EN 300 356-1 [9] Original Called Number signalling information. The use of this parameter in the context of the "Connect" operation is to be specified by the network operator. For CAMEL this parameter carries the dialled digits if the call is forwarded by the SCF.

- routeList:
  This parameter is used to select the outgoing trunk group used for routeing the call. A sequence of routes is provided to allow flexible routeing for applications such as VPN without increasing the number of queries required for such applications.

- scfID:
  See ITU-T Recommendation Q.1290 [29]. The scfID is used in the context of a hand-off procedure and only if the SCF is not embedded in the "destinationRoutingAddress". It may be used (e.g. for mobile applications) to perform a "hand-off" from an initiating SSF (e.g.. in a visiting mobile domain) to an requesting SSF (e.g. in home mobile domain) . The SCFId is used to provide the INAP address of the SCF to establish a connection between the requesting SSF and the specified SCF. The SCFId is to convey the necessary SCF address information (e.g. GT) in the network to the requesting SSF.
  The network operators has to decide about the actual mapping of this parameter on the used signalling system.

- carrier:
  This parameter indicates the transit network(s) requested for the call. The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22].

- serviceInteractionIndicators:
  This parameter contain indicators sent from the SCF to the SSF for control of the network based services at the originating exchange and the destination exchange.

- callingPartyNumber:
  This parameter, if present, is used to identify the calling party for the call (see EN 300 356-1 [9] Calling Party Number). It may be used for applications such as UPT, where only the SCF can verify the identity of the calling party.

- callingPartysCategory:
  See EN 300 356-1 [9] Calling Party Category signalling information.

- redirectingPartyID:
  This parameter, if present, indicates the last directory number the call was redirected from.

- redirectionInformation:
  See ITU-T Recommendation Q.763 [22] Redirection Information signalling information.

- displayInformation:
  This parameter indicates a text string to be sent to the end user. This information can not be received by a PSTN end-user.

- forwardCallIndicators:
  This parameter indicates if the call shall be treated as a national or international call. It also indicates the signalling capabilities of the network access, preceding network connection and the preferred signalling capabilities of the succeeding network connection. The network access capabilities does not indicate the terminal type. For example, an Integrated Services Private Branch eXchange (ISPBX) will have an ISDN type of access, but the end user terminal behind the ISPBX may be ISDN or non-ISDN

- genericNumbers:
  This parameter allows the SCF to modify the GenericNumber information received from the SSF, if any. Also, it allows the SCF to precise a Generic Number information to the SSF if the SSF has not preciously done so.

- serviceInteractionIndicatorsTwo
  Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services and network based services, respectively between different IN based services.

- iNServiceCompatibilityResponse:
  This parameter is used by the SSF to overwrite the INServiceCompatibilityIndication which has been derived during triggering of the given IN service. It is up to the Network Operator whether or not the overwrite is allowed.

- forwardGVNS:
  Identifies the originating service provider and provides information about the calling VPN user in terms of a customerID or a GVNS user group. The parameter will also carry routeing information for the terminating GVNS network.

- backwardGVNS:
  Information sent backward to the originating side about how the VPN call is terminated at the terminating side.

- callSegmentID
  This parameter indicates the CS to which the operation shall apply. When not provided, a default CSID of 1 is assumed.

- legToBeCreated
  This parameter indicates the LegID to be assigned to the newly created party. When not provided, a default LegID of 2 is assumed

- locationNumber:
  This parameter is used to convey the geographical area address for mobility services, see ITU-T Recommendation Q.762 [21].

- suppressionOfAnnouncement:
  This parameter indicates that announcements and tones which are played in the exchange at non-successful call set-up attempts shall be suppressed.

- oCSIApplicable:
  This parameter indicates to the SSP that the Originating CAMEL Subscription Information (OCSI), if present, shall be applied on the outgoing call leg created with the Connect operation.

- bearerCapability:
  This parameter indicates the type of the bearer capability connection or the transmission medium requirements to the user.See IN CS2 Signalling Interworking Requirements [48].
  It is a network option to select one of the two parameters to be used:

  - bearerCap:
    This parameter contains the value of the DSS1 Bearer Capability parameter (EN 300 403-1 [11]) in case the SSF is at local exchange level or the value of the ISUP User Service Information parameter (ITU-T Recommendation Q.763 [22]) in case the SSF is at transit exchange level.

  - tmr:
    The tmr is encoded as the Transmission Medium Requirement parameter of the ISUP according to ITU-T Recommendation Q.763 [22].

18.29.2 Invoking entity (SCF)

18.29.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An SLPI has determined that a "Connect" has to be sent by the SCF.

SCF Postcondition:

(1) SLPI execution may continue.

In the SCSM FSM state "Preparing SSF Instructions", this operation is invoked by the SCF if the SL results in the request to the SSF to route a call to a specific destination. If no event monitoring has been requested and no reports (CallInformationReport and ApplyChargingReport) have been requested in a previously sent operation, a SCSM FSM transition to state "Idle" occurs. Otherwise, if event monitoring has been requested or any report (CallInformationReport and ApplyChargingReport) has been requested, the SCSM FSM transitions to state "Waiting for Notification or Report". When the "Connect" operation is used in the context of a hand-off procedure, the SCSM FSM transitions to state "Idle". However, in this case, the SCF must maintain sufficient information in order to correlate the subsequent "AssistRequestInstructions" operation (from the assisting SSF or SRF) to the existing SLPI.

18.29.2.2 Error handling

If reject or error messages are received, then the SCSM informs the SLPI and remains in the state "Preparing SSF Instructions".

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.29.3 Responding entity (SSF)

18.29.3.1 Normal procedure

SSF Preconditions:

(1) Originating or terminating call attempt has been initiated.

(2) Basic call processing has been suspended at a DP.

(3) The CS waits for instructions.

(4) Only 1 BCSM instance may apply to the Call Segment (i.e. 1 or 2 party Call Segment).

(5) In case of an O_BCSM, it shall be suspended at any DP before the O_Active PIC or O_MidCall, in case no passive leg exists.

(6) In case of a T_BCSM, it shall be suspended at a DP before the T_Active PIC (call forwarding) or T_MidCall in case no controlling leg exists.

SSF Postconditions:

(1) The SSF performs the call processing actions to route the call to the specified destination.

(2) In the O-BCSM, when only address information is included in the Connect operation, call processing resumes at PIC Analyze_Information.

(3) In the O-BCSM, when address information and routeing information is included in the Connect operation, call processing resumes at PIC Select_Route.

(4) In the T-BCSM, when the Connect operation is received with a DestinationRoutingAddress, then a new O_BCSM shall be created and chained to the T_BCSM. The T_BCSM shall pass the information available (e.g. new number to which the call is to be routed) to the O_BCSM. The call processing shall resume from the Select_Facility PIC in th eT_BCSM and from the O_Null PIC in the O_BCSM.

On receipt of this operation in the SSF FSM state "Waiting for Instructions", the SSP performs the following actions:

- The SSF cancels T_{SSF}.
- If "cutAndPaste" is present, then the SSF deletes ("cut") from the dialled IN number the indicated number of digits and pastes the remaining dialled digits at the end of the "destinationRoutingAddress" parameter delivered by the SCF. The resulting directory number is used for routeing to complete the related call.
- If "cutAndPaste" is not present, then the "destinationRoutingAddress" parameter delivered by the SCF is used for routeing to complete the related call. Note that in the case of hand-off, this results in routeing to an assisting SSP or IP.
- If the "callingPartyNumber" is supplied, this value may be used for all subsequent SSF processing.
- If no EDPs have been armed and neither a CallInformationReport nor an ApplyChargingReport has been requested, the FSM goes to state "Idle" . Otherwise, the FSM goes to state "Monitoring".

No implicit activation or deactivation of DPs occurs.

Statistic counter(s) are not affected.

Connect completes when the INAP processing of the operation is complete and before the SSP starts the processing necessary to select a circuit.

Therefore in order to detect route select failure after a "Connect" it is necessary to explicitly arm the "Route Select Failure" EDP before sending the "Connect" (although they may be in the same message).

18.29.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.30 ConnectAssociation procedure

18.30.1 General description

This operation is used to request the CUSF to proceed with processing. Additional information which shall be used in further connection establishment is provided by the SCF. CUSF continues connection establishment (connection oriented bearer independant transport) to the specified destination using the address information received from the SCF. A two party association between the originating user/network application and the terminating user/network application is established.The communication path toward each of the user/network applications is identified by leg ID. A follow-on connection establishment after association release request event is received from terminating user/network application is allowed.

18.30.1.1 Parameters

- address
  This parameter contains the called party number to be used in the further connection establishment (connection oriented bearer independent transport). The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22].

18.30.2 Invoking Entity (SCF)

18.30.2.1 Normal Procedure

SCF Precondition

(1) FSM for CUSF within the SCF is in the state N2: Preparing CUSF Instructions.

(2) The SLPI has determined that a "ConnectAssociation" has to be sent by the SCF.

SCF Postcondition

(1) FSM for CUSF prepares to send the "ConnectAssociation".

(2) FSM for CUSF within the SCF moves to the state N1: Idle, if monitoring is not required, or moves to state N2.2: Waiting for Notification or Request, if monitoring is required.

NOTE: The information provided by the SCF depends on the service ASE located in the CUSF.

18.30.2.2 Error Handling

If the error will occur within the SCF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.30.3 Responding Entity (CUSF)

18.30.3.1 Normal Procedure

CUSF Precondition

(1) Bearer unrelated connection processing has been suspended at "Activation Received And Authorized DP" or at "Association Release Requested DP".

(2) CUSF-FSM is in the state b: Waiting For Instructions

CUSF Postcondition

(1) CUSF continues connection establishment (connection oriented bearer independent transport) to the specified destination using additional information from the SCF - if provided. A two -party connection is established.

(2) CUSF-FSM moves to the state a: Idle, if monitoring is not required, or
to the state c: Monitoring, if monitoring of BCUSM events was requested in a previous operation.

(3) BCUSM-FSM processing is resumed.

On receipt of this operation in the CUSF FSM state b: Waiting For Instructions, the CUSF performs following actions:

- The CUSF cancels T_{CUSF}.

18.30.3.2 Error Handling

If the error will occur within the CUSF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.31 ConnectToResource procedure

18.31.1 General description

This operation is used to connect a call from the SSF to a specialized resource. After successful connection to the SRF, the interaction with the caller can take place. The SSF relays all operations for the SRF and all responses from the SRF.

18.31.1.1 Parameters

- resourceAddress:
  This parameter identifies the physical location of the SRF.
  - iPRoutingAddress:
    This parameter indicates the routeing address to set up a connection towards the SRF.
    It is only valid when used in a single call segment CSA.
  - legID:
    This parameter indicates to which party in the call the subsequent interaction shall apply while maintaining the speech connection between that leg and any other legs connected to the same CS.
  - callSegmentID:
    This parameter indicates to which call segment the subsequent user interaction shall apply, i.e. to all parties connected to the call segment.
  - iPAddressAndLegID:
    This parameter indicates that both legID and iPRoutingAddress shall be used..
  - none:
    This parameter indicates that the call party is to be connected to a predefined SRF in the initial CS.
    It is only valid when used in a single call segment CSA.

  - iPAddressAndCallSegment:
    This parameter indicates that both Call Segment ID and iPRoutingAddress shall be used..

- serviceInteractionIndicators:
  This parameter contains indicators sent from the SCP to the SSP for control of the network based services at the originating exchange and the destination exchange.

- serviceInteractionIndicatorsTwo
  This parameter contains indicators which are exchanged between SSP and SCP to resolve interactions between IN based services and network based services, respectively between different IN based services.

18.31.2 Invoking entity (SCF)

18.31.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) The SLPI has determined that additional information from/to the call party/parties is needed.

(3) The FSM for CS is in the state "Routing to Resource", substate "Determine Mode".

(4) The SLPI has determined that the SRF can be accessed from the SSF.

SCF Postconditions:

(1) The SCSM-FSM moves to the state "User Interaction (substate monitoring or suspended)".

(2) The SCSM sends out a valid SCF - SRF operation, e.g. "PlayAnnouncement", "PromptAndCollectUserInformation" or PromptAndReceiveMessage operation accompanying the "ConnectToResource".

NOTE: In substate monitoring (call processing not suspended) only PlayAnnouncement is possible.

18.31.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.31.3 Responding entity (SSF)

18.31.3.1 Normal procedure

SSF Preconditions:

(1) A control relationship has been established.

(2) The FSM for CS is in the state "Waiting for Instructions" or in the state "Monitoring".

SSF Postconditions:

(1) The call is switched to the SRF.

(2) A control relationship to the SRF is established.

(3) If in state "Waiting for Instructions" the FSM for the CS moves to the state "Waiting for End of User Interaction (WFI)". If necessary, T_{SSF} is set.

(4) If in state "Monitoring" the FSM for the CS moves to the state "Waiting for End of User Interaction (MON)". If necessary, a guard timer T_{SSF} is set.

NOTE 1: Whether the T_{SSF} is used or not in this case is network operator dependent. But it must be synchronized with $T_{SCF-SSF}$ in the SCSM.

NOTE 2: The successful connection to the SRF causes a state transition in the SRF FSM from "Idle" to "Connected".

18.31.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.32 Continue procedure

18.32.1 General description

This operation is used to request the SSF to proceed with call processing at the DP at which it previously suspended call processing to await SCF instructions. The SSF continues call processing without substituting new data from the SCF.

This operation is only valid when used in a single call segment CSA and there are no more than 2 legs in the call segment.

18.32.1.1 Parameters

-none

18.32.2 Invoking entity (SCF)

18.32.2.1 Normal procedure

SCF Precondition:

(1) FSM for CS is in the state "Preparing CS instructions" or "Suspended and User Interaction"..

SCF Postcondition:

(1) FSM for CS is in the state "Waiting for Notification or Request", in case monitoring was required, or in the state "Idle", in case no monitoring was required or in the state "Not Suspended and User Interaction" in case user interaction.

The FSM for CS is in state "Preparing CS instructions". The "Continue" operation is invoked by a SLPI. This causes a FSM for CS transition to state "Idle" if no subsequent monitoring is required. However, if monitoring is required, like in the case of armed EDPs or outstanding report requests, the FSM for CS transitions to state "Waiting for Notification or Request" or state "Not Suspended and User interaction".

18.32.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.32.3 Responding entity (SSF)

18.32.3.1 Normal procedure

SSF Preconditions:

(1) BCSM: Basic call processing has been suspended at any DP.

(2) FSM for CS is in the state "Waiting for Instructions" or "Waiting for End of User Interaction (WFI)" or "Waiting for End of Temporary Connection (WFI)".

NOTE: The only applicable SCF-SRF user interaction operation is PlayAnnouncement.

SSF Postconditions:

(1) BCSM: Basic call processing continues, if all required resumptions within the involved CS has been received, otherwise the operation is discarded. (For details refer to Subclause 11.5 *General rules and procedure principles for inclusion of CPH capabilities.)*

(2) FSM for CS is in the state "Monitoring", because at least one EDP was armed, or a "CallInformationReport" or "ApplyChargingReport" was requested and no user interaction is ongoing, or

FSM for CS is in the state "Idle", because no EDPs were armed and neither the "CallInformationReport" nor the "ApplyChargingReport" was requested.

(3) If in state "Waiting for End of User Interaction (WFI)" the FSM for the CS moves to the state "Waiting for End of User Interaction (MON)". If necessary, a guard timer T_{SSF} is set.

(4) If in state "Waiting for End of Temporary Connection (WFI)", the FSM for the CS moves to the state "Waiting for End of Temporary Connection (MON)". If necessary, a guard timer T_{SSF} is set.

The SSF-FSM is in state "Waiting for instructions". The SSME-Control receives the "Continue" operation and relays it to the appropriate SSF-FSM. The SSF-FSM transitions to state "Idle" in case no EDPs are armed and no outstanding report requests are present. The SSF-FSM transits to state "Monitoring" if at least one EDP is armed, or if there is at least one outstanding report request. Basic call processing is resumed.

18.32.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.33 ContinueAssociation procedure

18.33.1 General description

This operation is used to request the CUSF to proceed with processing. Additional information which is not related to further connection establishment may be provided by the SCF.
CUSF continues connection establishment (connection oriented bearer independent transport) for the implied destination using any address information available in the BCUSM,. A two party or a one party association between an originating user/network application and a terminating user/network application is established depending on the location of the termination point for the concerned service application. For example a one-party association connection may be established between the SCF representing the terminating ASE and a user/application in the network representing the originating ASE. The communication path toward user/network application is identified by leg ID.

18.33.1.1 Parameters

18.33.2 Invoking Entity (SCF)

18.33.2.1 Normal Procedure

SCF Precondition

(1) FSM for CUSF within the SCF is in the state N2: Preparing CUSF Instructions.

(2) The SLPI has determined that a "ContinueAssociation" has to be sent by the SCF.

SCF Postcondition

(1) FSM for CUSF prepares to send the "ContinueAssociation".

(2) FSM for CUSF within the SCF moves to the state N1: Idle, if monitoring is not required, or moves to state N2.2: Waiting for Notification or Request, if monitoring is required.

NOTE: The information provided by the SCF depends on the service ASE located in the CUSF.

18.33.2.2 Error Handling

If the error will occur within the SCF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.33.3 Responding Entity (CUSF)

18.33.3.1 Normal Procedure

CUSF Precondition

(1) Bearer unrelated connection processing has been suspended at a DP in the BCUSM FSM.

(2) CUSF-FSM is in the state b: Waiting For Instructions

CUSF Postcondition

(1) CUSF continues connection establishment (connection oriented bearer independent transport).

(2) CUSF-FSM moves to the state a: Idle, if monitoring is not required, or
to the state c: Monitoring, if monitoring of BCUSM events was requested in a previous operation.

(3) BCUSM FSM resumes processing from the current DP where processing was suspended.

(4) If address information is available in the BCUSM to be used in further connection establishment a two party connection is set up, otherwise a one party connection set-up applies.

On receipt of this operation in the CUSF FSM state b: Waiting For Instructions, the CUSF performs following actions:

- The CUSF cancels TCUSF.

18.33.3.2 Error Handling

If the error will occur within the CUSF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.34 ContinueWithArgument procedure

18.34.1 General description

This operation is used to request the SSF to proceed with call processing at the DP at which it previously suspended call processing to await SCF instructions. It is also used to provide additional service related information to a User (Called Party or Calling Party) whilst the call processing proceeds.

This operation has to be sent for each BCSM instance in a CS for which call processing has been suspended (indicated by legID) in case the call suspension is due to a signalling event, i.e. once for each reported intercepted event to the SCF. If call processing suspension is caused by the SCF sending a CPH operation all BCSM instances in the involved CSs will have been suspended and this operation has to be sent once for each involved CS to resume call processing (indicated by CSID). For details refer to subclause 11.

18.34.1.1 Parameters

- legorCSID:
  - legID:
    This parameter indicates the party (Calling Party or Called Party) in the call who is interested in the additional service related information.
  - callSegmentID
    This parameter indicates the CS to which the operation shall apply to resume call processing when call suspension is caused by CPH operation..

- alertingPattern:

  This parameter indicates a specific pattern that is used to alert a subscriber (e.g. distinctive ringing, tones, etc.). It only applies if the network signalling support this parameter or if SSF is the terminating local exchange for the subscriber.

- genericName

  This parameter indicates the Call Party Name to be displayed to the end-user.

- iNServiceCompatibilityResponse:
  This parameter is used by the SSF to overwrite the INServiceCompatibilityIndication which has been derived during triggering of the given IN service. It is up to the Network Operator whether or not the overwrite is allowed.

- forwardGVNS:
  Identifies the originating service provider and provides information about the calling VPN user in terms of a customerID or a GVNS user group. The parameter will also carry routeing information for the terminating GVNS network.

- backwardGVNS:
  Information sent backward to the originating side about how the VPN call is terminated at the terminating side.

- serviceInteractionIndicatorsTwo
  Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services and network based services, respectively between different IN based services.

- locationNumber:
  This parameter is used to convey the geographical area address for mobility services, see ITU-T Recommendation Q.762 [21].

18.34.2 Invoking entity (SCF)

18.34.2.1 Normal procedure

SCF Precondition:

(1) FSM for CS is in the state "Preparing CS instructions" or in case of user interaction in the state "Suspended and User Interaction".

SCF Postcondition:

(1) FSM for CS is in the state "Waiting for Notification or Request", in case monitoring was required and no user interaction, or in the state "Idle", in case no monitoring was required. FSM for CS is in the state "Not Suspended an User Interaction" in case user interaction in monitoring state.

The FSM for CS is in state "Preparing CS instructions" or "Suspended and User Interaction": The "ContinueWithArgument" operation is invoked by a SLPI. This causes a FSM for CS transition to state "Idle" if no subsequent monitoring is required. However, if monitoring is required, like in the case of armed EDPs or outstanding report requests, the FSM for CS transitions to state "Waiting for Notification or Request" or "Not Suspended and User Interaction".

18.34.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.34.3 Responding entity (SSF)

18.34.3.1 Normal procedure

SSF Preconditions:

(1) BCSM: Basic call processing has been suspended at any DP.

(2) The FSM for CS is in the state "Waiting for Instructions" or "Waiting for End of User Interaction (WFI)" or "Waiting for End of Temporary Connection (WFI)".

NOTE: The only applicable SCF-SRF user interaction operation is PlayAnnouncement.

SSF Postconditions:

(1) BCSM: Basic call processing continues, if all required resumptions within the involved CS has been received, otherwise the operation is discarded. (For details refer to Subclause 11.5 *General rules and procedure principles for inclusion of CPH capabilities.)*

(2) The FSM for CS is in the state "Monitoring", because no user interaction is ongoing and at least one EDP was armed, or a "CallInformationReport" or "ApplyChargingReport" was requested, or

The FSM for CS is in the state "Idle", because no EDPs were armed and neither the "CallInformationReport" nor the "ApplyChargingReport" was requested

(3) If in state "Waiting for End of User Interaction (WFI)" the FSM for the CS moves to the state "Waiting for End of User Interaction (MON)". If necessary, a guard timer T_{SSF} is set.

(4) If in state "Waiting for End of Temporary Connection (WFI)" the FSM for the CS moves to the state "Waiting for End of Temporary Connection (MON)". If necessary, a guard timer T_{SSF} is set.

When the SSF-FSM is in state "Waiting for instructions" and the SSME-Control receives the "ContinueWithArgument , it relays it to the appropriate FSM for CS. The FSM for CS transitions to state "Idle" in case no EDPs are armed and no outstanding report requests are present The FSM for CS transits to state "Monitoring" if at least one EDP is armed, or if there is at least one outstanding report request. Basic call processing is resumed.

18.34.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.35 CoordinateShadowUpdate procedure

18.35.1 Geral Description

The ITU-T Recommendation X.500 [36] "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525 [42].

18.35.1.1 Parameters

For the coordinateShadowUpdate operation, see ITU-T Recommendation X.525 [42], subclause 11.1.

18.35.2 Supplier entity (SDF)

18.35.2.1 Normal Pocedure

18.35.2.1.1 CoordinateShadowUpdate sent by itself.

SDF Preconditions:

(1) SDSM-ShM: "SDF Bound"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update" in case of success

(2) SDSM-ShM: "SDF Bound" in case of failure

When the SDSM-ShM is in the state "SDF Bound" and a need of coordinating the shadow exists, an internal event occurs. This event, called (e165) Shadow_Coordinate_to_Consumer, causes a transition to the state "Wait for Coordination Result" and the operation is sent to the consumer SDF. The SDSM-ShM waits for the response from the consumer. The reception of the response ((E18) Shadow_Coordinate_Confirmed) to the "coordinateShadowUpdate" operation previously issued to the consumer SDF causes a transition to the state "Wait for Update" if the result of the "coordinateShadowUpdate" operation is positive. Otherwise the reception of an error ((E19) Coordinate_Failure) moves back the SDSM-ShM to the state "SDF Bound".

18.35.2.1.2 CordinateShadowUpdate sent with DSAShadowBind

SDF Preconditions:

(1) SDSM-ShM: "Wait for Subsequent Requests"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update" in case of success

(2) SDSM-ShM: "SDF Bound" in case of failure

When the SDSM-ShM is in the state " Wait for Subsequent Requests " and a need of coordinating the shadow exists, an internal event occurs. This event, called (e2) Shadow_Coordinate_to_Consumer, causes a transition to the state "Bind with Coordinate Shadow". The reception of the *delimiter* causes a transition to the state "Bind with CoordinateShadow Only" through the internal event (e5) Send_Bind_with_CoordShadow and the operations are sent to the consumer SDF. The SDSM-ShM then waits for the response from the consumer. The reception of the response (E7) SDF_Bind_Success to the previously issued ëDSAShadowBindí causes a transition to the state "Wait for Coordination Result" . The SDSM-ShM waits for the response from the consumer to the "coordinateShadowUpdate" operation previously issued to the consumer SDF. If the result of the "coordinateShadowUpdate" operation is positive, the event ((E18) Shadow_Coordinate_Confirmed) causes a transition to the state "Wait for Update". Otherwise the reception of an error ((E19) Coordinate_Failure) moves back the SDSM-ShM to the state "SDF Bound".

18.35.2.1.3 coordinateShadowUpdate sent with DSAShadowBind and UpdateShadow

SDF Preconditions:

(1) SDSM-ShM: " Wait for Subsequent Requests "

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update Confirmation" in case of success

(2) SDSM-ShM: "SDF Bound" in case of failure

When the SDSM-ShM is in the state " Wait for Subsequent Requests " and a need of coordinating the shadow exists, an internal event occurs. This event, called (e2) Shadow_Coordinate_to_Consumer, causes a transition to the state "Bind with Coordinate Shadow". The subsequent reception of an "UpdateShadow" operation through the internal event (e4) Update_to_Consumer causes a transition to the state "Bind with CoordinateShadow and Update" and the operations are sent to the consumer SDF. The SDSM-ShM then waits for the response from the consumer. The reception of the response (E9) SDF_Bind_Success to the previously issued ëDSAShadowBindí causes a transition to the state "Bound with Coordinate Shadow Sent" . The SDSM-ShM waits for the response from the consumer to the "coordinateShadowUpdate" operation previously issued to the consumer SDF. If the result of the "coordinateShadowUpdate" operation is positive the event ((E10) Shadow_Coordinate_Confirmed) causes a transition to the state "Wait for Update Confirmation". Otherwise the reception of an error ((E11) Coordinate_Failure) moves back the SDSM-ShM to the state "SDF Bound".

18.35.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause18.1.

18.35.3 Consumer entity (SDF)

18.35.3.1 Normal Procedure

20.35.3.1.1 CoordinateShadowUpdate received by itself

SDF Preconditions:

SDSM-ShC: "SDF Bound"

SDF Postconditions:

(1) SDSM-ShC: "Wait for Update" in case of success

(2) SDSM-ShC: "SDF Bound" in case of failure

The SDF is initially in the state "SDF Bound". After accepting the external event (E75) Shadow_Coordinate_from_Supplier caused by the reception of a "coordinateShadowUpdate" operation from the supplier SDF, a transition to the state "Wait for Coordination Result" occurs. The SDF performs the "coordinateShadowUpdate" operation according to the contents of the "coordinateShadowUpdate" argument. Once the SDF has completed the "coordinateShadowUpdate" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "SDF Bound" if the coordinateShadowUpdate fails or to the state "Wait for Update" if the coordinateShadowUpdate is successful.

20.35.3.1.2 CoordinateShadowUpdate received with DSAShadowBind or DSAShadowBind and UpdateShadow

SDF Preconditions:

1) SDSM-ShC: "Wait for Bind Result"

SDF Postconditions:

(1) SDSM-ShC: "Wait for Update" in case of success

(2) SDSM-ShC: "SDF Bound" in case of failure

The SDF is initially in the state ìWait for Bind Resultî waiting for other operations to be received than the "DSAShadowBind" operation. When receiving the "CoordinateShadowUpdate" operation, a transition to the same state occurs through the external event (E3) Request_from_Supplier. If the "UpdateShadow" operation is also received, the same transition occurs. The SDF performs the "DSAShadowBind" operation and a transition to the state "SDF Bound" occurs through the internal event (e5) SDF_Bind_Success. Since the "CoordinateShadowUpdate" operation has already been received, a transition to the state "Wait for Coordination Result" occurs through the external event (E7) Shadow_Coordinate_from_Supplier. Then, the SDF performs the "CoordinateShadowUpdate" operation according to the contents of the "CoordinateShadowUpdate" argument. Once the SDF has completed the "CoordinateShadowUpdate" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state ìSDF Boundî if the "CoordinateShadowUpdate" fails or to the state "Wait for Update" if the "CoordinateShadowUpdate" is successful.

18.35.3.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause18.1.

18.36 CreateCallSegmentAssociation procedure

18.36.1 General description

This operation creates a new CSA. The new CSA will not contain any Call Segments after creation. The SSF is responsible for specifying a new CSA ID for the created CSA which is unique within the SSF.

18.36.1.1 Parameters

Result Parameters:

- newCallSegmentAssociation
  This parameter specifies the new CSAID.

18.36.2 Invoking entity (SCF)

18.36.2.1 Normal procedure

SCF Preconditions:

(1) An SLPI has been invoked,

(2) An SLPI has determined that an "CreateCallSegmentAssociation" operation should be sent by the SCF.

(3) An instance of the FSM for CSA is created by SCME-Control .The FSM for CSA is in state "SSF Control Idle".

SCF Postconditions:

(1) A control relationship is established between the SCF and SSF.

(2) The FSM for CSA is in state "Preparing SSF Instructions".

(3) SLPI execution continues.

18.36.2.2 Error handling

Generic error handling for the operation related errors is described in clauses 16 and the TC services that are used for reporting operating errors are described in clause 18.

18.36.3 Responding entity (SSF)

18.36.3.1 Normal procedure

SSF Preconditions:

(1) An instance of the FSM for CSA is created by SSME-Control .The FSM for the CSA is in state "Idle".

SSF Postconditions:

(1) The SSF performs the appropriate actions.

(2) The FSM for CSA has moved from "Idle" state to state "Active".

(3) A Return Result is sent to report the new CSA ID to the SCF.

18.37 DisconnectForwardConnection procedure

18.37.1 General Description

This operation is used in the following two cases:

1) To clear a connection to a SRF

This operation is used to explicitly disconnect a connection to a resource (SRF) established previously with a "ConnectToResource" or an "EstablishTemporaryConnection" operation. It is used for a forward disconnection from the SSF. An alternative solution is the backward disconnect from the SRF, controlled by the "DisconnectFromIPForbidden" parameter in the "PlayAnnouncement" and "PromptAndCollectUserInformation" and "PromptAndReceiveMessage" operations.

2) To clear a connection to an assisting SSF

This operation is sent to the non-assisting SSF of a pair of SSFs involved in an assist procedure. It is used to disconnect the temporary connection between the initiating SSF and the assisting SSF, and the assisting SSF and its associated SRF.

This operation is only valid when used in a single call segment CSA (i.e. a CSA with only one CS).

18.37.1.1 Parameters

None.

18.37.2 Invoking entity (SCF)

18.37.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An assist- or a relay procedure is in progress.

(3) An SLPI has determined that a "DisconnectForwardConnection" operation has to be sent by the SCF.

SCF Postcondition:

(1) SLPI execution may continue.

The "DisconnectForwardConnection" operation is used to instruct the SSF to disconnect the concerned forward connection to the assisting SSF or the PE containing the SRF.

In the SCSM FSM state "User Interaction", substate "Waiting for Response from the SRF", this operation is invoked by the SCF when the SL determines that user interaction is finished and requests the SSF to disconnect the temporary connection to the assisting SSF or the SRF. The SCSM FSM then transitions to state "Preparing SSF Instructions".

18.37.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.37.3 Responding entity (SSF)

18.37.3.1 Normal procedure

SSF Preconditions:

(1) Call origination or termination attempt has been initiated.

(2) If basic call processing has been suspended at a DP, then the FSM for CS in the initiating SSF is in the state "Waiting for End of User Interaction (WFI)" or "Waiting for End of Temporary Connection (WFI)".

(3) If basic call processing has not been suspended at a DP, then the FSM for CS in the initiating SSF is in the state "Waiting for End of User Interaction (MON)" or in the state "Waiting for End of Temporary Connection (MON)".

SSF Postconditions:

(1) The connection to the SRF or assisting SSF is released.

(2) The FSM for CS is in state "Waiting for Instructions" if basic call processing has been suspended at a DP, otherwise in state "Monitoring".

The receipt of "DisconnectForwardConnection" results in disconnecting the assisting SSF or the PE containing the SRF from the concerned call. It does not release the connection from the SSF back to the end user.

This operation is accepted in the SSF FSM states "Waiting for End of Temporary Connection" or "Waiting for End of User Interaction". On receipt of this operation in these states , the SSF must perform the following actions:

- The initiating SSF releases the connection to the assisting SSF or the relay SRF.
- The SSF resets T_{SSF}.
- The SSF FSM goes to state "Waiting for Instructions".

NOTE: The successful disconnection to the SRF causes a state transition in the SRF FSM to "Idle". A current order (e.g. "PlayAnnouncement" or "PromptAndCollectUserInformation" or "PromptAndReceiveMessage") is cancelled and any queued order (e.g. "PlayAnnouncement" or "PromptAndCollectUserInformation") is discarded.

18.37.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.38 DisconnectForwardConnectionWithArgument procedure

18.38.1 General Description

This operation is used in the following two cases:

1) To clear a connection to a SRF

This operation is used to explicitly disconnect a connection to a resource (SRF) established previously with a "ConnectToResource" or an "EstablishTemporaryConnection" operation. It is used for a forward disconnection from the SSF. An alternative solution is the backward disconnect from the SRF, controlled by the "DisconnectFromIPForbidden" parameter in the "PlayAnnouncement" and "PromptAndCollectUserInformation" and "PromptAndReceiveMessage" operations.

2) To clear a connection to an assisting SSF

This operation is sent to the non-assisting SSF of a pair of SSFs involved in an assist procedure. It is used to disconnect the temporary connection between the initiating SSF and the assisting SSF, and the assisting SSF and its associated SRF.

18.38.1.1 Parameters

- partyToDisconnect:
  - legID:
    This parameter indicates to which party in the call the resource is currently connected
  - callSegmentID:
    This parameter indicates to which call segment the resource is currently connected.

18.38.2 Invoking entity (SCF)

18.38.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An assist- or a relay procedure is in progress.

(3) An SLPI has determined that a "DisconnectForwardConnection" operation has to be sent by the SCF.

SCF Postcondition:

(1) SLPI execution may continue.

The "DisconnectForwardConnection" operation is used to instruct the SSF to disconnect the concerned forward connection to the assisting SSF or the PE containing the SRF.

In the SCSM FSM state "User Interaction", substate "Waiting for Response from the SRF", this operation is invoked by the SCF when the SL determines that user interaction is finished and requests the SSF to disconnect the temporary connection to the assisting SSF or the SRF. The SCSM FSM then transitions to state "Preparing SSF Instructions".

18.38.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.38.3 Responding entity (SSF)

18.38.3.1 Normal procedure

SSF Preconditions:

(1) Call origination or termination attempt has been initiated.

(2) If basic call processing has been suspended at a DP, then the FSM for CS in the initiating SSF is in the state "Waiting for End of User Interaction (WFI)" or "Waiting for End of Temporary Connection (WFI)".

(3) If basic call processing has not been suspended at a DP, then the FSM for CS in the initiating SSF is in the state "Waiting for End of User Interaction (MON)" or "Waiting for End of Temporary Connection (MON)".

SSF Postconditions:

(1) The connection to the SRF or assisting SSF is released.

(2) The FSM for CS is in state "Waiting for Instructions" if basic call processing has been suspended at a DP, otherwise in state "Monitoring".

The receipt of "DisconnectForwardConnection" results in disconnecting the assisting SSF or the PE containing the SRF from the concerned call. It does not release the connection from the SSF back to the end user.

This operation is accepted in the SSF FSM states "Waiting for End of Temporary Connection" or "Waiting for End of User Interaction". On receipt of this operation in these states , the SSF must perform the following actions:

- The initiating SSF releases the connection to the assisting SSF or the relay SRF.
- The SSF resets T_{SSF}.
- The SSF FSM goes to state "Waiting for Instructions".

NOTE: The successful disconnection to the SRF causes a state transition in the SRF FSM to "Idle". A current order ("PlayAnnouncement" or "PromptAndCollectUserInformation") is cancelled and any queued order ("PlayAnnouncement" or "PromptAndCollectUserInformation") is discarded.

18.38.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.39 DisconnectLeg procedure

18.39.1 General description

This operation is used to request the SSF to release a specific leg associated with the call at any phase of the call and to retain any other leg not specified in the DisconnectLeg operation.

After the execution of the CPH operation the FSM model for the involved CS in the SSF shall transit to the WFI state while the remaining BCSM instances for the involved CS shall move from the PIC to the DP of the corresponding mid call DP in order to handle subsequent EDP arming and call processing operation.

18.39.1.1 Parameters

- legToBeReleased:
  This parameter indicates the party in the call to be released. See ITU-T Recommendation Q.1290 [29] "LegID".
- releaseCause:
  See ITU-T Recommendation Q.850 [24]. This parameter may be used by the SSF for generating specific tones to the party to be released or to fill in the "cause" parameter in the release message

18.39.2 Invoking entity (SCF)

18.39.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An SLPI has determined that a call party shall be released.

(3) The FSM for CS is in state C2 "Preparing CS Instructions".

SCF Postconditions:

(1) SLPI execution may continue. The FSM for CSA transits to "idle" on receiving the last pending report (if any), if the released leg was the last leg within the Call Segment Association.

(2) The FSM for CS remains in the state C2 "Preparing CS Instructions" if not the last leg in the CS is released, otherwise the FSM for CS moves to the state Idle.

18.39.2.2 Error handling

Generic error handling for the operation related errors are described in clause16 and the TC services which are used for reporting operation errors are described in clause18.

18.39.3 Responding entity (SSF)

18.39.3.1 Normal procedure

SSF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) When the involved leg is an "outgoing" leg (i.e. the passive leg in an O_BCSM or the controlling leg in a T_BCSM), the corresponding BCSM shall be at least at the Send_Call PIC in case of an O_BCSM or Present_Call in case of a T_BCSM.

SSF Postconditions:

(1) The SSF performs the call processing actions to release the indicated party.

(2) Any outstanding EDPs on that leg are disarmed, any pending reports will be sent. If the leg is a controlling leg, the leg status becomes "surrogate" in all the CS of the CSA.

(3) The FSM for CS remains in the same state, or if the released leg was the last leg within the Call Segment, the FSM for CS for that CS returns to the "idle" state.

(4) If the leg was the last leg within the CSA, the CSA-FSM returns to idle state.

(5) The FSM for CS for the involved Call Segment will move to the "Waiting for instructions" state. The remaining BCSM instances within the Call Segment will move from the PIC to the DP of the corresponding O_/T_MidCall DP, when not already suspended at a DP. Note that no MidCall EDP will be reported for this case.

(6) A Return Result is sent immediately after the successful change of the leg configuration is executed, this allows the SCF to be updated with the established connection view and to cater for possible interference problems with signalling events.

18.39.3.2 Error handling

Generic error handling for the operation related errors are described in clause16 and the TC services which are used for reporting operation errors are described in clause18.

18.40 DSABind procedure

18.40.1 General description

The ITU-T Recommendation X.500 [36] "dSABind" operation is used by the invoking SDF to create an authenticated association between an invoking SDF and an responding SDF to enable distributed processing of operations on behalf of the end user. It carries the authentication information of the end user if any. For a full description of the dSABind operation, see ITU-T Recommendation X.518 [40] subclause 11.1. A full description of distributed operation procedures can be found in ITU-T Recommendation X.518 [40].

18.40.1.1 Parameters

See ITU-T Recommendation X.518 [40] subclause 11.1.

18.40.2 Invoking Entity (SDF)

18.40.2.1 Normal procedure

SDF Preconditions:

1) A request for data access operation from an end user has arrived which needs to be "chained" to a remote SDF for processing and no chaining association exists between the SDFs for the originating end user.

2) SDSM-ChI: "Idle".

SDF Postconditions:

1) SDSM-ChI: "SDF Bound" in case of success.

2) SDSM-ChI: "Idle" in case of failure.

3) A chaining association exists between the invoking SDF and the responding SDF for the end user.

When the SCSM-ChI is in the state "Idle" and a need of the SL to interrogate an SDF exists, an internal event occurs. This event, called (e1) DSABind_to_SDF, causes a transition to the state "Wait for Subsequent Requests" and other operations are awaited. Until the application process has not indicated by a delimitor that the Bind should be sent, the SCSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimitor causes a transition to the state "Wait for Bind result" through the internal transition (e3) Send_Bind_with_Requests. The operation is sent to the responding SDF. The SCSM-SDF waits for the response from the responding SDF. The reception of the response [(E5) DSABind_Successful] to the Bind operation previously issued to the responding SDF causes a transition of the invoking SDF to the state "SDF Bound" if the result of the Bind operation is positive. Otherwise the reception of an error [(E4) DSABind_Error] moves back the SCSM-SDF to the state "Idle".

18.40.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.40.3 Responding Entity (SDF)

18.40.3.1 Normal procedure

SDF Preconditions:

- SDSM-ChT: "Idle".

SDF Postconditions:

1) SDSM-ChT: "SCF Bound" (success).

2) SDSM-ChT: "Idle" (failure).

The SDSM-ChT is initially in the state "Idle". After accepting the external event (E1) DSABind_from_SDF caused by the reception of a "dSABind" operation from the invoking SDF, a transition to state "Bind Pending" occurs. The responding SDF performs the Bind operation according to the contents of the dSABind argument. Once the responding SDF has completed the "dSABind" operation, the result or error indication is returned to the invoking SDF. The responding SDF returns to the state "Idle" if the Bind fails or to the state "SDF Bound" if the Bind is successful. Should the Bind request succeed, the result returned may consist of credentials of the dSABindResult. These credentials allow the user to establish the identity of the Directory. They allow information identifying the responding SDF (that is directly providing the Directory service) to be conveyed to the invoking SDF. The credentials are of the same form as those supplied by the user.

18.40.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.518 [40] clause 13 and the TC services that are used for reporting operating errors are described in clause 18.

18.41 DSAShadowBind procedure

18.41.1 General Description

The ITU-T Recommendation X.500 [36] "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525 [42].

18.41.1.1 Parameters

The parameters for the DSA ShadowBind operation are the same as those for the in-DirectoryBind operation specified in Core INAP.

18.41.2 Supplier entity (SDF)

18.41.2.1 Normal Pocedure

18.41.2.1.1 Supplier-initiated DSAShadowBind

18.41.2.1.1.1 DSAShadowBind sent by itself.

SDF Preconditions:

(1) SDSM-ShM: "Idle"

SDF Postconditions:

(1) SDSM-ShM: "SDF Bound " in case of success

(2) SDSM-ShM: "Idle" in case of failure

When the SDSM-ShM is in the state "Idle" and a need of providing copies exists, an internal event occurs. This event, called (e1) Bind_to_Consumer, causes a transition to the state " Wait for Subsequent Requests " and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow these subsequent operations to be sent at the same time as the DSAShadowBind. Until the application process has indicated by a delimiter that the DSAShadowBind should be sent, the SDSM-ShM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimiter causes a transition to the state "Wait for Bind Result" through the internal event (e3) Send_Bind. The operation is then sent to another SDF(consumer SDF). The SDSM-ShM waits for the response from the consumer SDF. The reception of the response ((E14)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the consumer SDF causes a transition to the state "SDF Bound" if the result of the " DSAShadowBind" operation is positive. Otherwise the reception of an error((E13)SDF_Bind_Error) moves back the SDSM-ShM to the state "Idle".

18.41.2.1.1.2 DSA ShadowBind sent with CoordinateShadowUpdate

SDF Preconditions:

(1) SDSM-ShM: "Idle"

SDF Postconditions:

(1) SDSM-ShM: "Wait For Coordination Result" in case of success

(2) SDSM-ShM: "Idle" in case of failure

When the SDSM-ShM is in the state "Idle" and a need of providing copies exists, an internal event occurs. This event, called (e1) Bind_to_Consumer, causes a transition to the state " Wait for Subsequent Requests" and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow subsequent operations to be sent at the same time as the DSAShadowBind. The reception of a CoordinateShadowUpdate operation in the " Wait for Subsequent Requests" state through the internal event (e2) Shadow_Coordinate_to Consumer, causes a transition to the state "Bind with Coordinate Shadow".

The reception of the delimiter in the "Bind with Coordinate Shadow" state causes a transition to the state "Bind with CoordinateShadow Only" through the internal event (e5) Send_Bind_with_CoordShadow. This causes the two operations to be sent simultaneously to another SDF(consumer SDF). The SDSM-ShM waits for the response from the consumer SDF. The reception of the response ((E7)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the consumer SDF causes a transition to the state "Wait for Coordination Result" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E6)SDF_Bind_Error) moves back the SDSM-ShM to the state "Idle".

18.41.2.1.1.3 DSAShadowBind sent with CoordinateShadowUpdate and UpdateShadow

SDF Preconditions:

(1) SDSM-ShM: "Idle"

SDF Postconditions:

(1) SDSM-ShM: "Bound with CoordinateShadow Sent" in case of success

(2) SDSM-ShM: "Idle" in case of failure

When the SDSM-ShM is in the state "Idle" and a need of providing copies exists, an internal event occurs. This event, called (e1) Bind_to_Consumer, causes a transition to the state " Wait for Subsequent Requests" and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow subsequent operations to be sent at the same time as the DSAShadowBind. The reception of a CoordinateShadowUpdate operation in the "Wait for Subsequent Requests" state through the internal event (e2) Shadow_Coordinate_to_Consumer, causes a transition to the state "Bind with Coordinate Shadow".

The subsequent reception of an "UpdateShadow" operation through the internal event (e4) Update to_Consumer causes a transition to the state "Bind with CoordinateShadow and Update". This causes the three operations to be sent simultaneously to another SDF(consumer SDF). The SDSM-ShM waits for the response from the consumer SDF. The reception of the response ((E9)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the consumer SDF causes a transition to the state "Bound with Coordinate Shadow Sent" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E8)SDF_Bind_Error) moves back the SDSM-ShM to the state "Idle".

18.41.2.1.2 Consumer-initiated DSAShadowBind

SDF Preconditions:

(1) SDSM-ShM: "Idle"

SDF Postconditions:

(1) SDSM-ShM: "SDF Bound " in case of success

(2) SDSM-ShM: "Idle" in case of failure

The SDF is initially in the state "Idle". After accepting the external event (E1) Bind_from_Consumer caused by the reception of a " DSAShadowBind " operation from the SDF(consumer SDF), a transition to the state "Wait for Bind Result" occurs. The DSAShadowBind operation may be received at the same time as the RequestShadowUpdate operation. In this case, a transition to the same state through the external event (E3) Request_from_Consumer occurs. The SDF performs the "DSAShadowBind" operation according to the contents of the "DSAShadowBind" argument. Once the SDF has completed the "DSAShadowBind" operation, the result or error indication is returned to the consumer SDF. The SDF returns to the state "Idle" if the "DSAShadowBind" fails or to the state "SDF Bound" if the "DSAShadowBind" is successful.

18.41.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause 18.1.

18.41.3 Consumer entity (SDF)

18.41.3.1 Normal Procedure

18.41.3.1.1 Supplier-initiated DSAShadowBind

SDF Preconditions:

(1) SDSM-ShC: "Idle"

SDF Postconditions:

(1) SDSM-ShC: "SDF Bound" in case of success

(2) SDSM-ShC: "Idleî in case of failure

The SDF is initially in the state "Idle". After accepting the external event (E1) Bind_from_Supplier caused by the reception of a " DSAShadowBind " operation from the SDF(supplier SDF), a transition to the state "Wait for Bind Result" occurs. The DSAShadowBind operation may be received at the same time as the CoordinateShadowUpdate operation or the CoordinateShadowUpdate and the UpdateShadow operations. In these cases, a transition to the same state through the external event (E3) Request_from_Supplier occurs once or twice. The SDF performs the " DSAShadowBind" operation according to the contents of the " DSAShadowBind " argument. Once the SDF has completed the " DSAShadowBind " operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "Idle" if the "DSAShadowBind" fails or to the state "SDF Bound" if the " DSAShadowBind " is successful.

20.41.3.1.2 Consumer-initiated DSAShadowBind

18.41.3.1.2.1 DSAShadowBind sent by itself

SDF Preconditions:

(1) SDSM-ShC: "Idle"

SDF Postconditions:

(1) SDSM-ShC: "SDF Bound" in case of success

(2) SDSM-ShC: "Idle" in case of failure

When the SDSM-ShC is in the state "Idle" and a need of requesting updates exists, an internal event occurs. This event, called (e1) Bind_to_Supplier, causes a transition to the state " Wait for Subsequent Requests " and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow these subsequent operations to be sent at the same time as the DSAShadowBind. Until the application process has indicated by a delimiter that the DSAShadowBind should be sent, the SDSM-ShC remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimiter causes a transition to the state "Wait for Bind Result" through the internal event (e3) Send_Bind. The operation is then sent to another SDF(supplier SDF). The SDSM-ShC waits for the response from the supplier SDF. The reception of the response ((E7)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the supplier SDF causes a transition to the state "SDF Bound" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E6)SDF_Bind_Error) moves back the SDSM-ShC to the state "Idle".

18.41.3.1.2.2 DSAShadowBind sent with RequestShadowUpdate

SDF Preconditions:

(1) SDSM-ShC: "Idle"

SDF Postconditions:

(1) SDSM-ShC: "Wait for RequestShadow Result" in case of success

(2) SDSM-ShC: "Idle" in case of failure

When the SDSM-ShC is in the state "Idle" and a need of requesting updates exists, an internal event occurs. This event, called (e1) Bind_to_Supplier, causes a transition to the state " Wait for Subsequent Requests" and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow these subsequent operations to be sent at the same time as the DSAShadowBind. The reception of the RequestShadowUpdate in the " Wait for Subsequent Requests" state through the internal event (e2) Request_to_Supplier, causes a transition to the state "Bind with RequestShadow". The two operations are then sent to another SDF(supplier SDF). The SDSM-ShC waits for the response from the supplier SDF. The reception of the response ((E5)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the supplier SDF causes a transition to the state "Wait for RequestShadow Result" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E4)SDF_Bind_Error) moves back the SDSM-ShC to the state "Idle".

18.41.3.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause 18.1.

18.42 EntityReleased procedure

18.42.1 General description

This operation is used to inform the SCP about the release of an entity (CS, BCSM) caused by exception or errors. It is sent by the CSA FSM if this information cannot be conveyed within an TC_ABORT or TC_END if the TC dialogue has to be kept because of other existing entities(CS, BCSM) in this CSA which are not affected by this error/exception. This operation is not sent if the last CS was released.

The operation EntityReleased is not used if the release of the entity can be reported through other operations, e.g. EventReportBCSM, CallInformationRequest.

18.42.1.1 Parameters

- CSFailure
  Indicates that an CS was released
  - callSegmentID
    identifies the released CS
  - reason
    gives network specific information about the kind of error/exception (e.g. external or internal error or exception)
  - cause
    indicates the cause of releasing this specific entity. The cause may be used by the SCF to decide about the further handling of the call.
- BCSMFailure
  - legID
    identifies the released leg

- reason
  gives network specific information about the kind of error/exception (e.g. external or internal error or exception)

- cause
  indicates the cause of releasing this specific entity. The cause may be used by the SCF to decide about the further handling of the call.

18.42.2 Invoking entity (SSF)

18.42.2.1 Normal procedure

SSF Preconditions:

(1) Any state except idle

SSF Postconditions:

(1) If the released entity was a BCSM (leg) than only the appropriate resources are released.
If the released entity was a CS the related FSM goes to idle

18.42.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.42.3 Responding entity (SCF)

18.42.3.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

SCF Postconditions:

(1) The SCF-resources related to the released entity are released.

(2) The SLPI is further executed.

18.42.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.43 EstablishChargingRecord procedure

18.43.1 General Description

This operation is used by the supporting SCF to give charging information to another SCF controlling a call, so that it can charge the user (on-line charging included). The sent information is used to parameterize a charging function. This charging function is generic and can be used to charge any kind of calls. (This option might be used not used if the needed charging information has been prearranged).

18.43.1.1 Parameters

- userCredit:
  This parameter contains the user's credit. It is expressed in terms of telecommunication units.
- chargingParameters
  This parameter contains the different parameters that are used to parameterize the generic charging function.
- reportExpected:
  This parameter contains a boolean indicating if the charging information should be passed back to the supporting SCF at the end of the call.
- securityParameter:
  This is an optional parameter that conveys security related

18.43.2 Invoking entity (supporting SCF)

18.43.2.1 Normal procedure

SCF Precondition:

(1) "handlingInformationRequest" operation has been received.

(2) The need for providing the controlling SCF with charging information has been identified by the SLPI.

(3) The SCF FSM is in the "Assisting mode" state

SCF Postcondition:

(1) The SCF waits for the result of the "EstablishChargingRecordI" operation or none.

Before sending the "establishChargingRecord" operation, the SCF has received a "handlingInformationRequest" operation containing information about the call, the controlling network and the user. According to the agreementID contained in the received SCFBind operation, the supporting SCF knows if it (or the controlling SCF) can perform the charging of the call. When charging is delegated to the controlling SCF, the supporting SCF has to send the parameters of the charging function, unless all the needed charging information has been prearranged. The supporting SCF then sends the "establishChargingInformation" operation containing either the charging parameters or the remaining user credit.

Once the operation has been sent, depending on the contents of the "expectedReport" parameter, the SCF waits for a reply (either a "confirmedReportChargingInformation" or "reportChargingInformation" operation.

18.43.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.43.3 Responding entity (controlling SCF)

18.43.3.1 Normal procedure

SCF Precondition:

(1) A relationship between two SCFs has been established. A "handlingInformationRequest" operation has been sent.

(2) The SCF FSM is in the "Assisted Mode" state

SCF Postcondition:

(1) The SCSM is able to charge the call. The charging function is parameterized and ready to be used.

(2) The SCF FSM remains in the same state

On receipt of the "establishChargingInformation" operation, the charging function is parameterized. Depending on the type of call and the invoked services, the SCF decides where charging can take place (i.e. in which FE), but charging will be done according to the charging function.

Depending of the value of the "expectedReport" parameter, the SCF is ready to send the reply or not (the type of reply is determined autonomously by the SLPI in the controlling SCF, i.e. a "reportChargingInformation operation or a "confirmedChargingInformation" operation if further confirmation from the supporting SCF is expected) to the supporting SCF.

18.43.3.2 Error Handling

If the controlling SCF receives an "establishChargingInformation. operation without the "chargingParameter" and without the "userCredit" parameter, it replies with a "missingParameter". Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.44 EstablishTemporaryConnection procedure

18.44.1 General Description

This operation is used to create a connection between an initiating SSF and an assisting SSF as part of a service assist procedure. It can also be used to create a connection between a SSF and a SRF, for the case where the SRF exists in a separately addressable PE.

18.44.1.1 Parameters

- assistingSSPIPRoutingAddress:
  This parameter indicates the destination address of the SRF for assist procedure. The "assistingSSPIPRoutingAddress" may contain embedded within it, a "correlationID" and "scfID", but only if "correlationID" and "scfID" are not specified separately.

- correlationID:
  This parameter is used by the SCF to associate the "AssistRequestInstructions" from the assisting SSF (or the SRF) with the Request from the initiating SSF. The "correlationID" is used only if the correlation id is not embedded in the "assistingSSPIPRoutingAddress". The network operators has to decide about the actual mapping of this parameter on the used signalling system.

- scfID:
  See ITU-T Recommendation Q.1290 [29]. The "scfID" is used only if the SCF id is not embedded in the "assistingSSPIPRoutingAddress". The network operators has to decide about the actual mapping of this parameter on the used signalling system.

- carrier:
  This parameter indicates the transit network(s) requested for the call. The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22].

- serviceInteractionIndicators:
  This parameter contain indicators sent from the SCP to the SSP for control of the network based services at the originating exchange and the destination exchange.

- partyToConnect:
  This parameter shall be present when applied in a multi call segment CSA.
  When not present in a single call segment CSA it implies that user interaction shall apply to the call segment, i.e. to all parties connected to the call segment.

  - legID:
    This parameter indicates to which party in the call the subsequent interaction shall apply while maintaining the speech connection between that leg and any other legs connected to the same CS.

  - callSegmentID:
    This parameter indicates to which call segment the subsequent user interaction shall apply, i.e. to all parties connected to the call segment.

- serviceInteractionIndicatorsTwo
  Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services and network based services, respectively between different IN based services.

18.44.2 Invoking entity (SCF)

18.44.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) The SL has determined that a connection is needed between the SSF and SRF or between the SSF and an assisting SSF.

SCF Postcondition:

(1) The FSM for Assisting SSF is "Waiting for Assist Request Instructions".

In the SCSM FSM state "Routing to Resource", this operation is invoked by the SCF when the SL determines that an assisting SSF or a Direct SCF-SRF relation is needed. The SCSM FSM then transitions to state "Waiting for Assist Request Instructions".

18.44.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.44.3 Responding entity (SSF)

18.44.3.1 Normal procedure

SSF Preconditions:

(1) Call origination attempt has been initiated.

(2) The FSM for CS is in state "Waiting for Instructions" or in state "Monitoring".

(3) The SSF is not an assisting SSF.

SSF Postconditions:

(1) The SSF performs the call processing actions to route the call to the assisting SSF or SRF according to the "assistingSSPIPRoutingAddress" requested by the SCF.

(2) The CS waits for end of temporary connection.

(3) If in state "Waiting for Instructions" the FSM for the CS moves to the state "Waiting for End of Temporary Connection (WFI)". If necessary, T_{SSF} is set.

(4) If in state "Monitoring" the FSM for the CS moves to the state "Waiting for End of Temporary Connection (MON)". If necessary, a guard timer T_{SSF} is set.

On receipt of this operation in the FSM for CS state "Waiting for Instructions" or "Monitoring", the SSP has to perform the following actions:

- Reset the T_{SSF} (optional)

NOTE: This "optional" means that the application timer T_{SSF} is optionally set. Whether it is used or not is network operator dependent. But it must be synchronized with $T_{SCF\text{-}SSF}$ in the SCSM.

- Route the call to assisting SSF or SRF using "assistingSSPIPRoutingAddress".

- The SSF FSM goes to state "Waiting for End of Temporary Connection (WFI)" (e7).

On receipt of this operation in the SSF FSM state "Monitoring", the SSP has to perform the following actions:

- Route the call to assisting SSF or SRF using "assistingSSPIPRoutingAddress".

18.44.3.2 Error handling

Until the connection setup has been accepted (refer to ITU-T Recommendation Q.71 [17]) by the assisting SSF/SRF, all received failure indications from the network on the ETC establishment shall be reported to the SCF as ETC error ETCFailed (e.g., busy, congestion). Note that the operation timer for ETC shall be longer then the maximum allowed time for the signalling procedures to accept the connection.

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.45 EventNotificationCharging procedure

18.45.1 General description

This operation is used by the SSF to report to the SCF the occurrence of a specific charging event type as requested by the SCF using the "RequestNotificationChargingEvent" operation. The operation supports the options to cope with the interactions concerning the charging (refer to ETS 300 374-1 [10], annex B "Charging scenarios").

As several charging events may occur during a connection configuration a possibility exists for the EventNotificationCharging operation to be invoked on multiple occasions. For each connection configuration EventNotificationCharging may be used several times.

18.45.1.1 Parameters

- eventTypeCharging:
  This parameter indicates the charging event type which has occurred. Its content is network operator specific, which may be "charge pulses" or "charge messages".

- eventSpecificInformationCharging:
  This parameter contains charging related information specific to the event. Its content is network operator specific.

- legID:
  This parameter indicates the leg on which the charging event type applies.

- monitorMode:
  This parameter indicates how the charging event is reported. When the "monitorMode" is "interrupted", the event is reported as a request, if the "monitorMode" is "notifyAndContinue the event is reported as a notification. The "monitorMode" "transparent" is not applicable for the EventNotificationCharging operation.

- eventTypeTariff:
  This parameter indicates the charging event type related to tariff determination. This charging event may include information about tariff -, price -, tariff/price-acknowledgement - or tariff/price-acknowledgement timer expiry.

- eventSpecificInformationTariff:
  This parameter contains charging tariff related information specific to the event. It may include the ChargingTariffInformation -, ChargingPriceInformation -, or ChargingAcknowledgementInformation sub-parameter.

18.45.2 Invoking entity (SSF)

18.45.2.1 Normal procedure

SSF Preconditions:

(1) A control relationship exist between the SCF and the SSF.

(2) A charging event has been detected that is requested by the SCF.

SSF Postcondition:

(1) No FSM state transition,

The SSF-FSM is in any state except "idle". This operation is invoked if a charging event has been detected that is requested by the SCF. The detected charging event can be caused by: a) another SLPI or b) another exchange. Irrespective of the charging event cause, the SSF performs one of the following actions on occurrence of the charging event (according the corresponding monitorMode):

Interrupted:

Notify the SCF of the charging event using "EventNotificationCharging" operation: do not process the event, but discard it. However, call and existing charging processing will not be suspended in the SSF.

NotifyAndContinue:

Notify the SCF of the charging event using "EventNotificationCharging", and continue processing the event or signal.

18.45.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.45.3 Responding entity (SCF)

18.45.3.1 Normal procedure

SCF Precondition:

(1) A "RequestNotificationChargingEvent" has been sent at the request of a SLPI and the SLPI is expecting an "EventNotificationCharging" from the SSF.

SCF Postcondition:

(1) No FSM state transition.

On receipt of this operation the SLPI which is expecting this notification can continue. If the corresponding monitor mode was set by the SLPI to **Interrupted** the SLPI prepares instructions for the SSF if necessary

18.45.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.46 EventReportBCSM procedure

18.46.1 General description

This operation is used to notify the SCF of a call related event previously requested by the SCF in an "RequestReportBCSMEvent" operation. The monitoring of more than one event could be requested with a "RequestReportBCSMEvent" operation, but each of these requested events is reported in a separate "EventReportBCSM" operation.

18.46.1.1 Parameters

- eventTypeBCSM:
  This parameter specifies the type of event that is reported.

- eventSpecificInformationBCSM:
  This parameter indicates the call related information specific to the event.

  For "CollectedInfo" it will contain the "calledPartyNumber".

  For "AnalysedInfo" it will contain the "calledPartyNumber".

  For "RouteSelectFailure" it will contain the "FailureCause", if available.

  For O- or T-Busy it will contain the "BusyCause", if available.

  For O- or T-NotReachable it will contain the "notReachableCause", if available.

  For O- or T-NoAnswer it will be empty.

  For O- -Answer it will contain the "BackwardGVNS", if available,

  For T-Answer it will be empty.

  For O- or T-MidCall it will contain "connectTime", if available as well as the indication, which specific mid call event has been detected. The latter is necessary only in case the network provider has defined different mid call events which are interesting to be monitored in the context of IN. In case of monitoring on specific control codes,the "iNServiceControlCode" contains the detected control code.

  For O- or T-Disconnect it will contain the "releaseCause" and/or "connectTime", if available.

  For O- Abandon and T_Abandon it will contain the "abandonCause" , if available.

  The connect time, if available, indicates the duration between the received answer indication from the called party side and the release of the connection in units of 100 ms,

  For O- or T-Suspended, OriginationAttempt, OriginationAttemptAuthorized, TerminationAttemptAuthorized, O- or T-ReAnswer, FacilitySelectedAndAvailable, CallAccepted, it will be empty.

- legID:
  This parameters indicates the party in the call for which the event is reported. SSF will use the option "ReceivingSideID" only. Refer to RequestReportBCSMEvent for the applied leg numbering.

  The "legID" parameter shall always be included for the events O-MidCall, O-Disconnect, T-MidCall and T-Disconnect.

- miscCallInfo:
  This parameter indicates Detection Point (DP) related information.

  - messageType:
    This parameter indicates whether the message is a request, i.e. resulting from a "RequestReportBCSMEvent" with monitorMode = interrupted, or a notification, i.e. resulting from a "RequestReportBCSMEvent" with "monitorMode" = "notifyAndContinue".

18.46.2 Invoking entity (SSF)

18.46.2.1 Normal procedure

SSF Preconditions:

(1) The FSM for CS is in the state "Monitoring", or in a User Interaction monitoring state (WfEoUI(MON)/WfEoTC(MON)) or
the FSM for CS may be in state "Waiting for Instructions" if the O/TDisconnect DP is armed and encountered, or
the FSM for CS may be in any state, except Idle if the O/TAbandon DP or MidCall DP immediate report is armed and encountered.

(2) The BCSM proceeds to an EDP that is armed.

SSF Postconditions:

(1) The FSM for CS stays in the state "Monitoring" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested.

(2) The SSF for CS moves to the state "idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested. If this was the last CS within the CSA, also the FSM for CSA returns to idle.

(3) If the message type was request, the FSM for CS moves to the state "Waiting for Instructions" if the FSM for CS was in the state "Monitoring". If user interaction is ongoing the FSM for CS moves to a User Interaction waiting for instructions state (WfEoUI(WFI)/WfEoTC(WFI)). Call processing is interrupted.

18.46.2.2 Error handling

In case the message type is request, on expiration of T_{ssf} before receiving any operation, the SSF aborts the interaction with the SCF and the call is given final treatment, e.g. a final announcement.

Operation related error handling is not applicable, due to class 4 operation.

18.46.3 Responding entity (SCF)

18.46.3.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SSF and the SCF.

(2) The FSM for CS is in the state "Preparing CS Instructions", substate "Waiting for Notification or Request".

SCF Postconditions:

(1) The FSM for CS stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested, or

The FSM for CS moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested, or

The FSM for CS moves to the state "Preparing CS Instructions" if the message type was request.

(2) The event is reported to a SLPI, based on the dialogue ID. The SCF will prepare SSF or SRF instructions in accordance with the SLPI.

18.46.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.47 EventReportBCUSM procedure

18.47.1 General description

This operation is used to notify the SCF of a call unrelated event previously requested by the SCF in an "RequestReportBCUSMEvent" operation. The monitoring of more than one event could be requested with a "RequestReportBCUSMEvent" operation, but each of these requested events is reported in a separate "EventReportBCUSM" operation.

18.47.1.1 Parameters

- eventTypeBCUSM:
  This parameter specifies the type of event that is reported.

- eventSpecificInformationBCUSM:
  This parameter indicates the call unrelated information specific to the event.

  For "ComponentReceived" it may contain "componentReceivedInfo" providing additional information about the component, but not the component itself.

  For "Association Release Requested" it may contain the "associationReleaseInfo" and the "releaseCause", if available.

- miscCallInfo:
  This parameter indicates DP related information.

  - messageType:
    This parameter indicates whether the message is a request, i.e. resulting from a "RequestReportBCUSMEvent" with monitorMode = interrupted, or a notification, i.e. resulting from a "RequestReportBCUSMEvent" with "monitorMode" = "notifyAndContinue".

- cUApplicationInd:
  This parameter identifies the triggered application (case service ASE located in the CUSF). It shall indicate the operation code of the application specific operation for which a DP was armed. Two type of values shall be supported: object IDs for standardized applications and local values for non-standardized applications.

- legID:
  This parameter indicates the party in the association for which the event shall be reported. SCF will use the option "sendingSideID" only. Refer to RequestReportBCUSMEvent for the applied leg numbering.

  The "legID" parameter shall always be included for the events Component_Received and Association_Release_Requested events.

18.47.2 Invoking entity (CUSF)

18.47.2.1 Normal procedure

CUSF Preconditions:

(1) The CUSF FSM is in the state c: Monitoring.

(2) The BCUSM proceeds to an EDP that is armed.

CUSF Postconditions:

(1) The CUSF FSM stays in the state c: Monitoring, if the message type was notification and there are still EDPs armed.

(2) The CUSF FSM moves to the state a: Idle, if the message type was notification and there are no more EDPs armed.

(3) The CUSF FSM moves to the state b: Waiting for Instructions, if the message type was request. Bearer independent connection processing is interrupted.

18.47.2.2 Error handling

In case the message type is request, on expiration of TCUSF before receiving any operation, the CUSF aborts the interaction with the SCF and the connection oriented bearer independent connection is released.

Operation related error handling is not applicable, due to class 4 operation.

18.47.3 Responding entity (SCF)

18.47.3.1 Normal procedure

SCF Preconditions:

(1) The FSM for CUSF is in the state N2: Preparing CUSF Instructions, substate N2.2: Waiting for Notification or Request.

SCF Postconditions:

(1) The FSM for CUSF stays in the substate N2.2: Waiting for Notification or Request, if the message type was notification and there are still EDPs armed, or

the FSM for CUSF moves to the state N1: Idle, if the message type was notification and there are no more EDPs armed, or

the FSM for CUSF moves to the substate N2.1: Preparing CUSF Instructions, if the message type was request.

(2) The event is reported to a SLPI, based on the dialogue ID. The SCF will prepare CUSF instructions in accordance with the SLPI.

18.47.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.48 Execute procedure

18.48.1 General description

The "Execute" operation performs a sequence of execution steps, according to a predefined method, using input information and returns result information.. Each step is either a DAP operation (that could be an execute operation), the execution of an algorithm or a decision test.

The "Execute" operation is used to execute a method on an Entry of the SDF resident DIT and to return selected information determined from the result of the method. A method is used to encapsulate a sequence of complex SDF operations, into a single SCF-SDF operation. It does this by associating a data access script with the entry method in the SDF. This script can involve one or more of any operations that can be invoked on the SCF-SDF interface, along with additional logic required to make subsequent operations. This additional logic may include decision making and data manipulation. The script may use part of the input value parameter as a parameter of any of the internal operations invoked by the script.

The SCF continues to provide service specific logic and to command CCFs in the SSF.

18.48.1.1 Parameters

- object:
  This parameter identifies an entry of the SDF in the Directory Information Tree (DIT) from_/ on which the method is to be executed. The use of this parameter is defined in ITU-T Recommendation X.501 [37] clause 9.

- method-id
  This parameter identifies the method which is to be executed on the entry. The value of this parameter is identified from the ASN.1 description of the method defined in the data schema. It is unique for the associated object.

- inputAssertions
  This parameter provides a set of attribute values which are used as an input to the method execution

- input-Attributes
  This field identifies the attributes which may be submitted as input to the method execution.

- specificInput
  This parameter identifies the additional information which is required on the entry in order to perform the method. The type of this parameter is identified by the ASN.1 description of the method defined in the data schema.

- outputAssertions
  This parameter contains attribute values returned as a result of the method execution

- output-Attributes
  This field identifies the attributes which may be returned as output to the method execution.

- specificOutput
  This parameter contains information returned as a result of the operation executed when the method on the entry is invoked. The type of this parameter is identified by the ASN.1 description of the method defined in the data schema.

- CommonArguments
  All ITU-T Recommendation X.500 [36] operations contain common arguments such as security information. The use of these arguments is defined in ITU-T Recommendation X.511 [39] Clause 7.3.

18.48.2 Invoking entity (SCF)

18.48.2.1 Normal procedure

SCF Preconditions:

(1) SCSM: "SDF Bound" or "Wait for Subsequent Requests"

SCF Postconditions:

(1) SCSM: "SDF Bound"

When the SCSM is in the state "Wait for Subsequent Requests" an internal event ((e2) Request_to_SDF) occurs when SL needs to execute an entry method on the SDF. Until the application process indicates with a delimitor that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. When the delimitor is received, the operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response ((E5) Response_from_SDF_with_Bind or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM moves to state "Idle", the Execute operation is discarded. In the State "SDF Bound", the response of the Execute operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). The response from the SDF may be either the result of the Execute operation or an error.

When the SCSM is in the state "SDF Bound" an internal event ((e6) Request_to_SDF) occurs when SL needs to execute an entry method on the SDF. This event causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response ((E7) Response_from_SDF) to the Execute operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the Execute operation or an error.

18.48.2.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] clause 12. The TC services that are used for reporting operating errors are described in subclause 2.2.2.

18.48.3 Responding entity (SDF)

18.48.3.1 Normal procedure

SDF Preconditions:

(1) SDSM: "SCF Bound" or "Bind Pending"

SDF Postconditions:

(1) SDSM "SCF Bound"

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a "Execute" operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "Execute" operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before retrieving the data as specified in the operation parameters, the SDF takes the following actions:

- the SDF verifies that the object accessed by the request exists;
- the SDF verifies that the method referenced in the operation exists in the object, and that the argument is of the correct type.
- the SDF verifies that the user on behalf of whom the request is performed has sufficient access rights to execute the method on the entry.

If these actions are successfully executed the SDF executes the data access script associated with the entry method in the SDF. Before each internal operation initiated by the script is performed, the SDF takes the following actions:

- the SDF may verify that the user on behalf of whom the request is performed has sufficient access rights to perform the operation;
- the SDF may verify that object, attributes or methods on which an operation should be performed exists in the DIT.

If all of the specified actions indicated above are successfully executed, the SDF returns the result of the script to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

18.48.3.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] clause 12.

If basic-access-control is in effect for the entry on which the method is being executed, the following sequence of access controls applies.

- ExecuteMethod permission is required to the entry on which the method is being executed. If permission is not granted, the operation fails in accordance with ITU-T Recommendation X.511 [39]/7.11.3. In this case a SecurityError with problem insufficientAccessRights or noInformation shall be returned.
- Before each internal operation initiated by a data access script associated with entry method is performed basic access control is applied as if the operation had been invoked on the SCF-SDF interface. If any of these internal operations fails, the entire operation fails and an executionError with problem executionFailure shall be returned.

If the operation fails due to failure of any other data access logic, an executionError with problem executionFailure shall be returned.

If the operation fails due to an incorrect type or value of the input-value an executionError with problem missinginputValues shall be returned.

If a request is made to execute a method on a non existent entry a NameError with problem noSuchObject shall be returned.

The TC services that are used for reporting operating errors are described in clause 18.

18.49 FurnishChargingInformation procedure

18.49.1 General description

This operation is used to request the SSF to generate, register a call record or to include some information in the default call record. The registered call record is intended for off-line charging of the call. A possibility exists for the FurnishChargingInformation (FCI) operation to be invoked on multiple occasions. FCI could be applied at the beginning of the call in order to request to start call record generation. In addition FCI can also be applied at the end of the call or connection configuration (e.g., for follow-on calls). In this case FCI is used to include charge related information into the call record which was started at the beginning of the call. The charging scenarios supported by this operation are: 2.2, 2.3 and 2.4 (refer to ETS 300 374-1 [10], annex B "Charging scenarios")

18.49.1.1 Parameters

fCIBillingChargingCharacteristics:

This parameter indicates billing and/or charging characteristics. It consists of network operator specific parts and a part for the determination of tariff/price. Depending on the applied charging scenario the following information elements can be included. Refer to ETS 300 374-1 [10], annex B "Charging scenarios":

- complete charging record (scenario 2.2)
- charge party (scenario 2.3)
- charge level (scenario 2.3)
- charge items (scenario 2.3)
- correlationID (scenario 2.4)

The following sub-parameters may be included:

fCIBCCcs1:

The content of this parameter is network operator specific and replaces the fCIBillingChargingCharacteristics for CS1.

fCIBCCseq:

The content of this parameter consists of two parts:

fCIBCC:

The content of this sub-parameter is network operator specific.

tariff:

This parameter specifies the current tariff/price to be applied to the call. It may include either ChargingTariffInformation -, or ChargingPriceInformation sub-parameter.

18.49.2 Invoking entity (SCF)

18.49.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exist between the SCF and the SSF.

(2) An SLPI has determined that a "FurnishChargingInformation" has to be sent by the SCF.

SCF Postconditions:

(1) No FSM state transition,

(2) SLPI execution may continue.

The SCSM FSM is in state "Preparing SSF instruction" , "Waiting For Notification or Request", or is in state "Queuing FSM". This operation is invoked by the SCF if a SLPI results in the request of creating a call record to the SSF or to include some billing or charging information into the default call record. In the case of call queuing, this operation may contain information pertaining to the initiation of queuing or the call queuing time duration for call logging purpose. This causes no SCSM FSM state transition.

18.49.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.49.3 Responding entity (SSF)

18.49.3.1 Normal procedure

SSF Preconditions:

(1) FSM for CS State "Waiting for Instructions" or
FSM for CS State "Waiting for End of User Interaction" or
FSM for CS State "Waiting for End of Temporary Connection" or
FSM for CS State "Monitoring" or
Assisting/hand-off SSF-FSM State "Waiting for Instructions".

SSF Postcondition:

(1) No FSM state transition.

On receipt of this operation the SSF performs actions to create the call record according the off line charging scenario which is applicable using the information elements included in the operation:

registers the complete call record included in the operation;

generates and registers a call record according the information (charge party, charge level/tariff, charge items);

include the information received "correlationID" in the default call record which is generated and, registered by default at the SSF.

By means of a parameter at the "FurnishChargingInformation" operation the SCF can initiate the pulse metering function of the SSF.

In that case the SSF shall generate meter pulses according to the applicable charging level/tariff, account and record them.

The SSF records charge related data like for example the call duration, begin time stamp or end time stamp. Additionally the SSF records further data if required.

The charging level/tariff can be determined by

a) the SCF, or

b) the SSF, or

c) a succeeding exchange, or

d) the post processing function.

If a) applies the charging level is included in the "FurnishChargingInformation" operation.

If b) applies the SSF shall determine the charging level based on the corresponding parameters contained in the operation.

If c) applies either the "FurnishChargingInformation" operation contains the corresponding parameters indicating that the charging level shall be determined in a succeeding exchange or the SSF detects during the determination of the charging level based on the provided parameters that the charging level shall be determined in a succeeding exchange.

The SSF can either account received pulses or convert any charging messages received from the B-side to pulses. In both cases, the accumulated pulses are included when the IN call record is generated or ignored.

18.49.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.50 HandlingInformationRequest procedure

18.50.1 General description

This operation is used when a SCF controlling a "call" has not sufficient information to process the call (e.g. routeing, announcement) and request assistance from another SCF having knowledge on how to proceed with the call. The call could be a user request with only one user involved.The operation initiates the dialogue between two SCFs and to provide the supporting SCF with the context of the call so that it can help the controlling SCF in the processing of the call.

18.50.1.1 Parameters

- callingPartyNumber:
  This parameter, if present, is used to identify the calling party for the call (see EN 300 356-1 [9] Calling Party Number). It may be used for applications such as customized routeing service, where the supporting SCF needs to now the identity of the calling party to give routeing information to the controlling SCF.

- locationNumber
  This parameter contains information on the location of the user. It conveys the geographical address for mobility services, see EN 300 356-1 [9]. It is used when the "callingPartyNumber" does not contain any information about the location of the calling party. It may be used by the supporting SCF in case of a location dependent routeing.

- calledPartyNumber:
  This parameter is used to identify the called party in the forward direction, see EN 300 356-1 [9]. This parameter is present if the SL has been able to recognize a called party number in the information provided by the user. When the format has not be recognized the called party number can be conveyed in the parameter "dialledDigits".

- dialledDigits:
  This parameter is used to convey information collected from the user through user interaction procedure or in the set-up phase and that have not been recognized as information to be included in another parameter.

- redirectingPartyID:
  This parameter indicates the last directory number the call was redirected from

- redirectionInformation:
  See ITU-T Recommendation Q.763 [22] Redirection Information signalling information.

- callingpartyBusinessGroupID:
  This parameter gives the ID for the business group the user belongs to.

- originalCalledPartyID:
  See EN 300 356-1 [9] Original Called Party Number signalling information.

- numberOfCallAttempts:
  This parameter gives the number of previous call attempts before the one that is currently handled. The number of call attempts is considered within the same SLPI.

- highLayerCompatibility:
  This parameter indicates the type of high layer compatibility, which will be used to determine the ISDN-teleservice of a connected ISDN terminal. For encoding DSS1 (EN 300 403-1 [11]) is used. The highLayerCompatibility can also be transported by ISUP within ATP (see ITU-T Recommendation Q.763 [22]) parameter.

- bearerCapability:
  This parameter indicates the type of bearer capability connection or the transmission medium requirements to the user.

- invokedSupplementaryServices:
  This parameter contains the supplementary service that has been invoked by the user. Only information available to the controlling SCF can be provided.

- activeSupplementaryServices:
  This parameter contains the list of supplementary services that have been activated by the user. These activated supplementary services can have an impact on the call. Only information available to the SCF can be provided.

- causeOfLastCallFailure:
  This parameter gives the reason of the failure of the last call, if any. This last call is considered within the same SLPI.

- userInteractionMode:
  This parameter conveys the type of user interaction modes that are available in the invoking network.

- callingPartysCategory:
  See EN 300 356-1 [9] Calling Party's Category signalling information. It indicates the type of calling party (e.g. operator, pay-phone, ordinary user)..

- requestedType:
  This parameter is used to identify the context in which the operation will be used. The list of allowed values (and the associated semantic) is part of the definition of each SL type. The scope of the RequestedType IE is local to an AgreementID

- securityParameter:
  This is an optional parameter that conveys security related

18.50.2 Invoking entity (controlling SCF)

18.50.2.1 Normal procedure

SCF Precondition:

(1) A relationship has been requested by controlling SCF if it is the first occurance of the operation, otherwise the relationship has been already eestablished between the two SCFs

(2) In the first case, the SCF FSM is in state Preparing request for Assistance"., in the second case it is in state "Assisted Mode".

SCF Postcondition:

(1) The SCF FSM moves to the state "Waiting for Bind Result" from the state "Preparing request for Assistance" " or it remains in the "Assisted Mode" state, if it was already there.

(2) If SCF FSM has moved to the state "Waiting for Binds result state", it moves to "Assisted mode" state as soon as a positive result to SCF Bind operation is received, or to "Idle" state in case of a negative result

The SCF provides parameters, that depend on the SL invoked. The list of parameters that are needed depends on the agreement- referred with SCF Bind operation previuosly issued . When mandatory information for a given SL is missing, the controlling SCF is responsible to conduct the necessary actions to get the information before sending of the"handlingInformationRequest" operation.

18.50.2.2 Error Handling

If the supporting SCF is not accessible, the call is given final treatment which is SL dependent.

If the calling user abandons after the sending of the "handlingInformationRequest" operation, then the SCF FSM moves the state "Preparing for SCFUnbind request" and the SCF aborts the SCF-SCF relationship by means of an abort to TC. Note that TC will wait until the first response message from the SCF has been received before it sends an abort to the SCF.

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.50.3 Responding entity (supporting SCF)

18.50.3.1 Normal procedure

SCF Precondition:

(1) The SCF FSM is in the state "Processing SCF Bind" or in the state "Assisting Mode" or it has already successfully processed a previous HandlingInformationRequest.

SCF Postcondition:

(1) The SCSM remains in the state "Assisting Mode".

The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the agreement, i.e. the requested IN service, itself.

18.50.3.2 Error Handling

If the supporting SCF receives a "handlingInfromationRequest" operation while it is in the Assisting mode and a "handlingInformationResult is already pending for a pevious "handlingInformationRequest" operation, then a "taskRefused" error is returned to the controlling SCF.

If the "handlingInformationRequest" operation is rejected the SCSM remains in the state "Idle". The maintenance function is informed and no SLPI is invoked.

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.51 HandlingInformationResult procedure

18.51.1 General description

This operation is used by the supporting SCF to answer operations previously issued by the controlling SCF. Information contained in the "handlingInformationResult" operation can either be used to process the call (e.g. routeing, announcement) that has initiated the SCF-SCF relationship, or to indicate to controlling SCF when it should contact the supporting SCF to receive instructions.

18.51.1.1 Parameters

- routingAddress:
  This parameter contains indications on how the call should be handled. These indication can be a request to forbid the call or the list of called party numbers (see EN 300 356-1 [9]) towards which the call is to be routed.The encoding of the list is defined in ITU-T Recommendation Q.763 [22]. If no other agreements exist, the numbers from the list shall be used sequentially.

- highLayerCompatibility:
  This parameter indicates the type of high layer compatibility, which will be used to determine the ISDN-teleservice of a connected ISDN terminal. For encoding DSS1 (EN 300 403-1 [11]) is used. The highLayerCompatibility can also be transported by ISUP within ATP (see ITU-T Recommendation Q.763 [22]) parameter.

- supplementaryServices:
  This parameter contains the list of supplementary services that have been activated by the user.

- preferredLanguage:
  The parameter gives the language that should preferrably used in user interactions.

- carrier:
  This parameter indicates the identity of the carrier to use for the call.

- callingPartyNumber:
  This parameter, if present, is used to identify the calling party for the call (see EN 300 356-1 [9] Calling Party Number). The parameter may be used to force the signalling information to a certain value.

- originalCalledPartyID:
  See EN 300 356-1 [9] Original Called Party Number signalling information.

- redirectingPartyID:
  This parameter indicates the last directory number the call was redirected from

- redirectionInformation:
  See ITU-T Recommendation Q.763 [22] Redirection Information signalling information.

- callingPartysCategory:
  See EN 300 356-1 [9] Calling Party"s Category signalling information. It indicates the type of calling party (e.g. operator, pay-phone, ordinary user)..

18.51.2 Invoking entity (supporting SCF)

18.51.2.1 Normal procedure

SCF Precondition:

(1) A dialogue with the controlling SCF has been established.

(2) The supporting SCF has previously received a "handlingInformationRequest" operation

(3) The SCF FSM is in the state "Assisted Mode"

SCF Postcondition:

(1) The SLPI waits for operations coming from the controlling SCF or the SLPI can be ended.

(2) The SCF FSM remains in the state "Assisting Mode"

This operation is invoked by the supporting SCF when the SL is able to provide instructions to the controlling SCF on how to process the call.

Information contained in the "handlingInformationResult" operation can be used by the controlling SCF to build the operations needed to set-up the call and/or to control it.

18.51.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.51.3 Responding entity (controlling SCF)

18.51.3.1 Normal procedure

SCF Precondition:

(1) A "handlingInformationRequest" operation has been sent.

(2) The SCF FSM is in the state "Assisted Mode”.

SCF Postcondition:

(1) According to the information received, the SCF performs call processing actions and/or starts call monitoring functions.

(2) The SCF FSM moves to the state "Preparing SCF Unbind request" from the state "Assisted Mode" if no more assistance is needed or it remains in the the state "Assisted Mode, if a further Handling Information Result reception is still pending.

If the "routingAddress" parameter is present, it is used to handle the call. If the call has to be routed the other parameters contained in the operation can be used to set-up the call.

The contents of the parameter can be memorized by the SCF and/or store in the SDF. The contents of the parameter can be kept for a longer time than the relationship duration.

18.51.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.52 in-directoryBind procedure

18.52.1 General description

The ITU-T Recommendation X.500 [36] "directoryBind" operation is used by the SCF to create an authenticated association between an SCF and an SDF on behalf of the end user. It carries the authentication information of the end user if any. For a full description of the directoryBind operation, see 8.1/ITU-T Recommendation X.511 [39].

18.52.1.1 Parameters

See ITU-T Recommendation X.511 [39] subclauses 8.1.2 and 8.1.3.

18.52.2 Invoking Entity (SCF)

18.52.2.1 Normal procedure

SCF Preconditions:

- SCSM-SDF: "Idle".

SCF Postconditions:

1) SCSM-SDF: "SDF Bound" in case of success.

2) SCSM-SDF: "Idle" in case of failure.

When the SCSM-SDF is in the state "Idle" and a need of the SL to interrogate an SDF exists, an internal event occurs. This event, called (e1) Bind_Request, causes a transition to the state "Wait for Subsequent Requests" and other operations are awaited. Until the application process has not indicated by a delimitor that the Bind should be sent, the SCSM-SDF remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimitor causes a transition to the state "Wait for Bind result" through the internal transition (e3) Request_to_SDF_with_Bind. The operation is sent to the SDF. The SCSM-SDF waits for the response from the SDF. The reception of the response [(E5) Response_from_SDF_with _Bind] to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" if the result of the Bind operation is positive. Otherwise the reception of an error [(E4) Bind_Error] moves back the SCSM-SDF to the state "Idle".

18.52.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.511 [39] subclause 6.1.4 and the TCAP services that are used for reporting operating errors are described in clause 18.

18.52.3 Responding Entity (SDF)

18.52.3.1 Normal procedure

SDF Preconditions:

- SDSM-SCF: "Idle".

SDF Postconditions:

1) SDSM-SCF: "SCF Bound" (success).

2) SDSM-SCF: "Idle" (failure).

The SDSM-SCF is initially in the state "Idle". After accepting the external event (E1) Bind_from_SCF caused by the reception of a "directoryBind" operation from the SCF, a transition to state "Bind Pending" occurs. The SDF performs the Bind operation according to the contents of the directoryBind argument. Once the SDF has completed the "directoryBind" operation, the result or error indication is returned to the SCF. The SDF returns to the state "Idle" if the Bind fails or to the state "SCF Bound" if the Bind is successful. Should the Bind request succeed, the result returned may consist of credentials of the DirectoryBindResult. These credentials allow the user to establish the identity of the Directory. They allow information identifying the DSA (that is directly providing the Directory service) to be conveyed to the DUA. The credentials are of the same form as those supplied by the user.

18.52.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and ITU-T Recommendation X.511 [39] subclause 8.1.4 and the TCAP services that are used for reporting operating errors are described in clause 18.

18.53 In-directoryUnbind procedure

18.53.1 General description

The ITU-T Recommendation X.500 [36] "Unbind" operation is used by the SDF to end an authenticated association between an SCF and an SDF on behalf of the end user. For a full description of the Unbind operation, see ITU-T Recommendation X.511 [39] subclause 8.2.

18.53.1.1 Parameters

None

18.53.2 Invoking entity (SCF)

18.53.2.1 Normal procedure

SCF Preconditions:

(1) SCSM: "SDF Bound"

SCF Postconditions:

(1) SCSM: "Idle"

The SCSM has previously initiated a successful Bind operation to the SDF directory. It is in state "SDF Bound". The SL determines that the authenticated access to the SDF is to be terminated. It issues an Unbind operation ((e8) Unbind_request) that causes the SCSM to transit back to the state "Idle".

18.53.2.2 Error handling

The "Unbind" operation does not have operation related errors.

18.53.3 Responding entity (SDF)

18.53.3.1 Normal procedure

SDF Preconditions:

(1) SDSM: "SCF Bound"

SDF Postconditions:

(1) SDSM: "Idle"

A Bind operation was previously issued and the SDSM is in State "SCF Bound" waiting for a request from the SCF and/or performing an operation. The reception of the Unbind operation causes a transition to State "Idle" with the transition (E5) Unbind_from_SCF.

18.53.3.2 Error handling

The "Unbind" operation does not have operation related errors.

18.54 In-DSAShadowUnbind procedure

18.54.1 General Description

The ITU-T Recommendation X.500 [36] "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525 [42].

18.54.1.1 Parameters

None.

18.54.2 Supplier entity (SDF)

18.54.2.1 Normal Pocedure

20.54.2.1.1 Supplier-initiated DSAShadowUnbind

SDF Preconditions:

(1) SDSM-ShM: "SDF Bound"

(2) SDSM-ShM: "Bound with CoordinateShadow Sent"

(3) SDSM-ShM: "Wait for Coordination Result"

(4) SDSM-ShM: "Wait for Update"

(5) SDSM-ShM: "Wait for Update Confirmation"

SDF Postconditions: (1) SDSM-ShM: "Idle"

The SDSM-ShM has previously initiated a successful "DSAShadowBind" operation to the consumer SDF. It is in either of the states "SDF Bound", "Bound with CoordinateShadow Sent",Wait for Coordination Result", "Wait for Update", or "Wait for Update Confirmation". It determines that the "authenticated association" established between two SDFs is to be terminated (e.g., during a userís release procedure) and issues a " DSAShadowUnbind " operation ((e124), (e15), (e17), (e20) or (e22) SDF_Unbind) that causes the SDSM-ShM to transit back to the state

"Idle".

20.54.2.1.2 Consumer-initiated DSAShadowUnbind

SDF Preconditions:

(1) SDSM-ShM: "SDF Bound"

(2) SDSM-ShM: "Wait for RequestShadow Result"

(3) SDSM-ShM: "Wait for Update"

(4) SDSM-ShM: "Wait for Update Confirmation"

SDF Postconditions:

(1) SDSM-ShM: "Idle"

A "DSAShadowBind" operation has previously been issued and the SDSM-ShM is in either of the states "SDF Bound", "Wait for RequestShadow Result", "Wait for Update", or "Wait for Update Confirmation" waiting for a request/response from the consumer SDF or performing an operation. The reception of the " DSAShadowUnbind " operation causes a transition to the state "Idle" with the transition SDF_Unbind ((E64), (E8), (E12), or (E15)).

18.54.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause18.1.

18.54.3 Consumer entity (SDF)

18.54.3.1 Normal Procedure

20.54.3.1.1 Supplier-initiated DSAShadowUnbind

SDF Preconditions:

(1) SDSM-ShC: "SDF Bound"

(2) SDSM-ShC: "Wait for Coordination Result"

(3) SDSM-ShC: "Wait for Update"

(4) SDSM-ShC: "Wait for Update Confirmation"

SDF Postconditions:

(1) SDSM-ShC: "Idle"

A "DSAShadowBind" operation has previously been issued and the SDSM-ShC is in either of the states "SDF Bound", "Wait for Coordination Result", "Wait for Update", or "Wait for Update Confirmation" waiting for a request from the supplier SDF or performing an operation. The reception of the " DSAShadowUnbind " operation causes a transition to the state "Idle" with the transition SDF_Unbind ((E64), (E9), (E12), or (E14)).

20.54.3.1.2 Consumer-initiated DSAShadowUnbind

SDF Preconditions:

(1) SDSM-ShC: "SDF Bound"

(2) SDSM-ShC: "Wait for RequestShadow Result"

(3) SDSM-ShC: "Wait for Update"

(4) SDSM-ShC: "Wait for Update Confirmation"

SDF Postconditions:

(1) SDSM-ShC: "Idle"

The SDSM-ShC has previously initiated a successful "DSAShadowBind" operation to the supplier SDF. It is in either of the states "SDF Bound", "Wait for RequestShadow Result", "Wait for Update", or "Wait for Update Confirmation". It determines that the "authenticated association" established between two SDFs is to be terminated (e.g., during a userís release procedure) and issues a "_DSAShadowUnbind " operation ((e84), (e10), (e13), or (e15) SDF_Unbind) that causes the SDSM-ShC to transit back to the state "Idle".

18.55 in-DSAUnbind procedure

18.55.1 General description

The in-DSAUnbind operation is used by the invoking SDF to end an authenticated chaining association between an invoking SDF and a responding SDF on behalf of the end user.

18.55.1.1 Parameters

None

18.55.2 Invoking entity (SDF)

18.55.2.1 Normal procedure

SDF Preconditions:

1) An "Unbind" request has been received by the invoking SDF indicating that the end user association which required operation chaining has been released.

2) SDSM-ChI: "SDF Bound"

SDF Postconditions:

1) SDSM-ChI: "Idle"

The SCSM-ChI has previously initiated a successful Bind operation to the responding SDF directory. It is in state "SDF Bound". The invoking SDF has received an indication that the that the authenticated chained access to the responding SDF is to be terminated. It issues an in-DSAUnbind operation ((e6) DSAUnbind_to_SDF) that causes the SCSM-ChI to transit back to the state "Idle".

18.55.2.2 Error handling

The "Unbind" operation does not have operation related errors.

18.55.3 Responding entity (SDF)

18.55.3.1 Normal procedure

SDF Preconditions:

1) SDSM-ChT: "SCF Bound"

SDF Postconditions:

1) SDSM-ChT: "Idle"

A dSABind operation was previously issued and the SDSM-ChT is in State "SDF Bound" waiting for a request from the invoking SDF and/or performing an operation. The reception of the in-DSAUnbind operation causes a transition to State "Idle" with the transition (E5) DSAUnbind_from_SDF.

18.55.3.2 Error handling

The "Unbind" operation does not have operation related errors.

18.56 InitialDP procedure

18.56.1 General description

This operation is sent by the SSF after detection of a TDP-R in the BCSM, to request the SCF for instructions to complete the call.

18.56.1.1 Parameters

- serviceKey:
  This parameter identifies for the SCF unambiguously the requested IN service. It is used to address the correct application/SLP within the SCF (not for SCP addressing).

- calledPartyNumber:
  This parameter contains the number used to identify the called party in the forward direction, i.e. see EN 300 356-1 [9].

- callingPartyNumber:
  See EN 300 356-1 [9] Calling Party Number signalling information.

- callingPartyBusinessGroupID:
  See ITU-T Recommendation Q.1290 [29]. The SCF can use this IE to select SLPs based on the group and for authorization purposes. The network operators can specify that this IE should be used if their particular network has the information available.

- callingPartysCategory:
  See EN 300 356-1 [9] Calling Party Category signalling information.

- cGEncountered:
  See ITU-T Recommendation Q.1290 [29]

- iPSSPCapabilities:
  See ITU-T Recommendation Q.1290 [29].

- iPAvailable:
  See ITU-T Recommendation Q.1290 [29].

- locationNumber:
  This parameter is used to convey the geographical area address for mobility services, see ITU-T Recommendation Q.762 [21]. It is used when "callingPartyNumber" does not contain any information about the geographical location of the calling party (e.g., origin dependent routeing when the calling party is a mobile subscriber).

- originalCalledPartyID:
  See EN 300 356-1 [9] Original Called Number signalling information.

- terminalType:
  See ITU-T Recommendation Q.1290 [29]. Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

- cause:
  See ITU-T Recommendation Q.1290 [29].

- iSDNAccessRelatedInformation:
  Carries the same information as the protocol element ISUP Access Transport parameter in EN 300 356-1 [9].

- iNServiceCompatibilityIndication:
  This parameter contains the ID for a class of IN services that has been triggered during the call. A class of IN services is defined as IN services which have the same compatibility characteristics.

- highlayerCompatibility:
  This parameter indicates the type of the high layer compatibility, which will be used to determine the ISDN - teleservice of a connected ISDN terminal. For encoding DSS1 (EN 300 403-1 [11]) is used. The highlayerCompatibility can also be transported by ISUP (e.g. within the ATP (see ITU-T Recommendation Q.763 [22]) parameter).

- serviceInteractionIndicators:
  This parameter contain indicators sent from the SSF to the SCF for control of the network based services at the originating exchange and the destination exchange.

- additionalCallingPartyNumber:
  The calling party number provided by the access signalling system of the calling user, e.g. provided by a PBX.

- forwardCallIndicators:
  This parameter indicates if the call shall be treated as a national or international call. It also indicates the signalling capabilities of the network access, preceding network connection and the preferred signalling capabilities of the succeeding network connection. The network access capabilities does not indicate the terminal type. For example, an ISPBX will have an ISDN type of access, but the end user terminal behind the ISPBX may be ISDN or non-ISDN.

- bearerCapability:
  This parameter indicates the type of the bearer capability connection or the transmission medium requirements to the user. It is a network option to select one of the two parameters to be used:

  - bearerCap:
    This parameter contains the value of the DSS1 Bearer Capability parameter (EN 300 403-1 [11]) in case the SSF is at local exchange level or the value of the ISUP User Service Information parameter (ITU-T Recommendation Q.763 [22]) in case the SSF is at transit exchange level.

    The parameter "bearerCapability" shall only be included in the "InitialDP" operation in case the DSS1 Bearer Capability parameter or the ISUP User Service Information parameter is available at the SSP.

    If two values for bearer capability are available at the SSF or if User Service Information and User Service Information Prime are available at the SSF the "bearerCap" shall contain the value of the preferred bearer capability respectively the value of the User Service Information Prime parameter.

  - tmr:
    The tmr is encoded as the Transmission Medium Requirement parameter of the ISUP according to ITU-T Recommendation Q.763 [22].

If two values for transmission medium requirement are available at the SSF or if Transmission Medium Requirement and Transmission Medium Requirement Prime are available at the SSF the "bearerCap" shall contain the value of the preferred transmission medium requirement respectively the value of the Transmission Medium Requirement Prime parameter.

- eventTypeBCSM:
  This parameter indicates the armed BCSM DP event, resulting in the "InitialDP" operation.

- redirectingPartyID:
  This parameter indicates the last directory number the call was redirected from.

- redirectionInformation:
  See ITU-T Recommendation Q.763 [22] Redirection Information signalling information.

- genericNumbers:
  The SSF may inform the SCF about the additional calling party number but also with the called number, the additional connected number, the additional original.called party number, the additional redirecting number and/or the additional redirection number.

- serviceInteractionIndicatorsTwo
  Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services and network based services, respectively between different IN based services.

- forwardGVNS:
  Identifies the originating service provider and provides information about the calling VPN user in terms of a customerID or a GVNS user group. The parameter will also carry routeing information for the terminating GVNS network.

- createdCallSegmentAssociation:
  This parameteridentifies for the SCF unambiguously the CSA instance in the SSF under SCF control. This CSA ID assigned by the SSF may be used to associate different CSA instances in the SSF.

- USIServiceIndicator:
  It indicates the SL requesting the UTSI information element. It may be used as a Triggering criteria at the SSF level..

- USIInformation:
  This parameter conveys information provided by the User dedicated to the SL. It is transparent at the SSF level.

- carrier:
  This parameter indicates the transit network(s) requested for the call. The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22].

- iMSI:
  IMSI of the mobile subscriber for which the service is invoked. For encoding see GSM 09.02 [14].

- subscriberState:
  The state of the mobile subscriber for which the service is invoked. The possible states are busy, idle and not reachable. For encoding see GSM 09.02 [14].

- locationInformation:
  This parameter indicates the whereabouts of the MS, and the age of the information defining the whereabouts. For encoding see GSM 09.02 [14].

- ext-BasicServiceCode:
  Indicates the Basic Service Code. For encoding see GSM 09.02 [14].

- callReferenceNumber:
  This parameter gives the call reference number assigned to the call by the CCF. For encoding see GSM 09.02 [14].

- mscAddress:
  This parameter gives the mscId assigned to the GMSC/MSC. For encoding see GSM 09.02 [14].

- calledPartyBCDNumber:
  This parameter contains the number used to identify the called party in the forward direction. It may also include service selection information, including * and # digits.

18.56.2 Invoking entity (SSF)

18.56.2.1 Normal procedure

SSF Preconditions:

(1) An event fulfilling the criteria for the DP being executed has been detected.

(2) Call gapping and SS7 overload are not in effect for the call, and the call is not to be filtered.

SSF Postcondition:

(1) A control relationship has been established if the DP was armed as a TDP-R. The FSM for CS moves to the State "Waiting for Instructions".

Following a trigger detection (due to the DP criteria assigned being met) related to an armed TDP in the BCSM caused by a call origination attempt, the SSF checks if call gapping, SS7 overload or service filtering are not in effect for the related call segment.

If these conditions are met, then the "InitialDP" operation is invoked by the SSF. The address of the SCF the "InitialDP" operation has to be sent to is determined on the base of trigger related data. The SSF provide as many parameters as available. In some cases, some parameters must be available (such as "callingPartyNumber" or "callingPartyCategory"). This is to be handled appropriately by the SSF in its trigger table (to know that such parameter are necessary for some triggering conditions) and in conducting the necessary action to get these parameters if they are not available (For instance if non-SS7 signalling is used, it may be possible to request the "callingPartyCategory" from a preceding exchange).

Otherwise, the call control is given back to the underlying network.

If the DP was armed as a TDP-R a control relationship is established to the SCF. The SSF application timer T_{SSF} is set when the SSF sends "InitialDP" for requesting instructions from the SCF. It is used to prevent excessive call suspension time.

18.56.2.2 Error handling

If the destination SCF is not accessible then the call is given final treatment.

On expiration of T_{SSF} before receiving any operation, the SSF aborts the interaction with the SCF and the call is given final treatment, e.g., routeing to a final announcement.

If the calling party abandons after the sending of "InitialDP", then the SSF aborts the control relationship by means of an abort to TC. Note that TC will wait until the first response message from the SCF has been received before it sends an abort to the SCF (see alsoclause 16).

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.56.3 Responding entity (SCF)

18.56.3.1 Normal procedure

SCF Precondition:

None.

SCF Postcondition:

(1) An SLPI has been invoked.

On receipt of "InitialDP" operation the SCSM moves from "Idle" to the state "Preparing SSF Instructions", a control relationship to the related SSF is created. A SLPI (SLPI) is invoked for processing the "InitialDP" operation based on the "serviceKey" parameter. By means of this control relationship, the SCF may influence the Basic Call Processing in accordance with the SL invoked.

The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the SLPI, i.e. the requested IN service, itself.

18.56.3.2 Error handling

If the "InitialDP" operation is rejected then the SCSM remains in "Idle". The maintenance function is informed and no SLPI is invoked.

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.57 InitialAssociationDP procedure

18.57.1 General description

This operation is sent by the CUSF to the SCF after detecting a valid trigger condition at any BCUSM DP (reported as TDP). This operation may either be sent after triggering during processing of an service ASE located in the CUSF or on receipt of an USI information, for which a valid trigger criteria was met.

18.57.1.1 Parameters

- serviceKey:
  This parameter identifies for the SCF unambiguously the requested IN service. It is used to address the correct application/SLP within the SCF (not for SCP addressing).

- cUApplicationInd:
  This parameter identifies the triggered application (case service ASE located in the CUSF). It shall indicate the operation code of the triggered application specific operation. Two type of values shall be supported: object IDs for standardized applications and local values for non-standardized applications.

- miscCallInfo:
  This parameter is a sequence of DP type (notification or request) and DP assignment (individual line, group based or office based) DP type and DP assignment is network operator optional.

- eventTypeBCUSM:
  This parameter indicates the armed BCUSM DP event, resulting in the "InitialAssociationDP" operation.

- calledPartyNumber:
  This parameter contains the number used to identify the called party in the forward direction, i.e. see EN 300 356-1 [9].

- callingPartyNumber:
  See EN 300 356-1 [9] Calling Party Number signalling information.

- callingPartySubaddress:
  See EN 300 403-1 [11].

- highlayerCompatibility:
  This parameter indicates the type of the high layer compatibility, which will be used to determine the ISDN - teleservice of a connected ISDN terminal. For encoding DSS1 (EN 300 403-1 [11]) is used.

- bearerCapability:
  This parameter indicates the type of the bearer capability connection or the transmission medium requirements to the user. It is a network option to select one of the two parameters to be used:

  - bearerCap:
    This parameter contains the value of the DSS1 Bearer Capability parameter (EN 300 403-1 [11]) in case the CUSF is at local exchange level or the value received in TC message

    The parameter "bearerCapability" shall only be included in the "InitialAssociationDP" operation in case the parameter is available.

  - tmr:
    This sub-parameter shall be omitted for this operation.

- USIServiceIndicator:
  It indicates the SL requesting the Monitoring of an UTSI information element. It is used as a Monitoring criteria at the CUSF level. It also provides the correlation with the RequestReportUTSI operation.

- USIInformation:
  This parameter conveys information provided by the User dedicated to the SL. It is transparent at the CUSF level.

NOTE: Two alternative cases are possible:

Case 1: The service ASE is located in the CUSF and the CUSF acts as a relay function between the service ASE and the SCF. The SCF provides additional information for the connection processing. In this case USIServiceIndicator and USIInformation shall not be used.

Case 2: The service ASE is located in the SCF and the CUSF acts as a relay function between the user and the SCF. The SCF receives and may send USI information. In this case only USIServiceIndicator, USIInformation, ServiceKey, MiscCallInfo and EventTypeBCUSM shall be used.

18.57.2 Invoking Entity (CUSF)

18.57.2.1 Normal Procedure

CUSF Precondition

(1) CUSF-FSM is in the state a: Idle

(2) The association has been established between the user and the network.

(3) An event fulfilling the criteria for the DP being executed has been detected.

CUSF Postcondition

(1) CUSF-FSM moves to the state b: Waiting For Instructions (TDP-R),

(2) CUSF-FSM remains in or moves to the state a: Idle (TDP-N)

NOTE: The information provided by the SCF depends on the service ASE located in the CUSF (case 1).

If the DP was armed as a TDP-R a control relationship is established to the SCF. The CUSF application timer TCUSF is set when the CUSF sends "InitialAssociationDP" for requesting instructions from the SCF. It is used to prevent excessive call suspension time.

18.57.2.2 Error Handling

If the error will occur within the CUSF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.57.3 Responding Entity (SCF)

18.57.3.1 Normal Procedure

SCF Precondition

(1) FSM for CUSF within the SCF is in the state N1: Idle

SCF Postcondition

(1) FSM for CUSF moves to the state N2.1: Preparing CUSF Instructions (TDP-R)

(2) FSM for CUSF remains in state a: Idle (TDP-N)

(3) Waiting for the request from the SLPI and CUSF instructions are being prepared.

18.57.3.2 Error Handling

If the error will occur within the SCF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.58 InitiateAssociation procedure

18.58.1 General description

This operation is used to allow the SCF to initiate a call unrelated association with the user.

18.58.1.1 Parameters

- calledPartyNumber:
  This parameter indicates the target line identity when the SCF initiates the association.
- USIServiceIndicator:
  It indicates the SL requesting the Monitoring of an UTSI information element. It is used as a Monitoring criteria at the SSF level. It also provides the correlation with the RequestReportUTSI operation.
- USIInformation:
  This parameter conveys information provided by the User dedicated to the SL. It is transparent at the SSF level.

Result parameters:

None.

18.58.2 Invoking entity (SCF)

18.58.2.1 Normal procedure

SCF Precondition

(1) FSM for CUSF within the SCF is in the state N1: Idle.

(2) SLPI requests to initiate a call unrelated association with the user.

SCF Postcondition

(1) FSM for CUSF within the SCF prepares to send a USI information to the user and goes to the state N2: Preparing CUSF Instructions.

18.58.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.58.3 Responding entity (CUSF)

18.58.3.1 Normal procedure

CUSF Precondition

(1) CUSF-FSM is in the state a: Idle.

CUSF Postcondition

(1) CUSF-FSM goes to the state b: Waiting For Instructions.

(2) A Return Result is sent.

The BCUSM is instantiated and suspended at the ActivationReceivedAndAuthorized DP. The CUSF is waiting for subsequent instructions from the SCF.

18.58.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.59 InitiateCallAttempt procedure

18.59.1 General description

This operation is used to request the SSF to create a new call to one call party using the address information provided by the SCF (e.g. wake-up call). An EDP-R must be armed on answer and all the call failure events, in order to have the SCF treat this call appropriately when either of these events is encountered. InitiateCallAttempt can also be used to create additional Call Segments within an existing Call Segement Association of ICA).

If the SL wishes to know the created CSA-ID, e.g. to be able later on to merge CSs from other CSAs with this one, a CreateCSA operation needs to preceed the ICA operation.,

18.59.1.1 Parameters

- destinationRoutingAddress:
  This parameter contains the called party number towards which the call is to be routed. The destinationRoutingAddress may contain one called party number only for this operation. The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22].

- alertingPattern:
  See ITU-T Recommendation Q.1290 [29]. It only applies if the network signalling supports this parameter or if SSF is the terminating local exchange for the subscriber.

- iSDNAccessRelatedInformation:
  Carries the same information as the protocol element ISUP Access Transport parameter in EN 300 356-1 [9]. See IN CS2 Signalling Interworking Requirements [49].

- serviceInteractionIndicators:
  This parameter contain indicators sent from the SCF to the SSF for control of the network based services at the originating exchange and the destination exchange.

- callingPartyNumber:
  This parameter identifies which number shall be regarded as the calling party for the created call. If this parameter is not sent by the SCF, the SSF may supply a network dependent default value.

- LegToBeCreated
  This parameter indicates the LegID to be assigned to the newly created party. When not provided, a default LegID of 1 is assumed ("new CSA" case). In the existing "CSA" case, this parameter shall be provided by the SCF

- NewCallSegment
  This parameter indicates the CS ID to be assigned to thenewly created Call Segment. When not provided, a default CSID of 1 is assumed ("new CSA" case). If sent within an existing control relationship and there is already CSs exiting in the CSA, this parameter shall be provided by the SCF

- iNServiceCompatibilityResponse:

  This parameter is used by SCF to inform the SSF about the actual services/service features which have been invoked in the SCF.

- serviceInteractionIndicatorsTwo
  Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services and network based services, respectively between different IN based services.

- carrier:
  This parameter indicates the transit network(s) requested for the call. The encoding of the parameter is defined in ITU-T Recommendation Q.763 [22].

- locationNumber:
  This parameter is used to convey the geographical area address for mobility services, see ITU-T Recommendation Q.762 [21].

- bearerCapability:
  This parameter indicates the type of the bearer capability connection or the transmission medium requirements to the user. See IN CS2 Signalling Interworking Requirements [50].
  It is a network option to select one of the two parameters to be used:

  - bearerCap:
    This parameter contains the value of the DSS1 Bearer Capability parameter (EN 300 403-1 [11]) in case the SSF is at local exchange level or the value of the ISUP User Service Information parameter (ITU-T Recommendation Q.763 [22]) in case the SSF is at transit exchange level.

  - tmr:
    The tmr is encoded as the Transmission Medium Requirement parameter of the ISUP according to ITU-T Recommendation Q.763 [22].

18.59.2 Invoking entity (SCF)

18.59.2.1 Normal procedure

SCF Preconditions:

(1) An SLPI has been invoked,

(2) An SLPI has determined that an "InitiateCallAttempt" operation should be sent by the SCF.

(3) If no control relationship exists an instance of the FSM for CSA is created by SCME-Control .The FSM for CSA is put in state "SSF Control Idle".

(4) If a control relationship exists, the FSM for CSA is in state "Preparing SSF Instructions.

SCF Postconditions:

(1) A control relationship is established between the SCF and SSF, if not existing.

(2) The FSM for CS is in state "Preparing CS Instructions".

(3) SLPI execution continues.

The FSM for CS moves to state "Preparing SSF Instructions" when the SL invokes this operation. In order to enable the establishment of a control relationship between the SCF and SSF and to allow the SCF to control the created call appropriately, the SLPI shall monitor for the BCSM event(s) which report the result of the created call setup. This includes DP Analyze_Information or DP Route_Select_Failure, ,O_Called_Party_Busy,O_NO-Answer, and O_Answer. Any other Non-Call_Processing_Instructions may be sent as well. The "InitiateCallAttempt" operation creates a BCSM instance in the SSF but the SSF suspends the call processing of this BCSM. The SLPI shall send a "Continue" operation to request the SSF to route the call to the specified destination. The FSM for CS shall proceed as specified at the procedure for the "Continue" operation.

The above described procedure shall be part of the establishment of the control relationship, i.e. operations up to and including the "Continue" operation shall be sent together in the same message to the SSF.
The SCF shall start a response Timer T_{scf} when the "InitiateCallAttempt" operation is sent. The response Timer shall supervise the confirmation of the dialogue from SSF, the value of T_{scf} shall be equal or less than the network no answer timer.

18.59.2.2 Error handling

On expiration of T_{scf} the SCF shall abort the dialogue, report the abort to the maintenance functions and inform the SLPI on the failure of dialogue establishment. The FSM for CS moves to the Idle state.

Generic error handling for the operation related errors are described in clause 16, the TC services which are used for reporting operation errors are described in clause 18.

18.59.3 Responding entity (SSF)

18.59.3.1 Normal procedure

SSF Precondition:

(1) If no control relationship exists an instance of the FSM for CSA is created by SSME-Control .The FSM for the CSA is put in state "Idle" .

(2) If a control relationship exists the FSM for the CSA is in state "Active".

SSF Postconditions:

(1) A new O-BCSM has been created, call processing is suspended at DP Origination_Attempt_Authorized.

(2) The new FSM for CS has moved from "Idle" state to state "Waiting for Instructions".

Upon reception of "InitiateCallAttempt", the SSF creates a new O-BCSM and suspends the call processing of this BCSM at DP Origination_Attempt_Authorized. All subsequent operations are treated according to their normal procedures.

The properties and capabilities, normally received from or associated to the calling party, required for the call setup shall have a network dependent default value. If a calling party number is supplied by the SCF, these properties may be dependent on the received calling party number

18.59.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16, the TC services which are used for reporting operation errors are described in clause 18.

18.60 ManageTriggerData procedure

18.60.1 General description

This operation is used to activate, deactivate or retrieve the status of a trigger DP linked to a subscriber profile known at a switch, e.g. related to an access line. This operation is used for SL controlled IN management purposes. The status or a success indication are sent to the SCF as Return Result of this operation.

18.60.1.1 Parameters

- actionIndicator
  This parameter indicates the action to be performed, i.e.
  - activate a TDP,
  - deactivate a TDP,
  - interrogate the current state of the TDP.

- triggerDataIdentifier
  Identifies the TDP and the corresponding subscriber profile that is to be managed

  - triggerID
    This parameter identifies the TDP-Type

  - profileIdentifier
    Provides several addressing schemes to identify the line/subscriber profile linked to the TDP:

    - access
      Identifies a subscriber access line

    - group
      Identifies a facility group.

- registratorIdentifier
  This parameter indicates the SCF which is to be verified by the SSF against the administrated information associated with the TDP.

- actionPerformed
  This parameter indicates the result of the operation (activated, deactivated, TDP status)

18.60.2 Invoking Entity (SCF)

18.60.2.1 Normal Procedure

SCF Precondition:

(1) SLPI detects that trigger data managing actions are to be performed.

SCF Postconditions:

(1) SCME is in the state "Idle"

If the SL at the SCF requests the activation, deactivation or status interrogation of trigger data at the SSF the SCF-FSM sends a ManageTriggerData operation with corresponding ActionIndicator.

If ManageTriggerData has been successfully processed the ReturnResult indicates the action performed or in case of interrogation the state of the TDP.

18.60.2.2 Error Handling

Generic error handling for the operation related error are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.60.3 Responding Entity (SSF)

18.60.3.1 Normal Procedure

SSF Precondition:

(1) None

SSF Postconditions:

(1) SSME-FSM is in the state "Idle"

It is checked whether corresponding TDP and subscriber profile addressed by operation ManageTriggerData exist. If so, the TDP is activated, deactivated or the status is retrieved. The result or an error indication are sent back as ReturnResult of ManageTriggerData to the initiating SCF.

The (de)activation of an already (in)active TDP is not an error case. It will only be indicated that this TDP was already (in)active so that the SCF may provide an appropriate indication to the requesting entity, (e.g. an Service Management Function (SMF)).

18.60.3.2 Error Handling

Generic error handling for the operation related error are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.61 MergeCallSegments procedure

18.61.1 General description

This operation is used to request the SSF to merge two associated CSs into a single CS. It (re)establishes the bearer connection between all involved legs. This operation is the inverse of the SplitLeg operation.

In merging the specified "source" CS, the conditions of the leg which the CS has: the armed EDPs, the ApplyChargingReport pending, the EventNotificationCharging pending , and the CallInformationReport pending, are also applied for the same leg after being merged.

After the execution of the CPH operation the FSM models for the involved CS's in the SSF shall transit to the WFI state while the associated BCSM instances for the involved CS's shall move from the PIC to the DP of the corresponding mid call DP in order to handle subsequent EDP arming and call processing operation.

18.61.1.1 Parameters

- sourceCallSegment:
  This parameter indicates the CS that shall be merged with another CS. After the merge, this CS instance will be deleted.

- targetCallSegment:
  This parameter indicates the CS that shall be merged with another CS. After the merge, this CS instance will be kept. When not specified, the source CS shall be merged with the initial CS.

18.61.2 Invoking entity (SCF)

18.61.2.1 Normal procedure

SCF Precondition:

(1) A control relationship exists between the SCF and the SSF.

(2) An SLPI has determined that two call segments shall be merged into a single call segment.

(3) The FSM for CS is in state C2 "Preparing CS Instructions".

SCF Postcondition:

(1) SLPI execution may continue.

(2) The FSM for CS remains in the same state C2 "Preparing CS Instructtions".

18.61.2.2 Error handling

Generic error handling for the operation related error are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.61.3 Responding entity (SSF)

18.61.3.1 Normal procedure

SSF Precondition:

(1) A control relationship exists between the SCF and the SSF.

(2) All involved BCSMs in the source CS are in state O/T_Active PIC, O/T_Suspended PIC, O_Alerting PIC, Send_Call PIC or Present_Call PIC or in one of the mentioned PIC associated DPs.

(3) The FSM for CS of the source CS is in the state "Waiting For Instructions" or "Monitoring".

(4) Any passive legs in the source CS and the target CS to be merged have the status "joined".

(5) When the involved leg is an "outgoing" leg (i.e. the passive leg in an O_BCSM or the controlling leg in a T_BCSM), the corresponding BCSM shall be at least at the Send_Call PIC in case of an O_BCSM or T_Active PIC in case of a T_BCSM.

SSF Postcondition:

(1) The SSF performs the necessary actions to merge the indicated Call Segments. All legs with status "joined" of the source CS are now connected to the Connection Point of the target CS.

(2) The FSM for CS of the source CS returns to idle state.

(3) The FSM for CS for the target Call Segment will move to the "Waiting for instructions" state. The associated BCSM instances of the involved Call Segment will move from the PIC to the DP of the corresponding O_/T_MidCall DP, when not already suspended at a DP. Note that no MidCall EDP will be reported for this case.

(4) A Return Result is sent immediately after the successful change of the leg configuration is executed, this allows the SCF to be updated with the established connection view and to cater for possible interference problems with signalling events.

18.61.3.2 Error handling

Generic error handling for the operation related error are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.62 ModifyEntry procedure

18.62.1 General description

The ITU-T Recommendation X.500 [36] "ModifyEntry" operation is used to request the SDF to make one or several modifications to a data object. The modifications concern the attributes and their values of which the object is composed. The type of modifications to perform is given in the operation argument provided by the SCF. The modifications do not concern the values of the attributes used to identify the object (i.e., the object name). For a full description of the ModifyEntry operation, see ITU-T Recommendation X.511 [39] subclause 11.3.

18.62.1.1 Parameters

See ITU-T Recommendation X.511 [39] subclauses 11.3.2 and 11.3.3.3.

18.62.2 Invoking entity (SCF)

18.62.2.1 Normal procedure

SCF Preconditions:

(1) SCSM: "SDF Bound" or "Wait for Subsequent Requests"

SCF Postconditions:

(1) SCSM: "SDF Bound"

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the SL to modify information of the SDF exists, an internal event ((e2) Request_to_SDF) occurs. Until the application process indicates, with a delimitor (or a timer expiry), that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response ((E5) Response_from_SDF_with_Bind or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the ModifyEntry operation was discarded. In the State "SDF Bound", the response of the ModifyEntry operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the ModifyEntry operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the SL to modify information of the SDF exists an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response ((E7) Response_from_SDF) to the ModifyEntry operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the ModifyEntry operation or an error.

18.62.2.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 11.3.4 and 11.3.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.62.3 Responding entity (SDF)

18.62.3.1 Normal procedure

SDF Preconditions:

(1) SDSM: "SCF Bound"or "Bind Pending"

SDF Postconditions:

(1) SDSM "SCF Bound"

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a "ModifyEntry" operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "ModifyEntry" operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before updating the different attributes specified in the operation parameters, the SDF shall take the following actions:

- verify that the object accessed by the request exists;
- verify that the user on behalf of whom the request is performed has sufficient access rights to modify the object for each elementary modifications contained in the operation;
- verify that each attribute or value on which an operation should be performed exists in the object;
- verify that the proposed modifications are compatible with the object class of the object or with the syntax of the attribute.

After the specified actions indicated above are successfully executed for all the modifications requested in the operation, all the modifications for a same attribute take place in the order given in the "changes" parameter. A result is returned to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

18.62.3.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 11.3.4 and 11.3.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.63 MoveCallSegments procedure

18.63.1 General description

This operation moves a Call Segment from the source Call Segment Association to the target Call Segment Association. The SCF specifies a new identifier for the moved CS, as well as for each leg associated with the moved CS.

This operation ends the association between the moved Call Segment and any Call Segments remaining in the source Call Segment Association.

A CS may not be moved into a CSA containing another CS with a relationship to a different controlling leg party. A moved CS inherits the TC transaction opened for the target CSA.

A Call Segment Association may contain any number of Call Segments. The number of Call Segments that may be moved into or out of a CSA is not limited by physical restrictions on the number of parties a particular switch implementation can support in a multi-party call.

If the MoveCallSegments operation results in a null source Call Segment Association (i.e., no remaining Call Segments), the source CSA is deleted.

After the execution of the CPH operation the FSM models for the involved CS's in the SSF shall transit to the WFI state while the associated BCSM instances for the involved CS's shall move from the PIC to the DP of the corresponding mid call DP in order to handle subsequent EDP arming and call processing operation.

18.63.1.1 Parameters

- targetCallSegmentAssociation:
  This parameter indicates the target Call Segment Association into which the moved Call Segment is to be placed.
- callSegments
  - sourceCallSegment:
    This parameter indicates the source Call Segment to be moved.
  - newCallSegment
    This parameter specifies the new CSID of the moved CS.
- legs
  - sourceLeg
    This parameter specifies a source leg ID from the moved CS.
  - newLeg
    This parameter specifies the new LegID of the source leg

18.63.2 Invoking entity (SCF)

18.63.2.1 Normal procedure

SCF Preconditions:

(1) The Call is in an appropriate Call Connection View state.

(2) The FSM for CS is in state C2 "Preparing CS Instructions".

(3) A control relationship has been established and the SLPI is processing the incoming request.

SCF Postconditions:

(1) SLPI execution may continue.

(2) The FSM for CS remains in the state C2 "Preparing CS Instructions".

18.63.2.2 Error handling

Generic error handling for the operation related errors is described in clauses 16 and the TC services that are used for reporting operating errors are described in clause 18.

18.63.3 Responding entity (SSF)

18.63.3.1 Normal procedure

SSF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An appropriate Call Connection View state exists.

(3) In the source CSA if the CS to be moved has a joined controlling leg, then the target CSA must not contain a joined controlling leg.

SSF Postconditions:

(1) The SSF performs the appropriate call processing actions.

(2) The appropriate Call Connection View state is determined.

(3) In the source CSA if the CS to be moved has a joined controlling leg, then the controlling leg status in the remaining CS must transition to surrogate. In the target CSA, the status of the controlling leg in others CS does not change.

(4) The FSM for CS for the "new" Call Segment in the target CSA will move to the "Waiting for Instructions" state. The associated BCSM instances within the "new" Call Segment will move from the PIC to the DP of the corresponding O_/T_MidCall DP, when not already suspended at a DP. Note that no MidCall EDP will be reported for this case.

18.63.3.2 Error handling

Generic error handling for the operation related errors is described in clauses 16 and the TC services that are used for reporting operating errors are described in clause 18.

18.64 MoveLeg procedure

18.64.1 General description

This operation requests the SSF to move the Leg from one source CS to another target CS with which it is associated.

The effect of MoveLeg for the controlling Leg is to interrupt the current communication of the controlling Leg, without clearing the passive Leg on that communication, and to establish communication for the controlling Leg with the other passive leg. Only the controlling Leg is moved.

The effect of MoveLeg for the passive Leg is to move the passive Leg and associated BCSM instance from one CS to another CS with which it is associated.

In moving the specified leg, the conditions of the leg: the armed EDPs, the ApplyChargingReport pending, the EventNotificationCharging pending , and the CallInformationReport pending, are also applied for the same leg after moved.

After the execution of the CPH operation the FSM models for the involved CS's shall transit to the WFI state while the associated BCSM instances for the involved CS's shall move from the PIC to the DP of the corresponding mid call DP in order to handle subsequent EDP arming and call processing operation.

18.64.1.1 Parameters

- legIDToMove:
  This parameter indicates the Leg that shall be moved.

- targetCallSegment:
  This parameter indicates the CS that the leg shall be moved to. After the move, this CS instance will be kept. When not specified, the leg shall be moved to the initial CS.

18.64.2 Invoking entity (SCF)

18.64.2.1 Normal procedure

SCF Preconditions:

(1) The Call is in an appropriate Call Connection View state.

(2) The FSM for CS is in state C2 "Preparing CS Instructions".

(3) A control relationship has been established and the SLPI is processing the incoming request.

SCF Postconditions:

(1) SLPI execution may continue.

(2) The FSM for CS remains in the state C2 "Preparing CS Instructions".

18.64.2.2 Error handling

Generic error handling for the operation related errors is described in clauses 16 and the TC services that are used for reporting operating errors are described in clause 18.

18.64.3 Responding entity (SSF)

18.64.3.1 Normal procedure

SSF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An appropriate Call Connection View state exists.

(3) When the involved leg is an "outgoing" leg (i.e. the passive leg in an O_BCSM or the controlling leg in a T_BCSM), the corresponding BCSM shall be at least at the Send_Call PIC in case of an O_BCSM or T_Active in case of a T_BCSM.

SSF Postconditions:

(1) The SSF performs the appropriate call processing actions.

(2) The appropriate Call Connection View state is determined.

(3) The FSM for the source and target Call Segments will move to the "Waiting for instructions" state. The associated BCSM instances within the two involved Call Segments will move from the PIC to the DP of the corresponding O_/T_MidCall DP, when not already suspended at a DP. Note that no MidCall EDP will be reported for this case. If the last leg in the source CS is moved, the FSM for the source Call Segment will move to the "Idle" state.

(4) A Return Result is sent immediately after the successful change of the leg configuration is executed, this allows the SCF to be updated with the established connection view and to cater for possible interference problems with signalling events.

18.64.3.2 Error handling

Error handling for the operation related errors is described in clauses 16 and the TC services that are used for reporting operating errors are described in clause 18.

18.65 NetworkCapability procedure

18.65.1 General Description

This operation provides the different types of services that are supported for the user involved in the call and in the context of that call, if not already specified in the agreement. It is used by the two interworking networks to agree on a level of service that can be expected for the call.

18.65.1.1 Parameters

- bearerCapabilities:
  This parameter (either in the invoke or in the result) contains the list of the bearer services that are available to the user in the context of the call. The list is specific of the user and of the call context.

- highLayerCompatibilities:
  This parameter (either in the invoke or in the result) contains the list of the teleservices that are available to the user in the context of the call. The list is specific of the user and of the call context.

- supplementaryServices:
  This parameter (either in the invoke or in the result) contains the list of the supplementary services that are available to the user in the context of the call. The list is specific of the user and of the call context.

- securityParameter:
  This is an optional parameter that conveys security related information.

18.65.2 Invoking entity (supporting SCF)

18.65.2.1 Normal procedure

SCF Precondition:

(1) A "handlingInformationRequest" operation has been received and the preparation of a "handlingInformationResult" parameter is pending.

(2) The need for knowing the level of service to be available to the user has been identified by the SLPI.

(3) The SCF FSM is in the state "Assisting Mode".

SCF Postcondition:

(1) The SCF moves to the state "Waiting for Information".

Before sending the "networkCapability" operation, the supporting SCF has received a "handlingInformationRequest" operation containing information about the call, the controlling network and the user. If the information received is not enough, the supporting SCF can build the "networkCapability" operation, by sending the types of services that it expects can be available in the controlling network for the call conditions specified in the earlier operation, and for the given user. The "networkCapability" operation can take into account the agreements between the network operators, the user's service profile and the call context.

Once the operation has been sent, the SCF FSM moves to the state "Waiting for Information". The SCF waits for the answer from the other SCF.

18.65.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.65.3 Responding entity (controlling SCF)

18.65.3.1 Normal procedure

SCF Precondition:

(1) A "handlingInformationRequest" operation has been sent.

(2) The SCF FSM is in the state "Assisting Mode".

SCF Postcondition:

(1) The SCSM moves to the state "Preparing Additional Information".

(2) A Return Result is sent in answer

On receipt of the "networkCapability" operation, the SCF FSM moves to the state "Preparing Information". To prepare its answer the SCF looks at the list of services provided in the argument of the operation and removes the ones that it is not able to provide to the user in the conditions of the call.

18.65.3.2 Error Handling

If none of the services specified in the argument of the "networkCapability" operation can be provided by the controlling SCF, it returns to the supporting SCF a "taskRefused" error.

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.66 NotificationProvided procedure

18.66.1 General description

This operation is used to report that a call condition previously specified by the supporting SCF or pre-arranged between two network operators was met.

18.66.1.1 Parameters

- notifications:
  This parameter contains an indication that a call condition previously expressed by the supporting SCF or pre-arranged between the two operators has been met. It links together the call condition met and some information related to call (if any).

- notificationInformation:
  This parameter contains any other kind of information that might be needed to be notified by a specific kind of SL. Information that can be conveyed has been agreed between network operators when defining the SL object.

- securityParameter:
  This is an optional parameter that conveys security related information.

18.66.2 Invoking entity (controlling SCF)

18.66.2.1 Normal procedure

SCF Precondition:

(1) The SCF has received a "Request Notification" operation if call conditions at which this operation is sent has not been pre-arranged between two network operators.

(2) A call condition specified earlier by the supporting SCF or pre-arranged between two network operators has been met.

(3) The SCF-SCF relationship has been maintained since the "SCF Bind handlingInformationRequest" operation has been sent.

(4) The SCF FSM is in the state "Assisted Mode".

SCF Postcondition:

(1) SCF FSM remains in the same state.

If any call resource has been engaged before the "NotificationProvided" operation is sent (e.g. as a result of the "Requested Notification" handlingInformationResult" operation), it remains as it is.

If several call conditions as specified by the supporting SCF or as pred-arranged between two network operators have been met, they are reported in sequence to the supporting SCF, which takes the appropriate actions.

18.66.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.66.3 Responding entity (supporting SCF)

18.66.3.1 Normal procedure

SCF Precondition:

(1) A dialogue between the two SCFs has been previously established

(2) The SCF FSM is in the state "Assisting Mode".

SCF Postcondition:

(1) SCF FSM remains is in the same state.

On receipt of the "notificationProvided" operation the SLPI determines whether the call configuration should be modified as a consequence of the received notification information. If the call configuration in the controlling SCF needs to be modified, the supporting SCF prepares instructions to assist the controlling SCF and sends them with a "handlingInformationResult" operation. Otherwise the supporting SCF does not undertake any actions towards the controlling SCF, but the notification can be used in the SLPI (e.g. for charging purposes).

18.66.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.67 PlayAnnouncement procedure

18.67.1 General description

This operation is used for inband interaction with an analogue user or for interaction with an ISDN user. When used to apply user treatment to the indicated leg, the user treatment can be audible (e.g. inband tone) or visual (e.g. text displayed).

18.67.1.1 Parameters

- informationToSend:
  This parameter indicates an announcement, a tone or display information to be sent to the end user by the SRF.
  - inbandInfo:
    This parameter specifies the inband information to be sent.
    - messageID:
      This parameter indicates the message(s) to be sent, this can be one of the following:
      - elementaryMessageID:
        This parameter indicates a single announcement.
      - text:
        This parameter indicates a text to be sent. The text shall be transformed to inband information (speech) by the SRF. This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.
      - elementaryMessageIDs:
        This parameter specifies a sequence of announcements.
      - variableMessage:
        This specifies an announcement with one or more variable parts.
    - numberOfRepetitions:
      This parameter indicates the maximum number of times the message shall be sent to the end-user.

- duration:
  This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

- interval:
  This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

- tone:
  This parameter specifies a tone to be sent to the end-user.

  - toneID:
    This parameter indicates the tone to be sent.

  - duration:
    This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

- displayInformation:
  This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

NOTE: As the current signalling systems (DSS1/ISUP) do not provide an indication whether or not information can be displayed by the user's terminal, in case of user interaction with an ISDN user two consecutive "PlayAnnouncement" operations are sent. The first contains the display information, the second contains the inband information to be sent to the user. Since the execution of the display information by the SRF should take a limited amount of time, the inband information will be immediately sent by the SRF to the user, in sequence with the display information.

- disconnectFromIPForbidden:
  This parameter indicates whether or not the SRF should be disconnected from the user when all information has been sent.

- requestAnnouncementComplete:
  This parameter indicates whether or not a "SpecializedResourceReport" shall be sent to the SCF when all information has been sent.

- connectedParty:
  This parameter shall be present when applied in a multi call segment CSA.
  When not present in a single call segment CSA it implies that user interaction shall apply to the call segment, i.e. to all parties connected to the call segment.

  - legID:
    This parameter indicates to which party in the call the interaction shall apply while maintaining the speech connection between that leg and any other legs connected to the same CS.

  - callSegmentIdentifier:
    This parameter indicates to which call segment the user interaction shall apply, i.e. to all parties connected to the call segment.

18.67.2 Invoking entity (SCF)

18.67.2.1 Normal procedure

SCF Preconditions:

(1) The SLPI detects that information should be sent to the user.

(2) A connection between the user and a SRF has been established.

(3) The SCSM-FSM is in the state "User interaction", substate "Waiting for response from the SRF".

SCF Postconditions:

(1) If "RequestAnnouncementComplete" was set TRUE, the SCSM will stay in substate "Waiting for Response from the SRF" and wait for the "SpecializedResourceReport".

(2) If "RequestAnnouncementComplete" was set FALSE and more information needs to be sent ("DisconnectFromIPForbidden" was set to TRUE), the SCSM will stay in substate "Waiting for Response from the SRF".

(3) If "RequestAnnouncementComplete" was set FALSE and no more information needs to be sent ("DisconnectFromIPForbidden" was set to FALSE), the SCSM will move to the state "Preparing SSF Instructions".

18.67.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.67.3 Responding entity (SRF)

18.67.3.1 Normal procedure

SRF Precondition:

(1) The SRSM-FSM is in the state "Connected", or
in the state "User Interaction" if the SRF received previously an operation from the SCF..

SRF Postconditions:

(1) The SRF sends the information to the user as indicated by "informationToSend".

(2) The SRSM-FSM moves to the state "User Interaction", or
remains in the same state.

(3) If all information has been sent and "RequestAnnouncementComplete" was set TRUE, the SRSM sends a "SpecializedResourceReport" operation to the SCF.

(4) If all information has been sent and "disconnectFromIPForbidden" was set FALSE, the SRSM disconnects the SRF from the user.

The announcement send to the end-user is ended in the following conditions:

- if neither "duration" or "numberOfRepetitions" is specified, then the network specific announcement ending conditions shall apply; or
- if "numberOfRepetitions" is specified, when all repetitions have been sent, or
- if duration is specified, when the duration has expired. The announcement is repeated until this condition is met, or
- if "duration" and "numberOfRepetitions" is specified, when one of both conditions is satisfied (whatever comes first).

18.67.3.2 Error handling

If a Cancel operation is received before or during the processing of the operation then the operation is immediately cancelled and the error "Canceled" is reported to the invoking entity.

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.68 PromptAndCollectUserInformation procedure

18.68.1 General description

This operation is used to interact with a call party in order to collect information.

18.68.1.1 Parameters

- collectedInfo
  - collectedDigits
    - minimumNbOfDigits:
      If this parameter is missing, the default value is defined to be 1. The "minimumNbOfDigits" specifies the minimum number of valid digits to be collected.
    - maximumNbOfDigits:
      This parameter should always be present and specifies the maximum number of valid digits to be collected. The following applies: "maximumNbOfDigits" >= "minimumNbOfDigits".
    - endOfReplyDigit:
      This parameter indicates the digit used to signal the end of input.

      In case the "maximumNbOfDigits" = "minimumNbOfDigits", the "endOfReplyDigit" (could be present but) has no further meaning. This parameter can be one or two digits.

      In case the "maximumNbOfDigits" > "minimumNbOfDigits" the following applies:

      If "endOfReplyDigit" is not present, the end of input is indicated:

      - when the inter-digit timer expires, or
      - when the number of valid digits received equals the "maximumNbOfDigits".

      If "endOfReplyDigit" is present, the end of input is indicated:

      - when the inter-digit timer expires, or
      - when the end of reply digit is received, or
      - when the number of valid digits received equals the "maximumNbOfDigits".

      When the end of input is attained, the collected digits are send from SRF to the SCF, including the "endOfReplyDigit" if received by the SRF.

      In the case the number of valid digits received is less than the "minimumNbOfDigits" when the inter-digit timer expires or when the end of reply digit is received, the input is specified as being erroneous.
    - cancelDigit:
      If this parameter is present, the cancel digit can be entered by the user to request a possible retry. All digits already received by the SRF are discarded and the same "PromptAndCollectInformation" procedure is performed again, thus e.g. the same announcement to request user information is given to the user and information is collected. This parameter can be one or two digits .

      If this parameter is not present, the user is not able to request a possible retry.
    - startDigit:
      If this parameter is present, the start digit indicates the start of the valid digits to be collected. The digits that are received by the SRF before this start digit is received, are discarded and are not considered to be valid. This parameter can be one or two digits.

      If this parameter is not present, all received digits are considered to be valid.

      When the end of input is attained, the collected digits are sent from SRF to the SCF, including the "startDigit" if received by the SRF.

- firstDigitTimeout:
  If this parameter is present, the first digit should be received by the SRF before the first-digit timer expiration. In case the first digit is not received before first-digit timer expiration, the input is regarded to be erroneous. After receipt of the first valid or non-valid input digit, the corresponding first-digit timer is stopped.

  If this parameter is not present, then the SRF uses a default value (network operator specific) for the first-digit timer in which the first valid or non-valid input digit is received.

  If "startDigit" is present, the first-digit timer is stopped after the start digit is received.

- interDigitTimeOut:
  If this parameter is present any subsequent valid or non-valid digit, should be received by the SRF before the inter-digit timer expires. As result the inter-digit timer is reset and restarted.

  In case a subsequent valid or non-valid digit is not received before the inter-digit timer expires and the number of received valid digits is less than the "minimumNbOfDigits", the input is regarded to be unsuccessful.

  In case a subsequent valid or non-valid digit is not received before the inter-digit timer expires and the number of received valid digits is greater than the "minimumNbOfDigits", and less than or equal to the "maximumNbOfDigits", the input is regarded to be successful.

  If the "interDigitTimeOut" is not present, then the SRF uses a default value for the inter-digit time.

- errortreatment:
  This optional parameter defines what specific action should be taken by the SRF in the event of error conditions occurring. The default value is reportErrorToSCF.

- interruptableAnnInd:
  This parameter is optional, where the default value is specified being TRUE.

  If this parameter is TRUE, the announcement is interrupted after the first valid or non-valid digit is received by the SRF. If the announcement is interrupted, a possible start-digit timer will not apply anymore. However, if the announcement has not been interrupted, a possible start-digit timer is started after the announcement has been finished.

  If this parameter is present and explicitly set to FALSE, the announcement will not be interrupted after the first digit is received by the SRF. The received digits during the announcement are discarded and considered to be non-valid. All other specified parameters ("minimumNbOfDigits", "maximumNbOfDigits", "endOfReplyDigit", etc.) do not apply before the announcement has been finished. The possible start-digit timer is started after the announcement has been finished.

- voiceInformation:
  This parameter is optional, where the default value is specified being FALSE. If the "voiceInformation" parameter is FALSE, all valid or non-valid digits are entered by DTMF.

  If this parameter is present and explicitly set to TRUE, calling user is required to provide all valid or non-valid information by speech. The SRF will perform voice recognition and translation of the provided information into digits. A possible end of reply digit will also have to be provided by speech.

- voiceBack:
  This parameter is optional, where the default value is specified being FALSE. If the "voiceBack" parameter is FALSE, no voice back information is given by the SRF.
  If this parameter is present and explicitly set to TRUE, the valid input digits received by the SRF will be announced back to the calling user immediately after the end of input is received. The non-valid input digits will not be announced back to the calling user.

  A possible end of reply digit is not voiced back.

- disconnectFromIPForbidden:
  This parameter indicates whether the SRF should initiate disconnection to the SSF/CCF after the interaction has been completed. If the parameter is not present or set to TRUE, the SRF shall not initiate disconnection.

- informationToSend:
  This parameter indicates an announcement, a tone or display information to be sent to the end user by the SRF.

  - inbandInfo:
    This parameter specifies the inband information to be sent.

    - messageID:
      This parameter indicates the message(s) to be sent, this can be one of the following:

      - elementaryMessageID:
        This parameter indicates a single announcement.

      - text:
        This parameter indicates a text to be sent. The text shall be transformed to inband information (speech) by the SRF. This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

      - elementaryMessageIDs:
        This parameter specifies a sequence of announcements.

      - variableMessage:
        This parameter specifies an announcement with one or more variable parts.

    - numberOfRepetitions:
      This parameter indicates the maximum number of times the message shall be sent to the end-user.

    - duration:
      This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

    - interval:
      This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

  - tone:
    This parameter specifies a tone to be sent to the end-user.

    - toneID:
      This parameter indicates the tone to be sent.

    - duration:

      This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

  - displayInformation:
    This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

NOTE: As the current signalling systems (DSS1/ISUP) do not provide an indication whether or not information can be displayed by the user's terminal, in case of user interaction with an ISDN user, the "displayInformation" parameter is not used in the "PromptAndCollectUserInformation" operation. Instead a "PlayAnnouncement" operation containing the "displayInformation" parameter followed by a "PromptAndCollectUserInformation" operation containing inband information are sent to the user. Since the execution of the displayed information by the SRF should take a limited amount of time, the inband information will be immediately sent after by the SRF to the user, in sequence with the displayed information.

- callSegmentIdentifier:
  This parameter indicates to which call segment the user interaction shall apply
  This parameter shall be present when applied in a multi call segment CSA.
  When not present in a single call segment CSA it implies that user interaction shall apply to the call segment, In this case only one call party is allowed to be connected to the call segment.

Result Parameters:

- digitsResponse:
  This parameter contains the information collected from the end-user.

18.68.2 Invoking entity (SCF)

18.68.2.1 Normal procedure

SCF Preconditions:

(1) The SLPI detects that information should be collected from the end-user.

(2) A connection between the end-user and a SRF has been established.

(3) The SCSM FSM is in state "User Interaction", substate "Waiting for Response from the SRF".

SCF Postconditions:

(1) The collected information is received from the SRF as response to the "PromptAndCollectUserInformation" operation.

(2) If the "disconnectFromIPForbidden" was set to FALSE, the SCSM FSM will move to the state "Preparing SSF Instructions".

(3) Otherwise the SCSM FSM remains in the same state.

The SLPI may continue execution before the response is received from the "PromptAndCollectUserInformation" operation, more then one operation may be sent to the SRF before the response is received. The "disconnectFromIPForbidden" parameter may only be set to FALSE if the "PromptAndCollectUserInformation" operation is the last operation sent to the SRF.

18.68.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16, the TC services which are used for reporting operation errors are described in clause 18.

18.68.3 Responding entity (SRF)

18.68.3.1 Normal procedure

SRF Precondition:

(1) The SRSM-FSM is in the state "Connected", or in state "User Interaction" if the SRF received previously an operation from the SCF.

SRF Postconditions:

(1) The SRF has sent the information to the end-user as indicated by "informationToSend".

(2) The collected information from the end-user is sent to the SCF as RETURN RESULT of the "PromptAndCollectUserInformation".

(3) If the "disconnectFromIPForbidden" was set to FALSE, the SRF initiates a bearer channel disconnect to the SSF and the SRSM FSM moves to the state "Idle".

(4) Otherwise the SRSM FSM moves to the state "User Interaction", or
remains in the same state..

The announcement send to the end-user is ended in the following conditions:

- if neither "duration" or "numberOfRepetitions" is specified, then the network specific announcement ending conditions shall apply; or
- if "numberOfRepetitions" is specified, when all repetitions have been sent, or
- if duration is specified, when the duration has expired. The announcement is repeated until this condition is met, or
- if "duration" and "numberOfRepetitions" is specified, when one of both conditions is satisfied (whatever comes first).

The above conditions are overruled if the parameter "interruptableAnnInd" is not set to FALSE and the end-user has responded with a digit during the sending of the announcement. In this case the announcement is ended immediately. The above procedures apply only to inband information and tones send to the end-user, for "displayInformation" the end conditions are met upon sending, i.e. no interruption can occur.

The parameter "errorTreatment" specifies how the SRF shall treat the error. The default value "reportErrorToSCF" means that the error shall be reported to SCF by means of Return Error with "ImproperCallerResponse". The value "help" indicates that no error shall be reported to SCF but assistance shall be given to the end-user in form of a network dependent default announcement (which may dependent on the context, i.e. the send message). The value "repeatPrompt" indicates that no error shall be reported to the SCF but the prompt shall be repeated to the end-user. The last two procedures shall only be done once per "PromptAndCollectUserInformation" operation.

Note on processing "endOfInput"

The receipt of any "endOfInput" condition (e.g endOfReplyDigit, cancelDigit, firstDigitTimeout, interDigitTimeout) terminates immediately the ongoing input. In other words when e.g an endOfReplyDigit is received, the receipt of a subsequent cancelDigit will not be processed anymore.

18.68.3.2 Error handling

If a Cancel operation is received before or during the processing of the operation then the operation is immediately cancelled and the error "Canceled" is reported to the invoking entity.

Generic error handling for the operation related errors are described in clause 16, the TC services which are used for reporting operation errors are described in clause 18.

If any of the parameter restrictions are violated (e.g. minimumNbOfDigits > maximumNbOfDigits) then an operation error has occured.

18.69 PromptAndReceiveMessage procedure

18.69.1 General description

This operation is used to interact with a call party in order to record information.

18.69.1.1 Parameters

- disconnectFromIPForbidden:
  This parameter indicates whether the SRF should initiate disconnection to the SSF/CCF after the interaction has been completed. If the parameter is not present or set to TRUE, the SRF shall not initiate disconnection.
- informationToSend:
  This parameter indicates an announcement, a tone or display information to be sent to the end user by the SRF.
  - inbandInfo:
    This parameter specifies the inband information to be sent.
    - messageID:
      This parameter indicates the message(s) to be sent, this can be one of the following:

- elementaryMessageID:
  This parameter indicates a single announcement.

- text:
  This parameter indicates a text to be sent. The text shall be transformed to inband information (speech) by the SRF. This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

- elementaryMessageIDs:
  This parameter specifies a sequence of announcements.

- variableMessage:
  This parameter specifies an announcement with one or more variable parts.

- numberOfRepetitions:
  This parameter indicates the maximum number of times the message shall be sent to the end-user.

- duration:
  This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

- interval:
  This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

- tone:
  This parameter specifies a tone to be sent to the end-user.

- toneID:
  This parameter indicates the tone to be sent.

- duration:

  This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

- displayInformation:
  This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

NOTE: As the current signalling systems (DSS1/ISUP) do not provide an indication whether or not information can be displayed by the user's terminal, in case of user interaction with an ISDN user, the "displayInformation" parameter is not used in the "PromptAndCollectUserInformation" operation. Instead a "PlayAnnouncement" operation containing the "displayInformation" parameter followed by a "PromptAndreceiveMessage" operation containing inband information are sent to the user. Since the execution of the displayed information by the SRF should take a limited amount of time, the inband information will be immediately sent after by the SRF to the user, in sequence with the displayed information.

- subscriberID
  This parameter identifies to the SRF, the subscriber for whom information shall be recorded.

- mailBoxID
  This parameter identifies the maibox for which information shall be recorded, in case the subscriber owns multiple mailboxes.

- informationToRecord

- messageID
  This parameter indicates the ID that shall be assigned to the recorded message. This option is used when the recording is not intended for a mailbox belonging to a specific subscriber, but a temporary recording, eg. within the context of 1 call.

- messageDeletionTimeOut
  This parameter indicates the maximum time duration a message recording shall be stored in the SRF.

- timeToRecord
  This parameter indicates the maximum allowed time for the recording

- controlDigits

  - endOfRecordingDigit
    If this parameter is present, the end of recording digit can be entered by the user in order to mark the end of the recording. This parameter can be one or two digits .

  - cancelDigit:
    If this parameter is present, the cancel digit can be entered by the user in order to cancel the ongoing recording. Any information received sofar shall be erased. This parameter can be one or two digits .

  - replayDigit:
    If this parameter is present, the replay digit can be entered by the user to have the recorded message replayed. This parameter can be one or two digits .

  - restartRecordingDigit:
    If this parameter is present, the restart digit can be entered by the user to request a possible retry. All information already received by the SRF is erased and the same "PromptAndReceiveMessage" procedure is performed again, thus e.g. the same announcement to request user information is given to the user and information is collected. This parameter can be one or two digits .

  - restartAllowed:
    This parameter indicates whether or not a possible restart recording is allowed.

  - replayAllowed:
    This parameter indicates whether or not a possible replay of the recording is allowed.

- media:
  This parameter indicates the type of media for the recordeding.

- callSegmentIdentifier:
  This parameter indicates to which call segment the user interaction shall apply.
  This parameter shall be present when applied in a multi call segment CSA.
  When not present in a single call segment CSA it implies that user interaction shall apply to the call segment, In this case only one call party is allowed to be connected to the call segment.

  i.e. to all parties connected to the call segment. When not provided, a default CSID of 1 is assumed.

- receivedStatus:
  This parameter indicates the result of the recordeing. Three values are distinguished:
  MessageComplete - the recording was successfully completed. End of recording may be indicated, eg. by an end of recording digit, voice activity monitoring or by onhook.
  MessageInterrupted - the user has abandoned the recording, eg.by going onhook or by pushing the cancel digit.
  MessageTimeOut - the maximum recording time has been exceeded.

- recordedMessageID
  This parameter reports to the SCF the ID assigned to the recorded message. It is only used, if the MessageID was not assigned by the SCF. This option can be used when the recording is intended for a mailbox belonging to a specific subscriber.

- recordedMessageUnits
  This parameter indicates the amount of resources occupied by the recorded message. This can be expressed in units of time, memory usage, etc.

18.69.2 Invoking entity (SCF)

18.69.2.1 Normal procedure

SCF Preconditions:

(1) The SLPI detects that information should be recorded from the end-user.

(2) A connection between the end-user and a SRF has been established.

(3) The SCSM FSM is in state "User Interaction", substate "Waiting for Response from the SRF".

SCF Postconditions:

(1) The ID of the recorded message is received from the SRF as response to the "PromptAndReciveMessage" operation.

(2) If the "disconnectFromIPForbidden" was set to FALSE, the SCSM FSM will move to the state "Preparing SSF Instructions".

(3) Otherwise the SCSM FSM remains in the same state.

The SLPI may continue execution before the response is received from the "PromptAndReciveMessage" operation, more then one operation may be sent to the SRF before the response is received. The "disconnectFromIPForbidden" parameter may only be set to FALSE if the "PromptAndReceiveMessage" operation is the last operation sent to the SRF.

18.69.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16, the TC services which are used for reporting operation errors are described in clause 18.

18.69.3 Responding entity (SRF)

18.69.3.1 Normal procedure

SRF Precondition:

(1) The SRSM-FSM is in the state "Connected", or in state "User Interaction" if the SRF received previously an operation from the SCF.

SRF Postconditions:

(1) The SRF has sent the information to the end-user as indicated by "informationToSend".

(2) After having recorded the message from the end-user, the idenitifier assigned to the recorded message is sent to the SCF as RETURN RESULT of the "PromptAndReciveMessage".

(3) If the "disconnectFromIPForbidden" was set to FALSE, the SRF initiates a bearer channel disconnect to the SSF and the SRSM FSM moves to the state "Idle".

(4) Otherwise the SRSM FSM moves to the state "User Interaction", or
remains in the same state..

The announcement send to the end-user is ended in the following conditions:

- if neither "duration" or "numberOfRepetitions" is specified, then the network specific announcement ending conditions shall apply; or
- if "numberOfRepetitions" is specified, when all repetitions have been sent, or
- if duration is specified, when the duration has expired. The announcement is repeated until this condition is met, or
- if "duration" and "numberOfRepetitions" is specified, when one of both conditions is satisfied (whatever comes first).

The above procedures apply only to inband information and tones send to the end-user, for "displayInformation" the end conditions are met upon sending, i.e. no interruption can occur.

Note on processing "endOfInput"

The receipt of any "endOfInput" condition (e.g endOfRecordingDigit timeToRecord terminates immediately the ongoing input. In other words when e.g an endOfReplyDigit is received, the receipt of a subsequent cancelDigit will not be processed anymore.

18.69.3.2 Error handling

If a Cancel operation is received before or during the processing of the operation then the operation is immediately cancelled and the error "Canceled" is reported to the invoking entity.

Generic error handling for the operation related errors are described in clause 16, the TC services which are used for reporting operation errors are described in clause 18.

18.70 ProvideUserInformation procedure

18.70.1 General description

This operation is used by a supporting SCF to request user information through the help of the SCF controlling the call. The received user information will be used in the supporting SCF to determine the way the call should be handled. The operation provides the means for the controlling SCF to build the operations to prompt the user and collect information from him/her.

The operation result sends back user information to an SCF that has requested it in order to assist the controlling SCF. It can also sent back indication that a user interaction has failed and the user information could not be collected from the user.

18.70.1.1 Parameters

- infoToSend:
  This parameter is used to convey the contents of the information passed to the user to request inputs from him/her. This information could be provided to the user with an IP, a display or a simple tone, but the mode of user interaction should be in line with the available modes that might have been provided in the "handlingInformationRequest" operation.

- errorInfo:
  This parameter contains the information to be passed to the user if the latter has failed to provide correct inputs to a previous interaction. To verify its correctness, the user input is only challenged against the information provided in the "constraints" parameter.

- constraints:
  This parameter defines what information should be expected from the user. It indicates the type and the size of the input to be dialled by the user. It also contains the number of times an announcement can be played to a user without getting a corrcet user input, before being considered as failed.

- actions:
  The parameter provides the type of action, either PA or playannouncement&collectdigits, to be performed.

- preferredLanguage:
  The parameter gives the language that should preferrably used in user interactions. If the preferred language is not available it is the default language that should be used.

- numberOfAllowedRetries:
  The parameter provides the number of retries the user is allowed to make

- typeOfRequestedInfo:
  The parameter provides the type info to be requested to the user

- securityParameter:
  This optional parameter conveys securtity related information.

- userInformation:
  The parameter contains the information collected from the user

18.70.2 Invoking entity (supporting SCF)

18.70.2.1 Normal procedure

SCF Precondition:

(1) A "handlingInformationRequest" operation has been received.

(2) The SLPI has identified the need for more information than what is already available.

(3) The SCF FSM is in state "Assisting mode".

SCF Postcondition:

(1) The SCSM moves to the state "Assisting Mode".

After receiving a "handlingInformationRequest", the supporting SCF was not able to provide call instructions to the controlling SCF, because information known from the user is not available. The supporting SCF then sends a "provideUserInformationRequest" operation. One operation can be used to request several pieces of information.

Once the operation is sent, the supporting SCF moves to the state "Waiting for Additional Information" and awaits from the user information to be sent by the controlling SCF.

On receipt of the "provideUserInformationResult", the SCSM moves from the state "Waiting for Information" to the state "Assisting Mode". The user information received in the operation are used to provide assistance to the controlling SCF so that it can correctly handle the call. Only the supporting SCF knows how to use and to interpret the received information.

When the SCF receives an indication that a user interaction has failed, the SLPI decides upon the follow-up actions and especially, which kind of call instructions should be provided to the controlling SCF. In particular, the SCF can provided a specific user announcement in case of error if not already provided in the "provideUserInformation" operation.

18.70.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.70.3 Responding entity (controlling SCF)

18.70.3.1 Normal procedure

SCF Precondition:

(1) A "handlingInformationRequest" operation has been sent.

SCF Postcondition:

(1) The SCF moves to the state "Assisted Mode".

On receipt of the "provideUserInformationRequest" operation the SCSM moves from "Assisted Mode" to the state "Preparing Information". The SCF uses user interaction procedures to receive the requested information from the user (connection of an IP, STUI...).

The information received from the user can be challenged against the constraints provided in the parameters. If a user gives an improper response, a retry of the error message can occur (if allowed).

If several pieces of information are requested from the user, a failure to obtain one of them leads to the end of the user interaction phase or to the request of the next piece of information depending on the user interaction strategy adopted. In the first case only responded pieces are reported back to the supporting SCF, in the second case the failure is reported to the supporting SCF through a "taskRefused" error.

In the state "Preparing Information", the SCF collects information from the user. When all the user interactions have been performed successful or not, the SCSM moves to the state "Assisted Mode". It sends the "provideUserInformationResult" operation to the supporting SCF. The operation containing information provided by the user and/or indications that the procedures to get information from the user has failed.

When a user fails to correctly provide a piece of information even after the number of retries specified in the "provideUserInformationRequest" operation, it is the user interaction strategy parameter that determines the actions to be performed.

18.70.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.71 ReleaseAssociation procedure

18.71.1 General description

This operation is used to release an existing association between the users or a user and the network by the SCF..

18.71.1.1 Parameters

- Cause
  A number giving an indication to the CUSF about the reason of releasing this specific association. This may be used by CUSF to fill in the "cause" in the association releaseing message.

18.71.2 Invoking entity (SCF)

18.71.2.1 Normal procedure

SCF Precondition

(1) FSM for CUSF within the SCF is in the state N2: Preparing CUSF Instructions.

(2) SLPI requests to release the association between the user and the network.

SCF Postcondition

(1) FSM for CUSF within the SCF moves to the state N1: Idle.

18.71.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.71.3 Responding entity (CUSF)

18.71.3.1 Normal procedure

CUSF Precondition

(1) CUSF-FSM is in anys state, except "Idle".

CUSF Postcondition

(1) CUSF releases the association to the user.

(2) CUSF-FSM moves to the state a: Idle.

18.71.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.72 ReleaseCall procedure

18.72.1 General description

This operation is used by the SCF to tear down an existing call segment or segments at any phase of the call for all parties involved in the relevant call segments. This operation may not be sent to an assisting SSF, except in the case of hand-off procedure. For CAMEL, this operation shall only be sent within a control relationship.

18.72.1.1 Parameters

- initialCallSegment
  Indicates that the initial Call Segment shall be released
  - releaseCause:
    A number giving an indication to the SSF about the reason of releasing this specific call. This may be used by SSF for generating specific tones to the different parties in the call or to fill in the "cause" in the release message.
- associatedCallSegment
  Indicates that the an associated Call Segment shall be released
  - CallSegment:
    This parameter indicates the Call Segment to be released
  - releaseCause:
    A number giving an indication to the SSF about the reason of releasing this specific call. This may be used by SSF for generating specific tones to the different parties in the call or to fill in the "cause" in the release message.
- allCallSegments
  Indicates that all Call Segments within th eCall Segment Association shall be released
  - releaseCause:
    A number giving an indication to the SSF about the reason of releasing these calls. This may be used by SSF for generating specific tones to the different parties in the call or to fill in the "cause" in the release message.

18.72.2 Invoking entity (SCF)

18.72.2.1 Normal procedure

SCF Precondition:

(1) FSM for CS State "Preparing CS instructions" or State "Waiting for Notification or Request".

SCF Postcondition:

(1) FSM for CS State , "Idle", if neither "CallInformationReport" nor "ApplyChargingReport" has to be received from the SSF. All resources (e.g. queue) related to the call are released by the SCF, or
State "Waiting for Notification or Request" if a "CallInformationReport" or "ApplyChargingReport" still has to be received from the SSF.

18.72.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.72.3 Responding entity (SSF)

18.72.3.1 Normal procedure

SSF Precondition:

(1) FSM for CS State ," Waiting for Instructions", or
State "Monitoring"

SSF Postcondition:

(1) FSM for CS "Idle", state a, after sending any outstanding "CallInformationReport". Possible armed EDPs are ignored. All connections and resources related to the call are released.

18.72.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.73 RemoveEntry procedure

18.73.1 General description

The ITU-T Recommendation X.500 [36] "RemoveEntry" operation is used to request the SDF to remove a leaf entry (either a object entry or an alias entry) from the DIT. For a full description of the RemoveEntry operation, see ITU-T Recommendation X.511 [39] subclause 11.2.

18.73.1.1 Parameters

See ITU-T Recommendation X.511 [39] subclauses 11.2.2 and 11.2.3.

18.73.2 Invoking entity (SCF)

18.73.2.1 Normal procedure

SCF Preconditions:

(1) SCSM: "SDF Bound" or "Wait for Subsequent Requests"

SCF Postconditions:

(1) SCSM: "SDF Bound"

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the SL to remove an entry from the SDF exists, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response ((E5) Response_from_SDF_with_Bind or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the RemoveEntry operation was discarded. In the State "SDF Bound", the response of the RemoveEntry operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the RemoveEntry operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the SL to remove an entry from the SDF exists an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response ((E7) Response_from_SDF) to the RemoveEntry operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the RemoveEntry operation or an error.

18.73.2.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 11.2.4 and 11.2.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.73.3 Responding entity (SDF)

18.73.3.1 Normal procedure

SDF Preconditions:

(1) SDSM: "SCF Bound"or "Bind Pending"

SDF Postconditions:

(1) SDSM "SCF Bound"

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a "RemoveEntry" operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "RemoveEntry" operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before removing the entry item, the SDF takes the following actions:

- verify that the object to be removed exists and is a leaf entry;
- verify that the access rights to remove the entry are sufficient;

After the specified actions indicated above are successfully executed, the entry is removed from the SDF database. A null result is returned to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

18.73.3.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 11.2.4 and 11.2.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.74 ReportChargingInformation procedure

18.74.1 General description

The operation reports to an supporting SCF the outcome of the call in terms of charging information, the call being controlled by the controlling SCF. It may be a response to a previously issued "establishChargingRecord" with the "expectedReport" arameter positioned to TRUE or the first operation relating to the charge to the supporting SCF, if it has been prearranged. In the latter case, the call by call charging related information exchange by establishChargingRecord operation is not necessary, because the charging rate, etc. is pre-defined and is properly applied to the call in the controlling SCF.

18.74.1.1 Parameters

- uniquecallIdentity:
  This parameter is used to indicate the identity of the call that has motivated the use of the operation. There is a one-to-one relationship between this identity, the identity of the SLPI that treats the call and the identity of the assisting SLPI. This information can be further used to address a specific logic program instance.

- remainingUserCredit:
  This parameter contains the user's credit after the call. It is expressed in terms of telecommunication units.

- callRecord:
  This parameter contains the call record related to the call. This information consists (at least) of call duration, calling party and called party number.

- accountNumber:
  This parameter provides the unique identification of the account to which the cost of the call is registered

18.74.2 Invoking entity (controlling SCF)

18.74.2.1 Normal procedure

SCF Precondition:

(1) A "establishChargingRecord" operation has been received with the "reportExpected" parameter positionned to TRUE or it has been prearranged.

(2) A call has taken place.

(3) SCF FSM is in the "Assisted Mode" state

SCF Postcondition:

(1) SCF FSM remains in the "Assisted Mode" state.

After a call has taken place and either if a "establishChargingRecord" operation has been sent, requesting the caharging information to be reported or it has been prearranged, the SCF sends a "reportChargingInformation" operation to report the outcome of the charging procedure that took place for the call. It contains the call identity to know the call it refers to and the user credit after the call.

18.74.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16, and the TC services which are used for reporting operation errors are described in clause18.

18.74.3 Responding entity (supporting SCF)

18.74.3.1 Normal procedure

SCF Precondition:

(1) The SCF has sent an "establishChargingRecord" operation or it has been prearranged.

(2) SCF FSM is in the "Assisting Mode" state

SCF Postcondition:

(1) SCF FSM remains in the "Assisting Mode" state.

On receipt of the "reportChargingInformation" operation, the SCF uses the information to update the user's data and it could also report the information to the management functions for security and billing purposes.

18.74.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause18.

18.75 ReportUTSI procedure

18.75.1 General description

This operation is used to notify the SCFof an UTSI IE previously requested by the SCF in a *RequestReportUTSI* operation. The operation applies to call related (SSF) and call unrelated transactions (CUSF)..

This is a Class 4 operation.

18.75.1.1 Parameters

- USIInformation:
  This parameter conveys information provided by the User dedicated to the SL. It is transparent at the SSF level.

- LegID:
  This parameter indicates the party in the call or call unrelated association from which the UTSI information element has beenreceived..For the applied leg numbering refer to RequestReportBCSMEvent for a call associated transaction and to RequestReportBCUSMEvent for a call unrelated transaction.

- USIServiceIndicator:
  It indicates the SL requesting the Monitoring of an UTSI information element. It is used as a Monitoring criteria at the SSF/CUSF level. It also provides the correlation with the RequestReportUTSI operation.

18.75.2 Invoking entity (SSF/CUSF)

18.75.2.1 Normal procedure

SSF preconditions:

(1) The SSF FSM/CUSF FSM is in any state except "Idle".

(2) The SSF_USI FSM/CUSF_USI FSM is in the state "Monitoring UTSI IE".

SSF postconditions:

(1) The SSF FSM /CUSF FSM remains in the same state.

(2) The SSF_USI FSM/CUSF_USI FSM remains in the same state.

18.75.2.2 Error handling

Operation related error handling is not applicable, due to Class 4 operation.

18.75.3 Responding entity (SCF)

18.75.3.1 Normal procedure

SCF preconditions:

(1) A control or monitor relationship exists either between the SSF and the SCF (call reated) or between the CUSF and the SCF (call unrelated)..

(2) The SCF_USI FSM is in the state "Monitoring UTSI IE".

SCF postconditions:

(1) SLPI execution continues.

(2) The SCF_USI FSM remains in the same state.

18.75.3.2 Error handling

Operation related error handling is not applicable, due to Class 4 operation.

18.76 RequestNotification procedure

18.76.1 General description

This operation is used by the supporting SCF to request an event notification to the controlling SCF

18.76.1.1 Parameters

- Requested Notifcations:
  This parameter contains a set of call conditions which request the notification from controlling SCF when met
- securityParameter:
  This is an optional parameter conveying security related information

18.76.2 Invoking entity (supporting SCF)

18.76.2.1 Normal Procedure

SCF Precondition:

(1) A relationship with the controlling SCF has been established.

(2) The supporting SCF has previously received a "handlingInformationRequest" operation

(3) The SCF FSM is in the state "Assisting Mode"

SCF Postcondition:

(1) The SCF FSM remains in the same state

18.76.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.76.3 Responding entity (controlling SCF)

18.76.3.1 Normal Procedure

SCF Precondition:

(1) A "handlingInformationRequest" operation has been sent.

(2) The SCF FSM is in the state "Assisted Mode"

SCF Postcondition:

(1) According to the information received, the controlling SCF starts call monitoring functions.

(2) The SCF FSM remains in the same state

18.76.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16. and the TC services which are used for reporting operation errors are described in clause 18.

18.77 RequestNotificationChargingEvent procedure

18.77.1 General description

This operation is used to instruct the SSF how to manage the charging events which are received from other FE's not under the control of the service logic instance. The operation supports the options to cope with the interactions concerning the charging (refer to ETS 300 374-1 [10], annex B "Charging scenarios"). As several connection configurations may be established during a call a possibility exists for the RequestNotificationChargingEvent (RNC) operation to be invoked on multiple occasions. For each connection configuration a RNC may be used several times.

18.77.1.1 Parameters

- Sequence of ChargingEvent:
  This parameter contains a list of the charging events and the corresponding monitor types and corresponding legs. For each element in the list the following information elements are included:

  - eventTypeCharging:
    This sub parameter indicates the charging event type. Its content is network operator specific, which may be "charge pulses" or "charge messages".

  - monitorMode:
    This sub parameter indicates the monitorMode applicable for the corresponding "eventTypeCharging" sub parameter. Monitor may be "interrupted", "notifyAndContinue" or "transparent".

  - legID:
    This sub parameter indicates the leg id of the corresponding event type charging sub parameter.

  - eventTypeTariff:
    This sub-parameter indicates the charging event type related to tariff determination. The charging event may include information about tariff -, price -, tariff/price-acknowledgement -, or tariff/price-acknowledgement timer expiry.

18.77.2 Invoking entity (SCF)

18.77.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exist between the SCF and the SSF.

(2) An SLPI has determined that a "RequestNotificationChargingEvent" has to be sent by the SCF.

SCF Postconditions:

(1) No FSM state transition,

(2) SLPI execution may continue.

The SCSM FSM for the CS is in state "Preparing CS Instruction" or is in state "Queuing FSM". This operation is invoked by the SCF if a SLPI results in the instruction of SSF how to cope with the interactions concerning the charging . This causes no SCSM FSM state transition.

18.77.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.77.3 Responding entity (SSF)

18.77.3.1 Normal procedure

SSF Preconditions:

(1) FSM: for CS State "Waiting for Instructions", or

FSM for CS State "Waiting for End of User Interaction", or

FSM for CS State "Waiting for End of Temporary Connection", or

FSM for CS: State "Monitoring", or

Assisting/Hand-off SSF-FSM State "Waiting for Instructions".

SSF Postcondition:

(1) No FSM state transition

On receipt of this operation the SSF performs actions to cope with the interactions concerning the charging according to the information elements included in the operation. The requested charging event can be caused by: a) another SLPI or b) another exchange. Irrespective of by what the charging event is caused the SSF performs one of the following actions on occurrence of the charging event (according the corresponding monitorMode):

Interrupted:

Notify the SCF of the charging event using "EventNotificationCharging" operation, do not process the event or propagate the signal. However call and existing charging processing will not be suspended in the SSF.

NotifyAndContinue:

Notify the SCF of the charging event using "EventNotificationCharging", and continue processing the event or signal without waiting for SCF instructions (handled like EDP-N for BCSM events).

Transparent:

do not notify the SCF of the event. This ends the monitoring of a previously requested charging event.

Requested charging events are monitored until ended by a transparent monitor mode (or in the case of charging events) until the end of the connection configuration.

In the case that multiple "RequestNotificationChargingEvent" operations are received for the same connection configuration with the same "eventTypeCharging"/"eventTypeTariff" and "legID", only the last received "monitorMode" will apply.

18.77.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.78 RequestReportBCSMEvent procedure

18.78.1 General description

This operation is used to request the SSF to monitor for a call-related event (e.g., BCSM events such as busy or no answer), then send a notification back to the SCF when the event is detected.

NOTE: If the RequestReportBCSMEvent requests arming of the current DP from which the call processing was suspended, the next occurrence of the DP encountered during BCSM processing will be detected (i.e. not the current one from which the call was suspended).

The DP arming principle is as follows:

- The DPs O-Mid_Call, T-Mid_Call, O_Disconnect, T_Disconnect can be armed as well as for the controlling as passive legs depending on what direction (either from the party which is connected to the controlling or passive leg) events have to be captured. As an example the Disconnect DP can be armed as well as for the controlling leg and the passive leg, in that case if a release request is received from the user it will be detected by the Disconnect DP armed for the controlling leg, while a release request from the remote parties shall be detected by arming the relevant passive leg Disconnect DP.

- The O_Abondon DP can only be armed for the controlling leg in the O_BCSM and the T_Abandon can only be armed for the passive leg in the T_BCSM.

- All DPs that can occur before DP-Analysed-Information in the O_BCSM are allowed for controlling leg only;

- All DPs that can occur before PIC-SelectFacility in the T_BCSM are allowed for passive legs only;

- The DP-Analysed-Information is allowed for controlling and passive legs;

- All other DPs for which no filtering applies arming is alllowed in the O_BCSM for passive legs only and in the T_BCSM for controlling leg only.

Table 17-1: DP Arming Table for O-BCSM:

O_BCSM	Controlling leg	Passive leg)	Default leg ID
Origination_Attempt DP 0[1)]	-	-	-
Orig._Attempt_Authorized DP	X	-	1
Collected_Information DP	X	-	1
Analysed_Information DP	X	X	1
O_Term_Seized DP	-	X	2
Route_Select_Failure DP	-	X	2
O_Not_Reachable DP	-	X	2
O_Called_Party_Busy DP	-	X	2
O_No_Answer DP	-	X	2
O_Answer DP	-	X	2
O_Suspend DP	-	X	2
O_Re-Answer DP	-	X	2
O_Mid_Call DP	X	X	- o2[)]
O_Disconnect DP	X	X	- o2[)]
O_Abandon DP	X	-	1

o1) Only applicable as TDP, because first DP that can be encountered cannot be armed as EDP.
o2) The "legID" parameter shall be included
Nomenclature: X = Arming Applicable
- = Not Applicable

Table 17-2: DP Arming Table for T-BCSM:

T_BCSM	Controlling leg	Passive leg	Default Leg ID
Termination_Attempt DP t1)	-	-	-
Terminating_Attempt_Authorized DP	-	X	1
Facility_Selected_and_Available DP	X	-	2
Call_Accepted DP	X	-	2
T_Not_Reachable DP	X	-	2
T_Busy DP	X	-	2
T_No_Answer DP	X	-	2
T_Answer DP	X	-	2
T_Suspend DP	X	-	2
T_Re-Answer DP	X	-	2
T_Midcall DP	X	X	- t2)
T_Disconnect DP	X	X	- t2)
T_Abandon DP	-	X t3)	1

t1) Only applicable as TDP, because first DP that can be encountered cannot be armed as EDP.
t2) The "legID " parameter shall be included
t3) T_Abandon can only be armed for the passive leg.
Nomenclature: X = Arming Applicable
- = Not Applicable

18.78.1.1 Parameters

- bcsmEvents:
  This parameter specifies the event or events of which a report is requested.

  - eventTypeBCSM:
    This parameter specifies the type of event of which a report is requested. Values origAttempt and termAttempt are not valid for the eventTypeBCSM parameter.

  - monitorMode:
    This parameter indicates how the event should be reported. When the "monitorMode" is "interrupted", the event shall be reported as a request, if the "monitorMode" is "notifyAndContinue", the event shall be reported as a notification, if the "monitorMode" is "transparent", the event shall not be reported.

  - legID:
    This parameter indicates the party in the call for which the event shall be reported.
    When more than one event is requested the applied legIDs in the RequestBCSMEvent operation shall belong to the same CS. SCF will use the option "sendingSideID" only.

    - sendingSideID:
      The following values for "legID" are assumed:

NOTE: The IN CS1 definition of this parameter makes assumptions regarding the allocation of LegID values. With the introduction inIN CS2 of CCF, these assumptions are no longer appropriate. For IN CS2 the leg numbering is based on the following principles:

legID = 1 is the controlling leg and legID = 2 is the passive leg in case the initial call segment created was an originating call segment (CS state "Originating setup").
For InitiateCallAttempt the default legID = 1 for the passive leg to be created.
Additional legs can only be created by the SCF, in which case the SCF assigns the leg numbers.

legID = 1 is the passive leg and legID = 2 is the controlling leg (i.e. inverse to the above) in case the initial call segment created was a terminating call segment (CS state "Terminating setup"). Additional legs can only be created by the SCF, in which case the SCF assigns the leg numbers. For IN CS1 implementations in the case of a mid call trigger it was assumed that legID = 2 was assigned to the party not causing the trigger and legID = 1 was assigned to the party causing the trigger.

The "legID" parameter shall always be included for the events O-MidCall, O-Disconnect, T-MidCall and T-Disconnect.

- dPSpecificCriteria:
  This parameter indicates information specific to the EDP to be armed.

  - numberOfDigits:
    This parameter indicates the number of digits to be collected by the SSF for the CollectedInfo event. If the indicated number of digits is collected, SSF reports the event to the SCF.

  - applicationTimer:
    This parameter indicates the application timer for the NoAnswer event. If the user does not answer the call within the allotted time, the SSF reports the event to the SCF. This timer is expected to be shorter than the network no-answer timer.

  - midCallControlInfo:
    This parameter indicates the specific mid call events, which are requested to be monitored by the CCF/SSF. This parameter also indicates how the detected event needs to be reported in terms of report in monitoring state or always immediate report. When the parameter is not provided a default of report in monitoring state is applied.
    Several mid call events can be required in parallel.It may contain one or more control codes to define the specific events.

    - iNServiceControlCodeLow
      This parameter contains a single control code or the lower bound of a control code interval. Value "0" for single control code is used for hook flash for analogue line.

    - iNServiceControlCodeHigh
      This parameter contains the upper bound of a control code interval.

- - midCallReportType
  This parameter indicates how a detected midcall event is to be reported by the SSF to the SCF. It specifies a choice between:

  - report event when CS for FSM is in state monitoring or

  - report event when CS for FSM is in any state, except idle (immediate report).

18.78.2 Invoking entity (SCF)

18.78.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) The SLPI has decided that a request for an event report BCSM is needed.

(3) The SCSM FSM is in the appropriate state to send "RequestReportBCSMEvent".

SCF Postconditions:

(1) The SCSM FSM remains in the same state.

(2) SLPI execution continues.

(3) If all EDPs have been disarmed and neither a CallInformationReport nor an ApplyChargingReport is pending, the control relationship with the concerned SSF is ended. If no other relationship persists, the FSM for CSA shall return to "Idle" state.

18.78.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.78.3 Responding entity (SSF)

18.78.3.1 Normal procedure

SSF Precondition:

(1) The SSF FSM is in either the state "Waiting for Instructions" or the state "Monitoring". If the SSF FSM is in the state "Monitoring" only RequestReportBCSMEvent operations can be received with the monitorMode set to "transparent".

SSF Postconditions:

(1) The requested EDPs have been armed as indicated.

(2) Previously requested events are monitored until ended by a transparent monitor mode, until the end of the call, until the EDPs are detected or until the corresponding leg is released.

(3) The SSF FSM remains in the same state.

(4) If all EDPs have been disarmed and no "CallInformationReport" or "ApplyChargingReport" has been requested, the SSF FSM moves to the state "Idle".

18.78.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.79 RequestReportBCUSMEvent procedure

18.79.1 General description

This operation requests the CUSF to monitor for call unrelated events matching the requested criteria.

Table 17-3: DP Arming Table for BCUSM:

BCUSM	Controlling leg	Passive leg	Default Leg ID
Activation_Received_And_Authorized DP 1)	-	-	-
Component_Received DP	X	X	-
Association_Release_Requested DP	X	X	-
1) Only applicable as TDP, because first DP that can be encountered cannot be armed as EDP. 2) The "legID" parameter shall be included Nomenclature: X = Arming applicable - = Not Applicable			

18.79.2 Parameters

- BCUSMEvents
  This parameter indicates how and which DP should be reported (DP name and report mode, Notify, Interrupt, or transparent).

- monitorDuration
  This parameter indicates how long the CUSF should monitor the specified event.

- cUDPCriteria
  This parameter identifies the DP criteria for the requested DP. In particular the DP criteria corresponds to the parameter cUApplicationInd in the "InitialAssociationDP" operation.

- legID:
  This parameter indicates the party in the connection for which the event shall be reported. SCF will use the option "sendingSideID" only.

  - sendingSideID:
    The following values for "legID" are assumed:

    The leg numbering is based on the following principles:

    legID = 1 is the leg that has initiated the association (controlling leg) and legID = 2 is the the other leg (passive leg) . For InitiateAssociation the default legID = 2 for the leg to be created (passive leg).
    .

    The "legID" parameter shall always be included for the events ComponentReceived and AssociationReleaseRequested.

18.79.3 Invoking Entity (SCF)

18.79.3.1 Normal Procedure

SCF Precondition

(1) FSM for CUSF within the SCF is in the state N2.1: preparing CUSF instructions.

(2) SLPI requests to monitor for call unrelated events matching the requested criteria.

SCF Post condition

(1) FSM for CUSF within the SCF remains in the same state N2.1 Preparing CUSF Instructions.

18.79.3.2 Error Handling

If the error will occur within the SCF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 18.

18.79.4 Responding Entity (CUSF)

18.79.4.1 Normal Procedure

CUSF Precondition

(1) CUSF-FSM is in the state b: waiting for instructions or c: Monitoring.

CUSF Post condition ((1) or (2))

(1) CUSF starts the monitoring process for the specified event(s) or clearing armed EDP(s).

(2) CUSF-FSM remains in the same state.

(3) Requested events are monitored until the EDPs are detected or until the monitor duration is elapsed.

18.79.4.2 Error Handling

If the error will occur within the CUSF, generic error handling for the operation related errors are described in clause 16 and the TC service which are used for reporting operation errors are described in clause 3..3.4.

18.80 RequestReportUTSI procedure

18.80.1 General Description

This operation is used to request the SSF or CUSF to activate or deactivate the monitoring for the receipt of an UTSI IE with a given *ServiceIndicator* value. The operation applies to call related (SSF) and call unrelated transactions (CUSF)..

This is a class 2 operation.

18.80.1.1 Parameters

- requestedUTSI:
  This parameter specifies the UTSI IE or IEs of which a report is requested.

  - USIServiceIndicator:
    It indicates the SL requesting the Monitoring of an UTSI information element. It is used as a Monitoring criteria at the SSF/CUSF level. It also provides the correlation with a subsequent ReportUTSI operation.

- USIMonitorMode:
  This parameter indicates if the UTSI IE should be reported. When the "USImonitorMode" is "monitoringActive" the previously requested UTSI IE (e.g. the UTSI IE with the previously indicated ServiceIndicator value) is reported as a notification. When the "USImonitorMode" is "monitoringInactive", the UTSI IE is not (or no more) reported.

- LegID:
  This parameter indicates the party in the call or call unrelated association on which the UTSI IE is to be monitored.For the applied leg numbering refer to RequestReportBCSMEvent for a call related transaction and to RequestReportBCUSMEvent for a call unrelated transaction.. Only the legID parameter on argument level is used. If not present a default legID = 1 is assumed..

18.80.2 Invoking entity (SCF)

18.80.2.1 Normal procedure

SCF preconditions

(1) A control or monitor relationship exists either between the SSF and the SCF(call related) or between CUSF and SCF (call unrelated).

(2) The SLPI has decided that it is necessary to monitor for the receipt of an UTSI IE with a given ServiceIndicator value to precize.

(3) The SCF_USI FSM is in any state.

SCF postconditions:

(1) SLPI execution continues.

(2) The SCF_USI FSM moves to the state "Monitoring UTSI" (if the USIMonitorMode is "monitoringActive") or to the state "Idle" (if the USIMonitorMode is "monitoringInactive").

18.80.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.80.3 Responding entity (SSF/CUSF)

18.80.3.1 Normal procedure

SSF preconditions:

(1) The SSF FSM /CUSF FSM is any state except "Idle".

(2) The SSF_USI FSM/ CUSF_USI FSM is in any state.

SSF postconditions:

(1) The SSF FSM/CUSF FSM remains in the same state.

(2) The SSF_USI FSM /CUSF_USI FSM moves to the state "Monitoring UTSI" (if the USIMonitorMode is "monitoringActive") or to the state "Idle" (if the USIMonitorMode is "monitoringInactive").

18.80.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.81 RequestShadowUpdate procedure

18.81.1 General Description

The ITU-T Recommendation X.500 [36] "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525 [42].

18.81.1.1 Parameters

For the requestShadowUpdate operation, see ITU-T Recommendation X.525 [42], subclause 11.2.

18.81.2 Supplier entity (SDF)

18.81.2.1 Normal Pocedure

20.81.2.1.1 RequestShadowUpdate received by itself

SDF Preconditions:

(1) SDSM-ShM: "SDF Bound"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update" in case of success

(2) SDSM-ShM: "SDF Bound" in case of failure

The SDF is initially in the state "SDF Bound". After accepting the external event (E7~~5~~) Request_for_Shadow_from_Consumer caused by the reception of a "requestShadowUpdate" operation from the consumer SDF, a transition to the state "Wait for RequestShadow Result" occurs. The SDF performs the "requestShadowUpdate" operation according to the contents of the "requestShadowUpdate" argument. Once the SDF has completed the "requestShadowUpdate" operation, the result or error indication is returned to the consumer SDF. The SDF returns to the state "SDF Bound" if the "requestShadowUpdate" fails or to the state "Wait for Update" if the "requestShadowUpdate" is successful.

18.81.2.1.2 DSA ShadowBind sent with CoordinateShadowUpdate

18.81.2.1.3 RequestShadowUpdate received with DSAShadowBind

SDF Preconditions:

(1) SDSM-ShM: "Wait for Bind Result"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update" in case of success

(2) SDSM-ShM: "SDF Bound" in case of failure

The SDF is initially in the state "Wait for Bind Result" waiting for other operations to be received than the "DSAShadowBind" operation. When receiving the "RequestShadowUpdate" operation, a transition to the same state occurs through the external event (E3) Request_from_Consumer. The SDF performs the "DSAShadowBind" operation and a transition to the state "SDF Bound" occurs through the internal event (e5) SDF_Bind_Success. Since the "RequestShadowUpdate" operation has already been received, a transition to the state "Wait for RequestShadow Result" occurs through the external event (E7) Request_for_Shadow_from_Consumer. Then, the SDF performs the "RequestShadowUpdate" operation according to the contents of the "RequestShadowUpdate" argument. Once the SDF has completed the "RequestShadowUpdate" operation, the result or error indication is returned to the consumer SDF. The SDF returns to the state "SDF Bound" if the "RequestShadowUpdate" fails or to the state "Wait for Update" if the "RequestShadowUpdate" is successful.

18.81.3 Consumer entity (SDF)

18.81.3.1 Normal Procedure

18.81.3.1.1 RequestShadowUpdate sent by itself

SDF Preconditions:

(1) SDSM-ShC: "SDF Bound"

SDF Postconditions:

(1) SDSM-ShC: "Wait for Update" in case of success

(2) SDSM-ShC: "SDF Bound" in case of failure

When the SDSM-ShC is in the state "SDF Bound" and a need of requesting the shadow to be updated exists, an internal event occurs. This event, called (e9) Shadow_Request_to_Supplier, causes a transition to the state "Wait for RequestShadow Result" and the operation is sent to the supplier SDF. The SDSM-ShC waits for the response from the supplier. The reception of the response ((E12) Request_Shadow_Result) to the "requestShadowUpdate" operation previously issued to the supplier SDF causes a transition to the state "Wait for Update" if the result of the "requestShadowUpdate" operation is positive. Otherwise the reception of an error ((E11) Failed_Shadow_Request) moves back the SDSM-ShC to the state "SDF Bound".

18.81.3.1.2 RequestShadowUpdate sent with DSAShadowBind

SDF Preconditions:

(1) SDSM-ShC: "Wait for Subsequent Requests"

SDF Postconditions:

(1) SDSM-ShC: "Wait for Update" in case of success

(2) SDSM-ShC: "SDF Bound" in case of failure

When the SDSM-ShC is in the state " Wait for Subsequent Requests " and a need of requesting the shadow to be updated exists, an internal event occurs. This event, called (e2) Request_to_Supplier, causes a transition to the state "Bind with RequestShadow" and the operations are sent to the supplier SDF. The SDSM-ShC waits for the response from the supplier. When the DSAShadowBind is successful, the event (E5) SDF_Bind_Success causes a transition to the state "Wait for RequestShadow Result". The SDSM-ShC then waits for the response from the supplier. The reception of the response ((E12) Request_Shadow_Result) to the "requestShadowUpdate" operation previously issued to the supplier SDF causes a transition to the state "Wait for Update" if the result of the "requestShadowUpdate" operation is positive. Otherwise the reception of an error ((E11) Failed_Shadow_Request) moves the SDSM-ShC to the state "SDF Bound".

18.81.3.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause18.1.

18.82 ResetTimer procedure

18.82.1 General description

This class 2 operation is used by the SCF to refresh the T_{ssf} application timer, in order to avoid the T_{ssf}time-out at the SSF. It is also used by the SCF to refresh the Tcusf application timer in the CUSF.

18.82.1.1 Parameters

- timerID:
  This parameter has a default value identifying the T_{ssf} timer.
- timerValue:
  This parameter specified the value to which the $T_{ssf/Tcusf}$ is to be set.
- callSegmentIdentifier:
  This parameter indicates for which CS in the SSF the timer shall be reset. When not specified, the timer associated with the initial CS is assumed.

18.82.2 Invoking entity (SCF)

18.82.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship or association exists between the SCF and the SSF/CUSF.

(2) An SLPI has determined by the $T_{scf\text{-}ssf/Tscf\text{-}cusf}$ guard timer expiration, that the "ResetTimer" operation has to be sent in order to avoid T_{ssf} time-out at the SSF or Tcusf time-out at the CUSF..

SCF Postcondition:

(1) The SLPI reset the $T_{scf\text{-}ssf/Tscf\text{-}cusf}$ guard timer.

18.82.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.82.3 Responding entity (SSF)

18.82.3.1 Normal procedure

SSF Preconditions:

(1) Call origination attempt has been initiated.

(2) Basic call processing has been suspended at a DP.

(3) The FSM for the CS is in the "Waiting for Instruction" state or in the "Waiting for End of User Interaction" state or in the "Waiting for End of Temporary Connection" state.

NOTE: Whether the T_{SSF} is used or not in the state "Waiting for End of User Interaction" or in the state "Waiting for End of Temporary Connection" is network operator dependent.

SSF Postconditions:

(1) The T_{ssf} timer has been reset.

(2) The FSM for the CS remains in the same state.

18.82.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.83 SCFBind procedure

18.83.1 General description

The SCF Bind operation is used to establish relationship between two SCFs. This operation is mandatory and it is sent by a controlling SCF each time it needs to initiate communications with another (supporting) SCF; in order to ensure that the called entity has all facilities to operate on messages to be sent.

18.83.1.1 Parameters

- Agreement ID: this parameter identifies the SL type which will be supported across the SCF-SCF interface during the lifetime of the association. An agreement may correspond to a SL type which is standardized at the ITU level or by any regional standard organization. It may also correspond to a SL bilaterally defined by two cooperation operators. In order to allow the exchange of this information across networks, each agreement should be assigned an object ID. The AgreementID value is the object ID value.

- OriginatingSCF Adress: this parameter is the address of the SCF that requests the agreement. The parameter should be absent in a chanined operation request which crosses an international internetwork boundary.

- RespondingcalledSCFAddress: this parameter (in the response) contains the address of the supporting SCF. The parameter should be absent in a chanined operation request which crosses an international internetwork boundary.

- credentials:
  This is an optional parameter, placed either in the request or in the response, that conveys security credentials for requiring an authentication process on a per-association basis, with the authentication being done for a user or a Invoking entity (controlling SCF)

18.83.1.2 Normal procedure

SCF Precondition:

(1) The SCF FSM is in the state 1 "Idle".

(2) The SCF has received a user request that it is not able to handle, or the SCF has recognized a call condition previously registered as a call condition for which assistance from another SCF is needed.

(3) The SCF knews the address of the SCF being able to provide assistance

SCF Postconditions:

(1) The SCF FSM moves to state 2 "Preparing Request for Assistance"

The address of the SCF the "SCF Bind" operation has to be sent to is determined on the base of user related information.

18.83.1.3 Error handling

If the destination SCF is not accessible, the call is given final treatment which is SL dependent. Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.83.2 Responding entity (supporting SCF)

18.83.2.1 Normal procedure

SCF Precondition:

(1) The SCF FSM is in the state 1 "Idle"

SCF Postcondition:

(1) An SLPI has been invoked.

(2) The SCSM moves to state 2, "Processing SCF Bind", from state 1 "Idle"

(3) If the Bind request is accepted, a positive result is sent to the controlling SCF; if not, it moves back to state 1 "Idle" and a negative result is sent to the controlling SCF. In the first case when the first "handlingInformationRequest" operation is received, SCSM moves to state 3 "Assisting Mode".

18.83.2.2 Error handling

If the "SCF Bind" operation is rejected the SCSM remains in the state "Idle". The maintenance function is informed and no SLPI is invoked.

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.84 ScfBind procedure (in the chaining case)

18.84.1 General description

This operation is used to establish a relationship between the two SCFs involved in the chaining.

This is achieved by forwarding the SCFBind Information operation received from the controlling SCF to the chaining terminator supporting SCF"

18.84.1.1 Parameters

- Agreement ID: this parameter identifies the SL type which will be supported across the SCF-SCF interface during the lifetime of the association. An agreement may correspond to a SL type which is standardized at the ITU level or by any regional standard organization. It may also correspond to a SL bilaterally defined by two cooperation operators. In order to allow the exchange of this information across networks, each agreement should be assigned an object ID. The AgreementID value is the object ID value.

- OriginatingSCF Adress: this parameter is the address of the SCF that requests the agreement. The parameter should be absent in a chanined operation request which crosses an international internetwork boundary.

- RespondingcalledSCFAddress: this parameter (in the response) contains the address of the supporting SCF. The parameter should be absent in a chanined operation request which crosses an international internetwork boundary.

- credentials:
  This is an optional parameter, placed either in the request or in the response, that conveys security credentials for requiring an authentication process on a per-association basis, with the authentication being done for a user or a FE.

18.84.2 Invoking entity (chaining initiator supporting SCF)

18.84.2.1 Normal procedure

(1) The SCSM-ChI is in the state "Idle"

(2) The chaining initiator supporting SCF has received a "SCFBind"operation from the controlling SCF

SCF Post conditions:

(1) The SCSM-ChI moves to the state "Prepare Chained Handling Information Request"

18.84.2.2 Error handling

If the destination SCF is not accessible, the call is given final treatment which is SL dependent. Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.84.3 Responding entity (chaining terminator supporting SCF)

18.84.3.1 Normal procedure

SCF Precondition:

(1) The SCF FSM is in the state 1 "Idle"

SCF Postcondition:

(1) An SLPI has been invoked.

(2) The SCSM moves to state "Bind Pending"

(3) If the Bind request is accepted, a positive result is sent to the controlling SCF; if not, it moves back to state 1 "Idle" and a negative result is sent to the controlling SCF.

18.84.3.2 Error handling

If the "SCF Bind" operation is rejected the SCSM remains in the state "Idle". The maintenance function is informed and no SLPI is invoked.

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.85 SCFUnBind procedure

18.85.1 General description

The SCF Unbind operation is used by controlling SCF to close the relationship with the supporting SCF.

18.85.1.1 Parameters

None

18.85.2 Invoking entity (controlling SCF)

18.85.2.1 Normal procedure

SCF Precondition:

(1) The SCF FSM is in state 5 "Preparing SCF Unbind Request"

SCF Postconditions:

(1) The SCF FSM moves back to state 1 "Idle"

18.85.2.2 Error handling

If the destination SCF is not accessible, the call is given final treatment which is SL dependent. Generic error handling for the operation related errors is described in clause 16. and the TC services which are used for reporting operation errors are described in clause 18

18.85.3 Responding entity (supporting SCF)

18.85.3.1 Normal procedure

SCF Precondition:

(1) The SCF FSM is in any the state except 1 "Idle" and 2 "SCF Bind Pending"

SCF Postcondition:

(1) The SCSM moves back to state 1, "Idle"

18.85.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.86 ScfUnBind procedure (in the chaining case)

18.86.1 General description

The operation is used to close a relationship between the chaining initiator and chaining terminator SCFs.

18.86.1.1 Parameters

None

18.86.2 Invoking entity (chaining terminator supporting SCF)

18.86.2.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChI is in the state "SCF Bound"

(2) The chaining initiator supporting SCF has realized the need to close the relationship, either autonomously or because of the reception of a SCFUnbind operation from the controlling SCF

SCF Post conditions:

(1) The SCSM-ChI moves to the state "Idle"

18.86.2.2 Error handling

If the destination SCF is not accessible, the call is given final treatment which is SL dependent. Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.86.3 Responding entity (chaining terminator supporting SCF)

18.86.3.1 Normal procedure

SCF Preconditions:

(1) The SCSM-ChT is in the state "SCF Bound"

SCF Post conditions:

(1) The SCSM-ChT moves to the state "Idle"

(2) If the SCF Bind operation is accepted and chainedHandlingInformationRequest operation has been received, SCSM-ChT moves to state "SCFBound" and a positive result is sent to chaining initiator supporting SCF; if SCFBind is not accepted SCSM-ChT moves back to state Idle and a negative result is sent to chaining initiator supporting SCF.

18.86.3.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.87 ScriptClose procedure

18.87.1 General description

This operation is used to deallocate the resources used to perform the instance of the "User Interaction" script: the context is released.

18.87.1.1 Parameters

- UIScriptId:
  This parameter is used to address the User Interaction script.

- UIScriptSpecificInformation:
  This parameter is used to give to the SRF information dependant on the User Interaction script invoked.

- callSegmentIdentifier:
  This parameter indicates to which call segment the user interaction shall apply, i.e. to all parties connected to the call segment. When not provided, a default of 1 is assumed.

18.87.2 Invoking entity (SCF)

18.87.2.1 Normal procedure

SCF preconditions

(1) The SLPI has decided that it is necessary to close an instance of a given User Interaction script.

(2) The SCSM FSM is in state C3.2.1 "User Interaction", "R2" "Controlling SRF", A3 "User Interaction" or H3 "User Interaction".

SCF postconditions:

(1) SLPI execution continues.

(2) The SCSM FSM remains in the same state.

18.87.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.87.3 Responding entity (SRF)

18.87.3.1 Normal procedure

SRF preconditions:

(1) The SRF can receive order from the SCF.

(2) The SRF FSM is in the state "User Interaction". A User Interaction script is executing.

SRF postconditions:

(1) The SRF FSM is in the state "Connected".

(2) The User Interaction script has been terminated and resources used to perform the script execution have been released.

(3) Tsrf is started.

18.87.3.2 Error handling

Errors specific to the "SCF to SRF relationship based on the User Interaction script concept" should be added to the current list of errors.

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.88 ScriptEvent procedure

18.88.1 General Description

This operation is used to return information to the SCF on the results of the execution of the instance of User Interaction script (yes/no/cancel, ID+PIN, dialled number, etc.)

18.88.1.1 Parameters

- UIScriptId:
  This parameter is used to address the User Interaction script.

- UIScriptResultInformation:
  This parameter is used to give to the SCF the result of the User Interaction.

- callSegmentIdentifier:
  This parameter indicates to which call segment the user interaction shall apply, i.e. to all parties connected to the call segment. When not provided, a default of 1 is assumed.

- lastEventIndicator:
  This parameter means that the ScriptEvent operation contains the final result of the script execution. If the event is the final one for the UI Script, "TRUE" must be set.

18.88.2 Invoking entity (SRF)

18.88.2.1 Normal procedure

SRF preconditions:

(1) The SRF can send information to the SCF.

(2) The SRF FSM is in the state "User Interaction". A User Interaction script has been or is being executed. A condition is reached to send an intermediate or, if execution is completed, the final result.

SRF postconditions (intermediate result):

(1) The SRF FSM remains in the same state.

SRF postconditions (final result, implicit termination):

(1) Possible data about automatic disconnection of the bearer channel is checked.

(2) If no such data was present, the SRF FSM transits back to state "Connected". Tsrf is started.

(3) If such data was present and indicates "disconnection not allowed", the SRF FSM transits back to state "Connected". Tsrf is started.

(4) If such data was present and indicates "disconnection allowed", the SRF initiates bearer channel disconnection. The SRF FSM transits to state "idle".

SRF postconditions (final result, explicit termination):

(1) The SRF FSM remains in state "User Interaction"

(2) Tsrf is started.

(3) A ScriptClose operation is awaited.

18.88.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.88.3 Responding entity (SCF)

18.88.3.1 Normal procedure

SCF preconditions

(1) The SCF can receive information from the SRF.

(2) The SCSM FSM is in state "C.3.2.1 User Interaction, R2 Controlling SRF, A2 User Interaction or H2 User Interaction

SCF postconditions: (intermediate result)

(1) SLPI execution continues.

(2) The SCSM FSM remains in the same state.

(3) Result data has been passed to the SLPI (which may e.g. decide to send more information via ScriptInformation).

SCF postconditions: (final result, no other instruction to send)

(1) A ScriptClose may have been sent to explicitly terminate the script dialogue.

(2) Possible data about automatic SRF disconnection of the bearer channel is checked.

(3) If no such data was present, the SCSM FSM remains in the same state. SLPI execution continues

(4) If such data was present and indicates "disconnection not allowed", the SCSM FSM remains in the same state.

(5) If such data was present and indicates "disconnection allowed", the SCSM FSM transits back to state C2 Preparing CS Instructions, R1 SRF Control Idle, A1 Assisting SSF Idle or H2 Preparing SSF Instructions.

SCF postconditions: (final result, other instruction to send)

(1) A ScriptClose may have been sent to explicitly terminate the script dialogue.

(2) The SCSM FSM remains in the same state, SLPI execution continues.

18.88.3.2 Error handling

Errors specific to the "SCF to SRF relationship based on the User Interaction script concept" should be added to the current list of errors.

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.89 ScriptInformation procedure

18.89.1 General description

This operation is used to send to the SRF additional information during the User Interaction script execution.

18.89.1.1 Parameters

- UIScriptId:
  This parameter is used to address the User Interaction script.

- UIScriptSpecificInformation:
  This parameter is used to give to the SRF information dependant on the User Interaction script invoked.

- callSegmentIdentifier:
  This parameter indicates to which call segment the user interaction shall apply, i.e. to all parties connected to the call segment. When not provided, a default of 1 is assumed.

18.89.2 Invoking entity (SCF)

18.89.2.1 Normal procedure

SCF preconditions

(1) The SLPI has decided that it is necessary to send to the SRF additional information during the User Interaction script execution.

(2) The SCSM FSM is in state C3.2.1 User Interaction, R2 Controlling SRF, A2 User Interaction or H2 User Interaction.

SCF postconditions:

(1) SLPI execution continues.

(2) The SCSM remains in the same state.

(3) Further results are awaited.

The SLPI has identified the need to send some call information to the user. This information to the user can be sent in any state of the SCF FSM as long as the dialogue between the two SCFs exists (this excludes the state "Idle"). The "UserInformResquest" operation contain the announcement that should be played to the user and the medium that should be used.

Once the operation has been sent the SCF FSM remains in the same state and waits for the result.

18.89.2.2 Error handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.89.3 Responding entity (controlling SRF)

18.89.3.1 Normal procedure

SRF preconditions:

(1) The SRF can receive order from the SCF.

(2) The SRF FSM is in the state "User Interaction".

SRF postconditions:

(1) The SRF FSM remains in the same state.

SCF Precondition:

(1) A dialogue between the two SCFs has been established.

SCF Postcondition:

(1) The SCF prepares an answer to the received operation.

On receipt of the "UserInformResult" operation, the SCF remains in the same state and starts a procedure to pass over to the user information it has received. It is free of the procedure it adopts, but it should use the preferred language specified in the operation to give the information to the user. If the default language is not available, the default language should be used.

18.89.3.2 Error Handling

Generic error handling for the operation related errors is described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.90 ScriptRun procedure

18.90.1 General description

This operation is used to allocate the resources necessary to perform the instance of the "User Interaction" script (a context is partially defined for it) if necessary, then to activate this "User Interaction" script instance.

18.90.1.1 Parameters

- UIScriptId:
  This parameter is used to address the User Interaction script.
- UIScriptSpecificInformation:
  This parameter is used to give to the SRF information dependant on the User Interaction script invoked.
- callSegmentIdentifier:
  This parameter indicates to which call segment the user interaction shall apply, i.e. to all parties connected to the call segment. When not provided, a default of 1 is assumed.
- disconnectFromIPForbidden:
  This parameter indicates whether or not the SRF should be disconnected from the user when the user interaction script has been completed.

18.90.2 Invoking entity (SCF)

18.90.2.1 Normal procedure

SCF preconditions

(1) The SLPI has decided that it is necessary to run an instance of a given User Interaction script.

(2) The SCSM FSM is in state C3.2.1 User Interaction, R2 Controlling SRF, A3 User Interaction or H3 User Interaction.

SCF postconditions:

(1) SLPI execution continues.

(2) The SCSM FSM remains in the same state.

(3) A result is awaited.

18.90.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.90.3 Responding entity (SRF)

18.90.3.1 Normal procedure

SRF preconditions:

(1) A relationship between the SCF and SRF has been established.

(2) The SRF FSM is in the state "Connected", or in the state "UserInteraction" if no ScriptRun operation has not been previously sent

SRF postconditions (state Connected):

(1) The SRF FSM is in the state "User Interaction".

(2) Tsrf is stopped.

(3) The User Interaction script is executing.

SRF postconditions (state User Interaction):

(1) The SRF FSM is in the state "User Interaction".

(2) The ScriptRun operation is buffered.

18.90.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

Errors specific to the "SCF to SRF relationship based on the User Interaction script concept" should be added to the current list of errors.

18.91 Search procedure

18.91.1 General description

The ITU-T Recommendation X.500 [36] "Search" operation is used to search a portion of the SDF resident DIT for entries of interest and to return selected information from those entries. For a full description of the Search operation, see ITU-T Recommendation X.511 [39] subclause 10.2.

18.91.1.1 Parameters

See ITU-T Recommendation X.511 [39] subclauses 10.2.2 and 10.2.3.

18.91.2 Invoking entity (SCF)

18.91.2.1 Normal procedure

SCF Preconditions:

(1) SCSM: "SDF Bound" or "Wait for Subsequent Requests"

SCF Postconditions:

(1) SCSM: "SDF Bound"

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the SL to search and/or read information of the SDF exists, an internal event ((e2) Request_to_SDF) occurs. Until the application process has not indicated with a delimitor that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response ((E5) Response_from_SDF_with_Bind or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the Search operation was discarded. In the State "SDF Bound", the response of the Search operation ((E7) Response_from_SDF) causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the Search operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the SL to search and/or read information of the SDF exists an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response ((E7) Response_from_SDF) to the Search operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the Search operation or an error.

18.91.2.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 10.2.4 and 10.2.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.91.3 Responding entity (SDF)

18.91.3.1 Normal procedure

SDF Preconditions:

(1) SDSM: "SCF Bound"or "Bind Pending"

SDF Postconditions:

(1) SDSM "SCF Bound"

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a "Search" operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a "Search" operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before retrieving the data as specified in the operation parameters, the SDF takes the following actions:

- verify that the objects accessed by the request exists;
- verify that the user on behalf of whom the request is performed has sufficient access rights to access the objects and attributes reached during the execution of the operation;
- verify that attributes on which an operation should be performed exist in the object.

After the specified actions indicated above are successfully executed, the SDF returns all the possible attributes that satisfy the retrieval criteria to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

18.91.3.2 Error handling

Generic error handling for the operation related errors is described in ITU-T Recommendation X.511 [39] subclauses 10.2.4 and 10.2.5 and the TC services that are used for reporting operating errors are described in clause 18.

18.92 SendChargingInformation procedure

18.92.1 General description

This operation is used to instruct the SSF on the charging information to be sent by the SSF. The sending of charging information can either be by charge pulses or signalling or internal if SSF is located in the Local Exchange (LE). In the LE, either charge meter can be updated or a standard call record created. A possibility exists for the SendChargingInformation (SCI) operation to be invoked on multiple occasions. The charging scenario supported by this operation are: 3.2 (refer to ETS 300 374-1 [10], annex B "Charging scenarios")

NOTE: This operation has many PSTN/IN interactions.

18.92.1.1 Parameters

- sCIBillingChargingCharacteristics:
  This parameter indicates billing and/or charging characteristics. Its content is network operator specific. Depending on the applied charging scenario the following information elements can be included (refer to ETS 300 374-1 [10], annex B "Charging scenarios"):
  - charge level (scenario 3.2)
  - chargePulses
  - chargeMessages
- partyToCharge:
  This parameter indicates where the charging information must be sent.
- tariffMessage:
  This parameter includes the contents of the charging tariff message. It may include tariff -, price -, or acknowledgement information in the ChargingTariffInformation -, ChargingPriceInformation -, or ChargingAcknowledgementInformation sub-parameters.

18.92.2 Invoking entity (SCF)

18.92.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exist between the SCF and the SSF.

(2) An SLPI has determined that a "SendChargingInformation" has to be sent by the SCF.

SCF Postconditions:

(1) No FSM state transition,

(2) SLPI execution may continue.

The SCSM FSM is in state "Preparing SSF Instruction" or is in state "Queuing FSM".The SendChargingInformation procedure shall be invoked by the SCF in accordance with the demands of the SLPI for relevant charging information. If appropriate this information shall be sent back down the call path.

This causes no SCSM FSM state transition.

18.92.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.92.3 Responding entity (SSF)

18.92.3.1 Normal procedure

SSF Preconditions:

(1) FSM: for CS State "Waiting for Instructions", or
FSM for CS State "Waiting for End of User Interaction", or
FSM for CS State "Waiting for End of Temporary Connection", or
FSM for CS State "Monitoring", or
Assisting/hand-off SSF-FSM State b: "Waiting for Instructions".

SSF Postcondition:

(1) No FSM state transition

On receipt of this operation the SSF performs actions to send the charging information. The sending of charging information can either be by charge pulses or signalling or internal if SSF is located in LE. In the LE, either charge meter can be updated or a standard call record created. This operation has much PSTN/IN interactions.

For instance, by sending an operation "SendChargingInformation" the SCF instructs the SSF to initiate the PSTN/ISDN charging functions according to the given information about the charging level to use.

The charging level can be determined either by one of the following functions

(a) the SCF, or

(b) the SSF, or

(c) the charging function in a succeeding exchange.

In case the SCF has determined the charging level the "SendChargingInformation" operation contains the charging level to be applied.

In case the SSF determines the charging level the "SendChargingInformation" operation contains the parameters to determine the charging level.

If the charging level was determined by the IN (SCF or SSF) the SSF provides the charging level to be applied to the PSTN/ISDN charging functions (cases a and b).

In case c the charging level is determined in a succeeding exchange. The "SendChargingInformation" operation either contains the corresponding parameters indicating this fact or the SSF detects during trying to determine the charging level based on SCF provided parameters that the charging level shall be determined in a succeeding exchange. Based on already existing PSTN/ISDN capabilities the SSF provides the PSTN/ISDN charging functions with the necessary information and backward charge messages shall be transferred down the call path when allowed by the SCF (generated by a succeeding exchange for example an international gateway).

In the scenario described above charging/billing is performed by means of existing mechanisms of the PSTN/ISDN initiated and controlled by the IN.

That means the determination of the charging method - on-line or off-line - and the items to be charged for shall be done in the basic network, just like the charge generation and the charge registration.

18.92.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.93 SendSTUI procedure

18.93.1 General description

This operation is used to request the SSF to forward an STUI IE with a given ServiceIndicator value to the User. The operation applies to call related (SSF) and call unrelated transactions (CUSF)..

This is a Class 2 operation.

18.93.1.1 Parameters

- USIInformation:
  This parameter conveys information provided by the SL dedicated to the user. It is transparent at the SSF/CUSF level.

- LegID:
  This parameter indicates the party in the call or call unrelated association to which the STUI information element has to be sent. For the applied leg numbering refer to RequestReportBCSMEvent for a call related transaction and to RequestReportBCUSMEvent for a call unrelated transaction.

- USIServiceIndicator:
  This parameter indicates the SL which provides the STUI information.

18.93.2 Invoking entity (SCF)

18.93.2.1 Normal procedure

SCF preconditions:

(1) A control or monitor relationship exists either between the SSF and the SCF (call related) or between CUSF and SCF (call unrelated).

(2) The SLPI has decided that it is necessary to send to the User an STUI Information.

(3) The SCF_USI FSM is in any state.

SCF postconditions:

(1) SLPI execution continues.

(2) The SCF_USI FSM remains in the same state.

18.93.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.93.2.2.1 Responding entity (SSF/CUSF)

18.93.2.2.1.1 Normal procedure

SSF preconditions:

(1) The SSF FSM/ CUSF FSM is any state except "Idle".

(2) The SSF_USI FSM/CUSF_USI FSM is in any state.

SSF postconditions:

(1) The SSF FSM/CUSF FSMremains in the same state.

(2) The SSF_USI FSM/CUSF_USI FSM remains in the same state.

On receipt of this operation, the SSF/CUSF will forward the STUI to the User (identified by the LegID).

18.93.2.3 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services which are used for reporting operation errors are described in clause 18.

18.94 ServiceFilteringResponse procedure

18.94.1 General description

This operation is used to report the values of counters specified in a previous sent "ActivateServiceFiltering" operation to the SCF.

18.94.1.1 Parameters

- countersValue:
  The parameter contains the count of calls filtered during the filtering period. It is a list of counter identifications and the related values.

- filteringCriteria:
  This parameter is used to address the concerned SL at the SCF.

- responseCondition:
  This parameter is used to identify the reason why the ServiceFilteringResponse is sent.

  intermediate response indicates that service filtering is active, a call is received and the interval timer is expired, or that service filtering is active and the threshold value, "numberOfCalls", is reached.

  lastResponse indicates that the duration time is expired and service filtering is stopped, or that the stop time is met and service filtering is stopped.

18.94.2 Invoking entity (SSF)

18.94.2.1 Normal procedure

SSF Preconditions:

(1) Service filtering is running and the interval time is expired and a call is received, or

(2) Service filtering is running and the threshold value is reached, or

(3) Service filtering has been finished (duration time expired or stop time met), or

(4) The operation "ActivateServiceFiltering" is received and encounters an active service filtering entity.

SSF Postcondition:

(1) Service filtering proceeds or is ended depending on the duration time.

The SSF sends the "ServiceFilteringResponse" operation to the SCF. The "filteringCriteria" parameter is provided to enable the addressing of the concerned SL at the SCF.

Before "ServiceFilteringResponse" is sent, it is checked whether call gapping criteria are met. If so, the "Service-FilteringResponse" is not sent and the counting continues without resetting the counters. The last "ServiceFiltering-Response" (stop time is met or duration time expired) is sent without checking any call gap criteria.

After sending "ServiceFilteringResponse" the service filtering counters are reset.

If service filtering proceeds after sending "ServiceFilteringResponse" (e.g. interval time expired) the SSME-FSM remains in the state "Non-Call Associated Treatment".

If service filtering is stopped after sending "ServiceFilteringResponse" (duration time expired or stop time is met) then the SSME-FSM moves to the "Idle Management" state. All allocated resources are released, i.e. the SSME-FSM is removed as well.

18.94.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.94.3 Responding entity (SCF)

18.94.3.1 Normal procedure

SCF Preconditions:

(1) Service filtering is running.

(2) The SCME is in the state "Waiting for Service Filtering Response".

SCF Postcondition:

(1) The SCME forwards the received counter values to the SLPI.

The operation is handled by the Service Filtering FSM part of the SCME. The SCME passes the received counter values to the SLPI where they are added to previously received counter values.

The "filteringCriteria" parameter as provided in "ServiceFilteringResponse" is used to address the SCME and the concerned SL instance.

The Service Filtering FSM of the SCME remains in the state "Waiting For SSF Service Filtering Response" until the internal service filtering duration time in the SLPI expires. Then the SLPI informs the SCME about timer expiration. Now the SCME moves to the state "Service Filtering Idle".

18.94.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.95 SpecializedResourceReport procedure

18.95.1 General description

This operation is used as the response to a "PlayAnnouncement" operation when the announcement completed indication is set.

18.95.1.1 Parameters

None

18.95.2 Invoking entity (SRF)

18.95.2.1 Normal procedure

SRF Preconditions:

(1) The SRSM FSM is in the state "User Interaction".

(2) A "PlayAnnouncement" operation is being executed for which the parameter "RequestAnnouncementCompelete" was set TRUE.

(3) All information has been sent to the user.

SRF Postconditions:

(1) The SRSM FSM remains in the same state.

(2) If the "DisconnectFromIPForbidden" parameter was set FALSE, the SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the "SpecializedResourceReport" operation to the SCF. The SRSM FSM moves to the state "Idle".

18.95.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.95.3 Responding entity (SCF)

18.95.3.1 Normal procedure

SCF Precondition:

(1) The SCSM FSM is in the state "User Interaction", substate "Waiting for response from the SRF".

SCF Postconditions:

(1) The SCSM FSM remains in the same state.

(2) If the "SpecializedResourceReport" relates to a "PlayAnnouncement" operation with permission of SRF initiated disconnection, the SCSM FSM moves to the state "Preparing SSF Instructions".

18.95.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

18.96 SplitLeg procedure

18.96.1 General description

This operation is used to request the SSF to separate one party from its Call Segment andplace it in a new associated CS. This operation is the inverse of the MoveLeg or MergeCallSegments operation.

In splitting the specified leg, the conditions of the leg: the armed EDPs, the ApplyChargingReport pending, the EventNotificationCharging pending , and the CallInformationReport pending, are also applied for the same leg after split.

After the execution of the CPH operation the FSM models for the involved CS's in the SSF shall transit to the WFI state while the associated BCSM instances for the involved CS's shall move from the PIC to the DP of the corresponding mid call DP in order to handle subsequent EDP arming and call processing operation.

18.96.1.1 Parameters

- legToBeSplit:
  This parameter indicates the party in the call to be split from its Call Segment. See ITU-T Recommendation Q.1290 [29] "LegID".

- newCallSegment:
  This parameter indicates the CallSegmentID to be assigned to the newly created Call Segment.

18.96.2 Invoking entity (SCF)

18.96.2.1 Normal procedure

SCF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) An SLPI has determined that a call party shall be split from its current Connection Point.

(3) The FSM for CS is in state C2 "Preparing CS Instructions"

SCF Postconditions:

(1) SLPI execution may continue.

(2) The FSM for CS remains in the state C2 "Preparing CS Instructions".

18.96.2.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services used for reporting operation errors are described in clause 18.

18.96.3 Responding entity (SSF)

18.96.3.1 Normal procedure

SSF Preconditions:

(1) A control relationship exists between the SCF and the SSF.

(2) The leg to be split has the status "joined". If the SplitLeg is performed for a passive leg the corresponding controlling leg shall have the status "joined".

(3) The corresponding BCSM is in state O/T_Active, O/T_Suspended, O_Alerting or Send_Call.

(4) When the involved leg is an "outgoing" leg (i.e. the passive leg in an O_BCSM or the controlling leg in a T_BCSM), the corresponding BCSM shall be at least at the Send_Call PIC in case of an O_BCSM or T_Active in case of a T_BCSM.

SSF Postconditions:

(1) The SSF performs the necessary actions to separate the indicated leg from its original Call Segment and place it in a new associated Call Segment

(2) The SSF FSM of the new Call Segment moves to the "WaitingForInstructions" state.

(3) The FSM for the source and target Call Segments will move to the "Waiting for instructions" state. The associated BCSM instances within the two involved Call Segments will move from the PIC to the DP of the corresponding O_/T_MidCall DP, when not already suspended at a DP. Note that no MidCall EDP will be reported for this case.

(4) A Return Result is sent immediately after the successful change of the leg configuration is executed, this allows the SCF to be updated with the established connection view and to cater for possible interference problems with signalling events.

18.96.3.2 Error handling

Generic error handling for the operation related errors are described in clause 16 and the TC services used for reporting operation errors are described in clause 18.

18.97 UpdateShadow procedure

18.97.1 General Description

The ITU-T Recommendation X.500 [36] "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525 [42].

18.97.1.1 Parameters

pdateShadow operation, see ITU-T Recommendation X.525 [42], subclause 11.3.

18.97.2 Supplier entity (SDF)

18.97.2.1 Normal Pocedure

18.97.2.1.1 Supplier-initiated updateShadow

18.97.2.1.1.1 updateShadow sent by itself

SDF Preconditions:

(1) SDSM-ShM: "Wait for Update"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update"

When the SDSM-ShM is in the state "Wait for Update" and a need of updating the shadow exists, an internal event occurs. This event, called (e21) Shadow_Update_to_Consumer, causes a transition to the state "Wait for Update Confirmation" and the operation is sent to the consumer SDF. The SDSM-ShM waits for the response from the consumer. The reception of the response ((E23) Shadow_Update_Confirmed) to the "updateShadow" operation previously issued to the consumer SDF causes a transition to the state "Wait for Update". The response from the consumer SDF may be either the result of the "updateShadow" operation or an error.

18.97.2.1.1.2 updateShadow sent with DSAShadowBind and CoordinateShadowUpdate

SDF Preconditions:

(1) SDSM-ShM: "Bind with CoordinateShadow"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update"

When the SDSM-ShM is in the state " Bind with CoordinateShadow " and a need of updating the shadow exists, an internal event occurs. This event, called (e4) Update_to_Consumer, causes a transition to the state "Bind with CoordinateShadow and Update" and the operation is sent to the consumer SDF with a DSAShadowBind and CoordinateShadowUpdate operation. The SDSM-ShM then waits for the response from the consumer. The reception of the response (E9) SDF_Bind_Success to the previously issued DSAShadowBind causes a transition to the state ìBound with Coordinate Shadow Sent" . The SDSM-ShM waits for the response from the consumer to the "coordinateShadowUpdate" operation previously issued to the consumer SDF. The reception of the event ((E10) Shadow_Coordinate_Confirmed) causes a transition to the state "Wait for Update Confirmation". The reception of the response ((E23) Shadow_Update_Confirmed) to the "updateShadow" operation previously issued to the consumer SDF causes a transition to the state "Wait for Update". The response from the consumer SDF may be either the result of the "updateShadow" operation or an error.

18.97.2.1.2 Consumer-initiated updateShadow

18.97.2.1.2.1 updateShadow sent by itself

SDF Preconditions:

(1) SDSM-ShM: "Wait for Update"

SDF Postconditions:

(1) SDSM-ShM: "Wait for Update"

When the SDSM-ShM is in the state "Wait for Update" and a need of updating the shadow exists, an internal event occurs. This event, called (e13~~10~~) Shadow_Update_to_Consumer, causes a transition to the state "Wait for Update Confirmation" and the operation is sent to the consumer SDF. The SDSM-ShM waits for the response from the consumer. The reception of the response ((E14~~11~~) Shadow_Update_Confirmed) to the "updateShadow" operation previously issued to the consumer SDF causes a transition to the state "Wait for Update". The response from the consumer SDF may be either the result of the "updateShadow" operation or an error.

18.97.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause18.1.

18.97.3 Consumer entity (SDF)

18.97.3.1 Normal Procedure

18.97.3.1.1 Supplier-initiated UpdateShadow

18.97.3.1.1.1 UpdateShadow received by itself

SDF Preconditions:

(1) SDSM-ShC: "Wait for Update"

SDF Postconditions:

(1) SDSM-ShC: "Wait for Update"

The SDF is initially in the state "Wait for Update". After accepting the external event (E1310) Shadow_Update_from_Supplier caused by the reception of a "updateShadow" operation from the supplier SDF, a transition to the state "Wait for Update Confirmation" occurs. The SDF performs the "updateShadow" operation according to the contents of the "updateShadow" argument. Once the SDF has completed the "updateShadow" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "Wait for Update".

18.97.3.1.1.2 UpdateShadow received with DSAShadowBind and CoordinateShadowUpdate

SDF Preconditions:

(1) SDSM-ShC: ìWait for Bind Resultî

SDF Postconditions:

(1) SDSM-ShC: ìWait for Updateî

The SDF is initially in the state "Wait for Bind Result" waiting for other operations to be received than the "DSAShadowBind" operation. When receiving the "UpdateShadow" operation after receiving the "CoordinateShadowUpdate" operation, a transition to the same state occurs through the external event (E3) Request_from_Supplier. The SDF performs the "DSAShadowBind" operation and a transition to the state "SDF Bound" occurs through the internal event (e5) SDF_Bind_Success. Since the "CoordinateShadowUpdate" operation has already been received, a transition to the state "Wait for Coordination Result" occurs through the external event (E7) Shadow_Coordinate_from_Supplier. Then, the SDF performs the "CoordinateShadowUpdate" operation and a transition to the state "Wait for Update" occurs through the internal event (e10) Shadow_Coordinate_Confirmed. Since the "UpdateShadow" operation has also already been received, a transition to the state "Wait for Update Confirmation" occurs through the external event (E13) Shadow_Update_from_Supplier. Finally, the SDF performs the "UpdateShadow" operation according to the contents of the "updateShadow" argument. Once the SDF has completed the "updateShadow" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "Wait for Update".

18.97.3.1.2 Consumer-initiated updateShadow

SDF Preconditions:

(1) SDSM-ShC: "Wait for Update"

SDF Postconditions:

(1) SDSM-ShC: "Wait for Update"

The SDF is initially in the state "Wait for Update". After accepting the external event (E14) Shadow_Update_from_Supplier caused by the reception of a "updateShadow" operation from the supplier SDF, a transition to the state "Wait for Update Confirmation" occurs. The SDF performs the "updateShadow" operation according to the contents of the "updateShadow" argument. Once the SDF has completed the "updateShadow" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "Wait for Update".

18.97.3.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 [42] clause 12 and the TC services that are used for reporting operating errors are described in subclause 18.1.

19 Services assumed from lower layers

19.1 Services assumed from TCAP

The SS7 application layer protocol defined in this ITU-T Recommendation, is a protocol to provide communication between a pair of application processes. In the SS7 environment this is represented as communication between a pair of application-entities (AEs) using the TC. The function of an AE is provided by a set of application-service-elements (ASEs). The interaction between AEs is described in terms of their use of the services provided by the ASEs.

If AC are to be used for FE differentiation within a physical node then the version of TC used must support the dialogue portion of TC (ie ETS 300 287-1 [7]).

This requirement applies to all interfaces, not just those used for internetworking.

Table 18-1 defines which versions of TC are the minimum versions required to support the defined IN interfaces:

Table 18-1: Minimum TC requirements for INAP interfaces

Interface	IN CS2
SSF - SCF	Blue Book (1)
SCF - SRF	Blue Book (1)
SCF - SDF	ETS 300 287-1 [7]
SCF - SCF	ETS 300 287-1 [7]
SDF - SDF	ETS 300 287-1 [7]
CUSF - SCF	Blue Book (1)
Note: If AC negotiation is required then ETS 300 287-1 [7] is the minimum version required	

19.1.1 Common procedures

This subclause defines the procedures and mapping which apply between INAP and TC to be used in the absence of specific procedures and mapping instructions for the specific INAP interfaces as defined in subsequent subclauses.

19.1.1.1 Normal procedures

This subclause describes the procedures and TCAP primitives that shall be used for transmitting messages between AEs under normal operation.

The INAP, as TC-user, uses only the structured dialogue facility provided by TCAP. The following situations can occur when a message is sent between two PE:

- a dialogue shall be established: the TC-user issues a TC-BEGIN request primitive.
- a dialogue shall be maintained: the TC-user issues a TC-CONTINUE request primitive.
- a dialogue shall no longer be maintained: the TC-user issues a TC-END request primitive with either basic end or with pre-arranged end depending on the following conditions:
  - Basic End
    - In the case the dialogue is established, operations, leading to a termination of the relationship, can be transmitted by the FE with a TC-END request primitive (basic) in case the FE is not interested in the reception of any ERROR or REJECT components for these sent operations. Once the FE dialogue resources have been released, any ERROR or REJECT components received for these operations will be discarded by TC as described in ETS 300 287-1 [7].
    - In case the dialogue is established and the FE has received an operation, leading to the termination of the relationship, does not wish to continue dialogue and there is no operation to be sent, a TC-END request primitive (basic) with zero components can be sent from the FE.
  - Pre-arranged End
    In the case, an entity is interested in possible ERROR or REJECT messages on response to sent operations leading to a termination of the relationship, the dialogue is ended with a TC-END request primitive (pre-arranged end) after the last associated operation timer expires. The receiving entity can end the dialogue with a TC-END request primitive (pre-arranged end) after successful processing of these operations (i.e. the relationship is terminated).
- in general, the use of prearranged end shall be limited to the case for both communicating entities clearly recognizable that peer entity applies prearranged end. In all other cases, basic end shall be used.
- a dialogue shall not be established: for class 2 or 4 operations only the sending TC-user issues a TC-BEGIN request primitive and ends the dialogue locally after operation timeout by means of a prearranged end. Upon reception of the TC-BEGIN indication primitive the receiving TC-user shall end the dialogue locally.

19.1.1.2 Abnormal procedures

This subclause describes the procedures and TCAP primitives that shall be used for reporting abnormal situations between AEs. The error cases are defined in clause 16.

The following primitives shall be used to report abnormal situations:

- operation errors, as defined in the INAP, are reported with TC-U-ERROR request primitive.
- rejection of a TCAP component by the TC-user shall be reported with TC-U-REJECT request primitive.
- when the FE detecting error or rejecting operation decides the termination of TC dialogue, TC-END request primitive (basic) with error or reject can be used for the termination of TC dialogue.
- when the SSF, the SRF, or the CUSF detecting error or rejecting operation recognizes the possibility to continue dialogue, TC-CONTINUE request primitive with error or reject can be used for the continuation of TC dialogue.
- a dialogue shall be aborted by the TC-user with a TC-U-ABORT request primitive.
- on expiration of application timer TSSF or TSRF or TCUSF, dialogue shall be terminated by means of by TC-U-ABORT primitive with an Abort reason, regardless of TCAP dialogue is established or not.

For abnormal situations detected by TCAP the same rules shall apply for reception of TC-R-REJECT indication as for transmission of TC-U-REJECT request and for transmission of TC-P-ABORT indication as for transmission of TC-U-ABORT request primitive.

The following rules shall be applied to terminate the TCAP dialogue under abnormal situations:

- in the case that abort condition is detected and TCAP dialogue is established, TCAP dialogue is terminated by TC-U-ABORT primitive with an Abort reason.
- in the case that abort condition is detected and TCAP dialogue is not established, TCAP dialogue is locally terminated by TC-U-ABORT primitive. (in the case such as application time out).

In error situations prearranged end shall not be used to terminate the TCAP dialogue. In case any AE encounters an error situation the peer entity shall be explicitly notified of the error, if possible. If from any entity"s point of view the error encountered requires the relationship to be ended, it shall close the dialogue via a TC-END request primitive with basic end or via a TC-U-ABORT request primitive, depending on whether any pending ERROR or REJECT component is to be sent or not.

In case an entity receives a TC-END indication primitive and after all components have been considered, the FSM is not in a state to terminate the relationship, an appropriate internal error should be provided.

In cases when a dialogue needs to be closed by the initiating entity before its establishment has been completed (before the first TC indication primitive to the TC-BEGIN request primitive has been received from the responding entity), the TC-user shall issue a TC-END request primitive with prearranged end or a TC-U-ABORT request primitive. The result of these primitives will be only local, any subsequent TC indication received for this dialogue will be handled according to the abnormal procedures as specified in ETS 300 287-1 [7](ETS 300 287-1 [7]).

19.1.1.3 Dialogue handling

19.1.1.3.1 Dialogue establishment

The establishment of an INAP dialogue involves two application processes as described in clause 1, one that is the dialogue-initiator and one that is the dialogue-responder.

AC negotiation may not be supported in all PE and/or all networks.

This procedure is driven by the following signals:

- A TC-BEGIN request primitive from the dialogue-initiator.
- A TC-BEGIN indication primitive occurring at the responding side
- The first TC-CONTINUE indication primitive occurring at the initiating side or under specific conditions:
  - A TC-END indication primitive occurring at the initiating side
  - A TC-U-ABORT indication primitive occurring at the initiating side
  - A TC-P-ABORT indication primitive occurring at the initiating side

Sending of a TC-BEGIN request

Before issuing a TC-BEGIN request primitive, TC-USER shall store the AC-name and if present the user-information parameter.

TC-USER shall request the invocation of the associated operations using the TC-INVOKE service. See subclause 18.1.1.4.1 for a description of the invocation procedure.

After processing of the last invocation request, TC-USER shall issue a TC-BEGIN request primitive.

The initiator TC-USER then waits for a TC indication primitive and will not issue any other requests, except a TC-U-ABORT request or a TC-END request with the release method parameter set to "pre-arranged release".

If no TC indication primitive is expected because no dialogue is to be established according to the rules as stated in subclauses 18.1.2.1.1 and 18.1.2.1.2, TC-USER will wait for the last associated TCAP operation timer to expire and issue a TC-END request with the release method parameter set to "pre-arranged release".

Receipt of a TC-BEGIN indication

On receipt of a TC-BEGIN indication primitive, responder TC-USER shall:

- Analyse the application-context-name if included in the primitive. If it is supported, process any other indication primitives received from TC as described in clause 18.1.1.4.1.
- If no dialogue is to be established according to the rules as stated in subclauses 18.1.2.1.1 and 18.1.2.1.2, TC-USER will wait for the last indication primitive from TC and issue a TC-END request with the release method parameter set to "pre-arranged release".
- If the application-context-name included in the primitive is not supported, issue a TC-U-ABORT request primitive. If an alternative application-context can be offered its name is included in the TC-U-ABORT request primitive.

It is for further study whether or not the application-context-negotiation is limited only for using the TC-U ABORT primitive.

Receipt of the first TC-CONTINUE indication

On receipt of the first TC-CONTINUE indication primitive for a dialogue, TC-USER shall check the value of the application-context-name parameter. If this value matches the one used in the TC-BEGIN request primitive, TC-USER shall process the following TC component handling indication primitives as described in subclause 18.1.1.4.1, otherwise it shall issue a TC-U-ABORT request primitive.

It is for further study whether or not the application-context-negotiation is limited only for using the TC-U ABORT primitive.

Receipt of a TC-END indication

On receipt of a TC-END indication primitive in the dialogue initiated state, TC-USER shall check the value of the application-context-name parameter. If this value match the one used in the TC-BEGIN request primitive, then the TC-USER shall process the following TC component handling indication primitives as described in subclause 18.1.1.4.1.

Receipt of a TC-U-ABORT indication

Receipt of a TC-U-ABORT indication primitive is described as part of user abort procedure (See 18.1.1.3.4.) If the abort reason is AC name not supported, the responding side may propose an alternative AC name in the TC-U-ABORT indication. If an alternative AC is proposed the receiving entity shall check this name and if it can be supported a new dialogue may be established.

Receipt of a TC-P-ABORT indication

Receipt of a TC-P-ABORT indication primitive is described as part of provider abort procedure (See 18.1.1.3.5.)

19.1.1.3.2 Dialogue continuation

Once established the dialogue is said to be in a continuation phase.

Both application processes can request the transfer of INAP APDUs until one of them requests the termination of the dialogue.

Sending entity

TC-USER shall process any component handling request primitives as described in subclause 18.1.1.4.1.

After processing the last component handling request primitive, TC-USER shall issue a TC-CONTINUE request primitive.

Receiving entity

On receipt of a TC-CONTINUE indication primitive TC-USER shall accept zero, one or several TC component handling indication primitives and process them as described in subclause 18.1.1.4.1.

19.1.1.3.3 Dialogue termination

Both the dialogue-initiator and the dialogue-responder have the ability to request the termination of a dialogue after it has been established when no dialogue is to be established or when a dialogue is no longer to be maintained according to the rules as stated in subclauses 18.1.2.1.1 and 18.1.2.1.2.

The dialogue termination procedure is driven by the following events:

- A TC-END request primitive
- A TC-END indication primitive

Sending of TC-END request

When the dialogue shall no longer be maintained, TC-USER shall process any component handling request primitives as described in subclause 18.1.1.4.1

After processing the last component handling request primitive (if any), TC-USER shall issue a TC-END request primitive with the release method parameter set to "basic end" or "prearranged release", according to the rules as stated in subclauses 18.1.2.1.1 and 18.1.2.1.2.

When no dialogue is to be established, refer to subclauses 18.1.1.3.1.

Receipt of a TC-END indication

On receipt of a TC-END indication primitive, the TC-USER shall accept any component handling indication primitives and process them as described in subclause 18.1.1.4.1.

After processing the last component handling primitive all dialogue related resources are released.

19.1.1.3.4 User abort

Both the dialogue-initiator and the dialogue-responder have the ability to abort a dialogue at any time.

The user abort procedure is driven by one of the following events:

- A TC-U-ABORT request primitive
- A TC-U-ABORT indication primitive

Sending of TC-U-ABORT request

After issuing a TC-U-ABORT request primitive, all dialogue related resources are released.

Receipt of a TC-U-ABORT indication

On receipt of a TC-U-ABORT indication all dialogue related resources are released.

19.1.1.3.5 Provider abort

TC has the ability to abort a dialogue at both the dialogue-initiator side and the dialogue-responder side.

The provider abort procedure is driven by the following event:

- A TC-P-ABORT indication primitive

Receipt of a TC-P-ABORT indication

On receipt of a TC-P-ABORT indication, all dialogue related resources are released.

19.1.1.3.6 Mapping to TC dialogue primitives

The TC-UNI service is not used by INAP.

The mapping of parameters onto the TC Dialogue services is as follows:

The use of parameters of the TC-BEGIN service is as defined in subclause 18.1.1.3.7 with the following qualifications:

- The Destination Address parameter of the TC-BEGIN service shall be set to the INAP address of the AE which is to respond to the TC-BEGIN service.

NOTE 1: The address used in this parameter may be mapped by SCCP address translation to one of a number of alternative AEs.

- The AC Name parameter of the TC-BEGIN service shall be set according to the specific interface being used between the initiating AE and the responding AE.

- The Originating Address parameter of the TC-BEGIN service shall be set to the unambiguous INAP address of the AE initiating the TC-BEGIN service.

The use of parameters of the TC-CONTINUE service is as defined in subclause 18.1.1.3.7 with the following qualifications:

- The AC Name parameter of the TC-CONTINUE service shall be set to the value of the AC Name parameter of the TC-BEGIN service for the same Dialogue ID parameter value.

- If present, the Originating Address parameter of the TC-CONTINUE service shall be set to the unambiguous INAP address of the AE initiating the TC-CONTINUE service. This parameter is only present in the first TC-CONTINUE service after a TC-BEGIN service with the same Dialogue ID parameter value.

The use of parameters of the TC-END service is as defined in subclause 18.1.1.3.7 with the following qualifications:

- The AC Name parameter of the TC-END service shall be set to the value of the AC Name parameter of the TC-BEGIN service for the same Dialogue ID parameter value. This parameter is only present if the TC-END service is used immediately after the TC-BEGIN service.

The use of parameters of the TC-U-ABORT service is as defined in subclause 18.1.1.3.7 with the following qualifications:

- The Abort Reason parameter of the TC-U-ABORT service shall be used as specified in ETS 300 287-1 [7].

- The AC Name parameter of the TC-U-ABORT service shall be set to either the value used in the TC-BEGIN service or an alternative value which can be used to establish the dialogue between the initiating AE and the responding AE.

NOTE 2: This parameter is only present if the TC-U-ABORT is the immediate response to a TC-BEGIN indication.

The use of parameters of the TC-P-ABORT service is as defined in subclause 18.1.1.3.7 with the following qualifications:

- The P-Abort parameter of the TC-P-ABORT service is set by TC to indicate the reason why TC aborted the dialogue. It shall take the values as defined in ETS 300 287-1 [7].

19.1.1.3.7 Default mapping to TC dialogue parameters

Dialogue Id

The value of this parameter is associated with the INAP invocation in an implementation dependent manner. This parameter uniquely identifies a specific TC dialogue to a remote INAP AE for an INAP AE.

Application-context-name

The application-context-name parameter is set according to the set of operations which need to be supported by the TC dialogue. The defined AC Names can be found in clauses 5 to 10.

User information

This parameter may be used by both initiating and responding application process in a network operator specific manner.

Component present

This parameter is used by TC-USER as described in ETS 300 287-1 [7].

Termination

The value of the release method parameter of the TC-END request primitive is set by TC-USER according to the rules as stated in subclauses 18.1.2.1.1 and 18.1.2.1.2.

Quality of service

The quality of service of TC request primitives is set by the TC-USER to the following value:

- Sequencing requested
- return option, this parameter is set by TC-USER in an implementation dependent manner

19.1.1.4 Component handling

19.1.1.4.1 Procedures for INAP operations

This subclause describes the procedures for INAP operations.

Operation invocation

TC-USER shall build an operation argument from the parameters received and request the invocation of the associated operation using the TC-INVOKE procedure. If a linked ID parameter is inserted in the primitive this indicates a child operation and implies that the operation is linked to a parent operation.

Operation invocation receipt

On receipt of a TC-INVOKE indication primitive, TC-USER shall

- If the operation code does not correspond to an operation supported by the application-context, request the transfer of a reject component using the TC-U-REJECT request primitive, with the appropriate problem code (unrecognized operation);
- If a linked ID is included, perform the following checks: If the operation referred to by the linked ID does not allow linked operations or if the operation code does not correspond to a permitted linked operation, or if the parent operation invocation is not active, issue a TC-U-REJECT request primitive with the appropriate problem code (linked response unexpected or unexpected linked operation);
- If the type of the argument is not the one defined for the operation, request the transfer of a reject component using the TC-U-REJECT request primitive, with the appropriate problem code (mistyped parameter);
- if the operation cannot be invoked because the INAP related dialogue is about to be released, requests the transfer of the reject component using the TC-U-REJECT request primitive with the problem code (Initiating Release);
- if sufficient INAP related resources are not available to perform the requested operation, request the transfer of a reject component using the TC-U-REJECT request primitive with the problem code (Resource Limitation);
- Otherwise, accept the TC-INVOKE indication primitive. If the operation is to be user confirmed, TC-USER waits for the corresponding response.

Operation Response

For user confirmed operations, TC-USER shall:

- If no error indication is included in the response to a class 1 or 3 operation, construct a result information element from the parameters received and request its transfer using the TC-RESULT-L service.

- If an error indication is included in the response to a class 1 or 2 operation, construct an error parameter from the parameters received and request its transfer using the TC-U-ERROR request primitive.

Receipt of a response

On receipt of a TC-RESULT-NL indication, TC-USER shall:

- Request the transfer of a reject component using the TC-U-REJECT request primitive, with the appropriate problem code (mistyped parameter).

On receipt of a TC-RESULT-L indication, TC-USER shall:

- if the type of the result parameter is not the one defined for the result of this operation, request the transfer of a reject component using the TC-U-REJECT request primitive, with the appropriate problem code (mistyped parameter);

- otherwise, accept the TC-RESULT-L indication primitive.

On receipt of a TC-U-ERROR indication, TC-USER shall:

- if the error code is not defined for the TC-USER or is not one associated with the operation referred to by the invoke ID, request the transfer of a reject component using the TC-U-REJECT request primitive, with the appropriate problem code (unrecognized error or unexpected error);

- if the type of the error parameter is not the one defined for this error, request the transfer of a reject component using the TC-U-REJECT request primitive, with the appropriate problem code (mistyped parameter);

- Otherwise, accept the TC-U-ERROR indication primitive.

On receipt of a TC-U-REJECT indication primitive which affects a pending operation, TC-USER shall:

- accept the TC-U-REJECT indication primitive.

On receipt of a TC-L-REJECT indicating "return result problem, return error unexpected", TC-USER shall inform the application process.

On receipt of a TC-L-REJECT indicating "return error problem, return error unexpected", TC-USER shall inform the application process.

This event occurs when the local TC detects a protocol error in an incoming component which affects an operation.

When the problem code indicates a general problem, it is considered that the event cannot be related to an active operation even if the invoke Id is provided by TC. This is because it is unclear whether the invoke Id refers to a local or remote invocation. The behaviour of TC-USER in such a case is described in the subclause headed "other events"..

On receipt of a TC-L-CANCEL indication, the TC-USER shall:

- if the associated operation is a class 1 operation, inform the application process;

- if the associated operation is a class 2 operation and no linked operations are defined for this operation, ignore the primitive;

- if the associated operation is a class 2 operation and has linked operations but none of them has been invoked, inform the application process;

- if the associated operation is a class 2 operation and a linked operation invocation has already been received in response to this operation, ignore the primitive;

- if the associated operation is a class 3 operation, inform the application process;

- if the associated operation is a class 4 operation, ignore the primitive;

Other events

This subclause describes the behaviour of TC-USER on receipt of a component handling indication primitive which cannot be related to any operation or which does not affect a pending one.

On receipt of a TC-U-REJECT indication primitive which does not affect an active operation (i.e. indicating a return result or return error problem), it is up to the application process to abort, continue or terminate the dialogue, if not already terminated by the sending application process according to the rules as stated in subclause 18.1.2.1.2. This is also applicable for invoke problems related to a class 4 linked operation.

On receipt of a TC-R-REJECT indication (i.e. when a protocol error has been detected by the peer TC entity) which does not affect an active operation, it is up to the application process to abort, continue or terminate the dialogue, if not already terminated by the sending application process according to the rules as stated in subclause 18.1.2.1.2.

On receipt of a TC-L-REJECT indication primitive (i.e. when a protocol error has been detected by the local TC entity) which cannot be related to an active operation, it is up to the application process to continue, or to terminate the dialogue and implicitly trigger the transmission of the reject component or to abort the dialogue.

On receipt of a TC-NOTICE indication primitive, which informs the TC-USER that a message cannot be delivered by the Network Layer, it is for the application process to decide whether to terminate the dialogue or retry.

This primitive can only occur if the Return Option has been set (see clause 18.1.1.3.6).

19.1.1.4.2 Mapping to TC component primitives

The mapping of parameters onto the TC Component services is as follows:

The TC-U-CANCEL service is not used.

The TC-RESULT-NL service is not used.

The use of parameters of the TC-INVOKE service is as defined in subclause 18.1.1.4.3 with the following qualifications:

- The Operation parameter of the TC-INVOKE service shall contain the *operation.&operationCode* value of the INAP operation to be invoked. The operation must be one of the valid operations supported by the negotiated AC for the TC dialogue and must be invokable by the local AE.

- The Parameters parameter of the TC-INVOKE service shall contain a value of the *operation.&ArgumentType* value for the operation being invoked, as specified by the Operation parameter.

The use of parameters of the TC-RESULT-L service is as defined in subclause 18.1.1.4.3 with the following qualifications:

- The Invoke Id parameter of the TC-RESULT-L service shall be set to the value of the Invoke Id parameter of the TC-INVOKE service from the remote AE to which a result is being sent.

- The Operation parameter of the TC-RESULT-L service be set to the value of the Operation parameter of the TC-INVOKE service from the remote AE which contains the same Invoke Id Parameter value.

- The Parameters parameter of the TC-RESULT-L service shall contain the *operation.&ResultType* value for the operation result, as specified by the Operation parameter.

The use of parameters of the TC-U-ERROR service is as defined in subclause 18.1.1.4.3 with the following qualifications:

- The Invoke Id parameter of the TC-U-ERROR service shall be set to the value of the Invoke Id parameter of the TC-INVOKE service from the remote AE to which an error is being sent.

- The Error parameter of the TC-U-ERROR service shall be set to the value of the *error.&errorCode* of the error to be sent. It must be one of the errors which is expected for the invoked operation as defined in the *operation.&Errors* specification

- The Parameters parameter of the TC-U-ERROR service shall be set to the value of the *error.&ParameterType* of the error to be sent, as identified by the Error parameter.

The use of parameters of the TC-U-REJECT service is as defined in subclause 18.1.1.4.3 with the following qualifications:

- The Invoke Id parameter of the TC-U-REJECT service shall be set to the Invoke Id Parameter of the TC component service from the remote AE which is being rejected.

The use of parameters of the TC-L-CANCEL service is as defined in subclause 18.1.1.4.3.

19.1.1.4.3 Default mapping to TC component parameters

Invoke Id

This parameter is set by the sending application process. It represents the unique identity of an instance of an operation which is invoked by a AE within a specific TC dialogue. The TC dialogue is identified by the Dialogue Id parameter.

Linked Id

This parameter is set by the sending application process. It represents the Invoke Id of an operation which was received from the remote AE for a specific TC dialogue to which the operation being invoked by the local AE is to be linked. This parameter is only present if the original operation invoked by the remote AE is defined as having linked operations. The type of local operation invoked must be the same type as one of the operations defined as being linked.

Dialogue Id

The value of this parameter is associated with the INAP invocation in an implementation dependent manner. It represents the identity of the established TC dialogue which will carry the component services between the local AE and the remote AE.

Class

The value of this parameter is set according to the type of the operation to be invoked according to the operation definitions in clauses 5 through 10.

Time out

The value of this parameter is set according to the type of operation invoked.

Last component

This parameter is used as described in ETS 300 287-1 [7].

Problem code

This parameter is used as described in subclause 18.1.1.4.1.

Abort reason

This parameter is used by TC-USER, and attributes and coding are specified by network operator.

19.1.2 SSF-SCF interface

19.1.2.1 Normal procedures

19.1.2.1.1.1 SSF-to-SCF messages

This subclause defines the normal procedures for TC messages from the SSF to the SCF.

SSF-FSM related messages

A dialogue shall be established when the SSF-FSM moves from the state **Idle** to the state **Active**. The relevant INAP operation, which can be the InitialDP operation or one of DP specific operations for TDP-R, shall be transmitted in the same message.

The INAP operation InitialDP shall be sent with a TC-BEGIN request primitive and the dialogue is locally ended by means of TC-END request primitive with prearranged end.

For all other operations sent from the SSF-FSM, the dialogue shall be maintained except for the following cases.

When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is one or more pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with component(s). When the SSF sends the last EventReportBCSM, ApplyChargingReport or CallInformationReport the dialogue may be ended from the SSF by a TC-END request primitive with basic end.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component or prearranged end. When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components, or the dialogue is locally ended by means of a TC-END request primitive with prearranged end.

In the case where a call release is initiated by any other entity than an SCF, the SSF can end a dialogue with a TC-END request primitive with zero component or prearranged end if a TCAP dialogue is established and the SSF has no pending call information requests (or pending requests which should be treated in the same way, see subclause 18.1.1.1) nor any armed EDP.

When the SSF has sent the last EventReportBCSM, ApplyChargingReport or CallInformationReport the dialogue may be ended from the SCF by a TC-END request primitive with basic end.

Assisting/Hand-off SSF FSM related messages

A dialogue shall be established when the Assisting/Hand-off SSF-FSM moves from the state **Idle** to the state **Active**. The AssistRequestInstructions operation shall be transmitted with a TC-BEGIN request primitive.

For all other operations sent from the Assisting/Hand-off SSF-FSM, the dialogue shall be maintained except for the following cases.

When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is one or more pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with component(s). When the SSF sends the last ApplyChargingReport, the dialogue may be ended from the SSF by a TC-END request primitive with basic end.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component or prearranged end. When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components, or the dialogue is locally ended by means of a TC-END request primitive with prearranged end.

When the SSF has sent the last ApplyChargingReport, the dialogue may be ended from the SCF by a TC-END request primitive with basic end.

SSME-FSM related messages

The following procedures shall be followed:

- The dialogue shall be maintained when the ActivityTest Return Result is sent.
- No dialogue shall be established when the ServiceFilteringResponse operation is sent. The operation is sent with a TC-BEGIN request primitive and the dialogue is ended by means of a TC-END request primitive with prearranged end.
- A dialogue shall no longer be maintained when the Return Result of the ActivateServiceFiltering operation is sent. The dialogue is ended by means of a TC-END request primitive with basic end, the Return Result is transmitted with the same request.
- The dialogue is locally terminated by means of a TC-END request primitive with prearranged end, upon reception of a TC-BEGIN indication primitive with a CallGap operation.

19.1.2.1.1.2 SCF-to-SSF messages

This subclause defines the normal procedures for TC messages from the SCF to the SSF.

SCSM-FSM related messages

No dialogue shall be established when the SCSM-FSM moves from state **Idle** to state **Idle** upon receipt of InitialDP operation. The operation is received with a TC-BEGIN indication primitive and the dialogue is locally terminated by means of a TC-END request primitive with prearranged end.

A dialogue shall be established when the SCSM-FSM moves from state **Idle** to state **Preparing SSF Instructions** upon the receipt of InitialDP operation for TDP-R or AssistRequestInstructions operation.

A dialogue shall be established when the SCSM-FSM sends an InitiateCallAttempt or a CreateCSA from the **Idle** state.

For subsequent operations sent from the SCSM-FSM, the dialogue shall be maintained except for the following cases, i.e. all other operations are sent after a dialogue was established from the SSF (the SCF has previously received a TC-BEGIN indication primitive with an InitialDP operation or an AssistRequestInstructions operation).

The dialogue shall no longer be maintained when the prearranged end condition is met in the SCF. When the SCF does not expect any messages other than possibly REJECT or ERROR messages for the operations sent and when the last associated operation timer expires, the dialogue is locally ended by means of a TC-END request primitive with prearranged end.
Alternatively, the sending of operations , leading to the termination of the relationship, by means of a TC-END request primitive (basic end) is possible.

SCME-FSM related messages

The operations sent from the SCME-FSM shall be issued according to the following procedures:

- The dialogue shall be maintained when the ActivityTest operation is sent.
- A dialogue shall not be established when a CallGap operation is sent without using a SCSM associated dialogue. The operation is sent using a TC-BEGIN request primitive and the dialogue is terminated with a prearranged end.
- For sending one or more CallGap operations, the SCME FSM may use an existing SCSM FSM associated dialogue which was initiated by a SSF-FSM (i.e. established for the transmission of the InitialDP operation). The dialogue shall be maintained and the CallGap operation(s) shall be sent with the first response of the SCSM FSM to the InitialDP operation.
- A dialogue shall be established when an ActivateServiceFiltering operation is sent. The operation shall be transmitted with a TC-BEGIN request primitive.
- The dialogue is locally terminated upon reception of a ServiceFilteringResponse operation using a TC-END request primitive with prearranged end.

SCF-SSF - Use of dialogue handling services

Dialogue handling services are used to trigger the sending of the APDUs associated with the operations involved in the INAP packages.

Component grouping is performed under the control of the application-process through an appropriate usage of the TC-BEGIN and TC-CONTINUE service.

The TC-END service is solely used to support the dialogue closing procedure (i.e it is never used to trigger the sending of components).

On receipt of an empty TC-CONTINUE.req primitive, the FE should ignore the primitive.

On receipt of an TC-END.req with a INAP request, the FE should not perform the request and consider the requested TC-END service as a dialogue closing procedure. The dialogue is then terminated (see clause 18.1.1.1).

It is an application-process responsibility to provide in the TC-BEGIN-req primitive a destination address which can be used by the underlying SCCP to route the message to the proper FE if this FE is addressed through the SS7 network.

The pre-arranged end can be used.

19.1.2.2 Abnormal procedures

The following procedures also apply to the SCF-SRF and SCF-CUSF interfaces.

19.1.2.2.1SCF-to-SSF/SRF/CUSF messages

Considering that SSF, SRF, and CUSF do not have the logic to recover from error cases detected on the SCF-SSF/SRF/CUSF interface, the following shall apply:

- Operation errors and rejection of TCAP components shall be transmitted to the SSF and, respectively, the SRF, and the CUSF with a TC-END request primitive, basic end.

If, in violation of the above procedure, an ERROR or REJECT component is received with a TC-CONTINUE indication primitive, the SSF and, respectively, the SRF and the CUSF shall abort the dialogue with a TC-U-ABORT request primitive.

In the case of the SSF relay, it is for further study how to map messages to ROSE capability of bearer signalling system between the SSF and the SRF, and what services are assumed from ROSE.

19.1.2.2.2SSF/SRF/CUSF-to-SCF messages

Operation errors and rejection of TCAP components shall be transmitted to the SCF according to the following rules:

- The dialogue shall be maintained when the preceding message, which contained the erroneous component, indicated that the dialogue shall be maintained. I.e. the error or reject shall be transmitted with a TC-CONTINUE request primitive if the erroneous component was received with a TC-CONTINUE indication primitive.
  On receipt of an ERROR or REJECT component the SCF decides on further processing. It may either continue, explicitly end or abort the dialogue.

- In all other situations the dialogue shall no longer be maintained. I.e. the error or reject shall be transmitted with a TC-END request primitive, basic end, if the erroneous component was received with a TC-BEGIN indication primitive.

- on expiration of application timer T_{SSF} or T_{SRF} or T_{CUSF}, dialogue shall be terminated by means of by TC-U-ABORT primitive with an Abort reason, regardless of TCAP dialogue is established or not.

If the error processing in the SSF/SRF/CUSF leads to the case where the SSF/SRF/CUSF is not able to process further SCF operations while the dialogue is to be maintained, the SSF/SRF/CUSF aborts the dialogue with a TC-END request primitive with basic end or a TC-U-ABORT request primitive, depending on whether any pending ERROR or REJECT component is to be sent or not.

The SSF can end a dialogue with a TC-U-ABORT request primitive in case call release is initiated by any other entity then the SCF and the SSF has no pending call information requests (or pending requests which should be treated in the same way, i.e., ApplyCharging nor any armed EDP to notify the SCF of the call release (for alternative way, see subclause 18.1.2.1.1).

In the case of the SSF relay, it is for further study how to map messages to ROSE capability of bearer signalling system between the SSF and the SRF, and what services are assumed from ROSE.

The CUSF can end a dialogue with a TC-U-ABORT request primitive in case the association release between the user and the network is initiated by any other entity.

19.1.2.2.3SCF-SSF - Use of dialogue handling services

On receipt of a TC-U-REJECT.ind in the FE, this primitive should be ignored. It is up to the application process to abort, continue or terminate the dialogue, if not already terminated by the sending application process according to the rules as stated in clause 18.1.1.2. This is also applicable for invoke problems related to a class 4 linked operation.

A TC-U-REJECT.req should be sent followed by a TC-CONTINUE.req.

On receipt of a TC-R-REJECT.ind in the FE, this primitive should be ignored. It is up to the application process to abort, continue or terminate the dialogue, if not already terminated by the sending application process according to the rules as stated in clause 18.1.1.2. This is also applicable for invoke problems related to a class 4 linked operation.

On receipt of a TC-L-REJECT indication primitive (i.e. when a protocol error has been detected by the local TC entity) which cannot be related to an active operation, it is up to the application process to continue or to terminate the dialogue and implicitly trigger the transmission of the reject component or to abort the dialogue.

On receipt of a TC-NOTICE indication the TC-USER is informed that a message cannot be delivered by the Network Layer. It occurs if the Return Option has been set (see subclause 18.1.1.3.7). It is for the application process to decide whether to terminate the dialogue or retry.

The application-process is the sole user of the TC-P-ABORT service and TC-NOTICE service.

The receipt of a TC-U-ABORT-Ind or TC-P-ABORT-Ind on a dialogue terminates all request processing.

19.1.2.3 Dialogue handling

19.1.2.3.1 Dialogue establishment

19.1.2.3.2 Dialogue continuation

19.1.2.3.3 Dialogue termination

19.1.2.3.4 User abort

19.1.2.3.5 Provider abort

19.1.2.3.6 Mapping to TC dialogue primitives

The SSF-SCF IN services can be mapped onto TC services. This subclause defines the mapping of the SSF-SCF IN services onto the services of the TC dialogue handling services defined in ETS 300 287-1 [7].

a) The TC-BEGIN service is used to invoke the operations of the scf-ssf connection packages as defined in clause 5. .

b) The TC-CONTINUE service is used to report the success of the operations invoked in a TC-BEGIN service and to invoke or respond to any other operations.

c) The TC-U-ABORT service is used to report the failure of operations of the connection pactages as defined in clause 5.

The mapping of the parameters onto the TC-BEGIN primitive is defined in subclause 18.1.1.3.6 with the following qualifications:

- The AC Name parameter shall take the value of the application-context-name field of the **cs2ssf-scfGenericAC**, **cs2ssf-scfAssistHandoffAC** or **cs2ssf-scfServiceManagementAC** object if the initiating AE is a SSF orthe **cs2scf-ssfGenericAC**, **cs2scf-ssfTrafficManagementAC**, **cs2scf-ssfServiceManagementAC** or **cs2scf-ssfTriggerManagementAC** object if the originating AE is a SCF.

The mapping of the parameters onto the TC-CONTINUE primitive is defined in subclause 18.1.1.3.6.

The mapping of the parameters onto the TC-U-ABORT primitive is defined in subclause 18.1.1.3.6 with the following qualifications:

- The Application-Context-Name parameter shall be used as specified in ETS 300 287-1 [7]. When the responding AE refuses a dialogue because the application-context-name it receives is not supported, this parameter shall have the value of the application-context-name field of the **cs2ssf-scfGenericAC**, **cs2ssf-scfAssistHandoffAC, cs2ssf-scfServiceManagementAC** or **cs2scf-ssfGenericAC** object if the responding AE is a SCF or the, **cs2scf-ssfTrafficManagementAC**, **cs2scf-ssfServiceManagementAC** or **cs2scf-ssfTriggerManagementAC** object if the responding AE is a SSF.

The use of the parameters of the TC-END service is defined in subclause 18.1.1.3.6.

19.1.2.4 Component Handling

19.1.2.4.1 Procedures for INAP operations

The INAP ASEs are users of the TC component handling services except for the TC-L-REJECT and TC-L-CANCEL services which are used by the application-process. Receipt of a TC-L-REJECT-Ind leads the application-process to abandon the dialogue (i.e. it issues a TC-U-ABORT-Request primitive).

The TC-U-CANCEL service is never used.

19.1.2.4.2 Mapping to TC component parameters

The SSF-SCF IN ASE services are mapped onto the TC component handling services. The mapping of operations and errors onto TC services is defined in subclause 18.1.1.4.2 with the following qualifications:

The timeout parameter of the TC-INVOKE-Req primitives is set according to the clause 5.

19.1.3 SCF-SRF interface

19.1.3.1 Normal procedures

19.1.3.1.1 SCF-to/from-SRF messages

A dialogue is established when the SRF sends an AssistRequestInstructions operation to the SCF. For all other operations sent to/from the SRF, the dialogue shall be maintained.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component. When the SCSM makes a non-error case state transition to end user interaction and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components.

The dialogue shall no longer be maintained when sending the SpecialisedResourceReport operation for PlayAnnoucement with disconnection from the SRF set to true or Return Result of the PromptAndCollectUserInformation with disconnection from the SRF set to true or ScriptRun or PromptAndReceiveMessage operations with disconnection from the SRF set to true. The dialogues is ended by means of a TC-END request primitive with basic end, and the one of above operations is transmitted with the same request.

Regardless of whether pending operation exists or not, when the SRSM-FSM is informed of the disconnection of bearer connection (in the case of SCF initiated disconnection or call abandon from call party) and dialogue is established, the dialogue is ended by means of a TC-END request primitive (basic) with zero components or TC-END request primitive (prearranged end).

The dialogue shall no longer be maintained when the prearranged end condition is met in the SRF. When the SRSM-FSM is informed the disconnection of bearer connection and TCAP dialogue is not established, TCAP dialogue is locally terminated by TC-END primitive with prearranged end.

When the SCF does not expect any messages other than possibly REJECT or ERROR messages for the operations sent and when the last associated operation timer expires, the dialogue is locally ended by means of a TC-END request primitive with prearranged end. Alternatively, the sending of operations, leading to the termination of the relationship, by means of a TC-END request primitive (basic end) is possible.

In the relay case, the SRF-SCF relationship uses the SSF-SCF TCAP dialogue. This is possible, because begin and end of the SRF-SCF relationship are embedded in the SSF-SCF relationship. SRF-SCF information shall be exchanged with TC-CONTINUE request primitives.

In the case of the SSF relay, it is for further study how to map messages to ROSE capability of bearer signalling system between the SSF and the SRF, and what services are assumed from ROSE.

19.1.3.2 Abnormal procedures

19.1.3.3 Dialogue handling

19.1.3.3.1 Dialogue establishment

19.1.3.3.2 Dialogue continuation

19.1.3.3.3 Dialogue termination

19.1.3.3.4 User abort

19.1.3.3.5 Provider abort

19.1.3.3.6 Mapping to TC dialogue primitives

The SCF-SRF IN services can be mapped onto TC services. This subclause defines the mapping of the SCF-SRF IN services onto the services of the TC dialogue handling services defined in ETS 300 287-1 [7].

a) The TC-BEGIN service is used to invoke the operations of the **srf-scf** connection packages as defined in clause 6..

b) The TC-CONTINUE service is used to report the success of the operations invoked in a TC-BEGIN service and to invoke or respond to any other operations.

c) The TC-U-ABORT service is used to report the failure of operation of the **scf-srf** operations packages as defined in clause 6.

The mapping of parameters onto the TC Dialogue services is as defined in subclause 18.1.1.3.6 with the following qualifications:

The mapping of the parameters onto the TC-BEGIN primitive is defined in subclause 18.1.1.3.6 with the following qualifications:

- The AC Name parameter shall take the value of the application-context-name field of the **srf-scf-ac** object.

19.1.3.4 Component handling

19.1.3.4.1 Procedures for INAP operations

19.1.3.4.2 Mapping to TC component parameters

The mapping of parameters for the TC component services is defined in subclause 18.1.1.4.2 with the following qualifications.

The Timeout Parameter of the TC-INVOKE service is set according to clause 6.

19.1.4 SCF-CUSF interface

19.1.4.1 Normal procedures

19.1.4.1.1 CUSF-to-SCF messages

CUSF-FSM related messages

A dialogue shall be established when the CUSF-FSM moves from the state **Idle** to the state **Waiting for Instructions**. The relevant INAP operation, which is one of operations for TDP-R, shall be transmitted in the same message.

The relevant INAP operation, which is one of operations for TDP-N, shall be sent with a TC-BEGIN request primitive and the dialogue is locally ended by means of TC-END request primitive with prearranged end.

For all other operations sent from the CUSF-FSM, the dialogue shall be maintained except for the following cases.

When the CUSF sends the last event report operation, the dialogue may be ended from the CUSF by a TC-END request primitive with basic end.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component or prearranged end. When the CUSF-FSM makes a non-error case state transition to the state **Idle** and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components, or the dialogue is locally ended by means of a TC-END request primitive with prearranged end. The CUSF can end a dialogue with a TC-END request primitive with zero component or prearranged end depending on that TCAP dialogue is established or not, in the case association release between the user and the network is initiated by any other entity.

When the CUSF has sent the last event report operation the dialogue may be ended from the SCF by a TC-END request primitive with basic end.

CUSME-FSM related messages

The dialogue shall be maintained when the ActivityTest Return Result is sent.

19.1.4.1.2 SCF-to-CUSF messages

SCSM-FSM related messages

The operation is received with a TC-BEGIN indication primitive and the dialogue is locally terminated by means of a TC-END request primitive with prearranged end.

A dialogue shall be established when the SCSM-FSM moves from state **Idle** to state **Preparing CUSF Instructions** upon the receipt of one of operations for TDP-R.

For subsequent operations sent from the SCSM-FSM, the dialogue shall be maintained except for the following cases, i.e. all other operations are sent after a dialogue was established from the CUSF (the SCF has previously received a TC-BEGIN indication primitive with one of operations for TDP-R).

The dialogue shall no longer be maintained when the prearranged end condition is met in the SCF. When the SCF does not expect any messages other than possibly REJECT or ERROR messages for the operations sent and when the last associated operation timer expires, the dialogue is locally ended by means of a TC-END request primitive with prearranged end.

Alternatively, the sending of operations , leading to the termination of the relationship, by means of a TC-END request primitive (basic end) is possible.

SCME-FSM related messages

The operation(s) sent from the SCME-FSM shall be issued according to the following procedure(s):

- The dialogue shall be maintained when the ActivityTest operation is sent.

19.1.4.2 Abnormal procedures

19.1.4.3 Dialogue handling

19.1.4.3.1 Dialogue establishment

19.1.4.3.2 Dialogue continuation

19.1.4.3.3 Dialogue termination

19.1.4.3.4 User abort

19.1.4.3.5 Provider abort

19.1.4.3.6 Mapping to TC dialogue primitives

The CUSF-SCF IN services can be mapped onto TC services. This subclause defines the mapping of the CUSF-SCF IN services onto the services of the TC dialogue handling services defined in ETS 300 287-1 [7].

a) The TC-BEGIN service is used to invoke the operations of the **cusf-scf** connection packages as defined in clause 10.

b) The TC-CONTINUE service is used to report the success of the operations invoked in a TC-BEGIN service and to invoke or respond to any other operations.

c) The TC-U-ABORT service is used to report the failure of operation of the **cusf-scf** operation packages as defined in clause 10..

The mapping of parameters onto the TC Dialogue services is as defined in subclause 18.1.1.3.6 with the following qualifications:

The mapping of the parameters onto the TC-BEGIN primitive is defined in subclause 18.1.1.3.6 with the following qualifications:

- The AC Name parameter shall take the value of the application-context-name field of the **cusf-scf-ac** object if the originating AE is a CUSF or the **scf-cusf-ac** object if the originating AE is a SCF.

19.1.4.4 Component handling

19.1.4.4.1 Procedures for INAP operations

19.1.4.4.2 Mapping to TC component parameters

The mapping of parameters for the TC component services is defined in subclause 18.1.1.4.2 with the following qualifications.

The Timeout Parameter of the TC-INVOKE service is set according to clause 10.

19.1.5 SCF-SCF interface

19.1.5.1 Normal procedures

Dialogue handling services are used to trigger the sending of the APDUs associated with the operations involved in the INAP packages.

Component grouping is performed under the control of the application-process through an appropriate usage of the TC-BEGIN and TC-CONTINUE service.

The TC-END service is solely used to support the dialogue closing procedure (i.e it is never used to trigger the sending of components).

On receipt of an empty TC-CONTINUE.ind primitive, the application process should ignore the primitive.

The pre-arranged end can be used.

It is an application-process responsibility to provide in the TC-BEGIN-req primitive a destination address which can be used by the underlying SCCP to route the message to the proper FE if this FE is addressed through the SS7 network.

19.1.5.2 Abnormal procedures

On receipt of a TC-U-REJECT.ind in the FE, this primitive should be ignored. It is up to the application process to abort, continue or terminate the dialogue, if not already terminated by the sending application process according to the rules as stated in subclause 18.1.1.4.1. This is also applicable for invoke problems related to a class 4 linked operation.

On receipt of a TC-L-REJECT indication primitive (i.e. when a protocol error has been detected by the local TC entity) which cannot be related to an active operation, it is up to the application process to continue or to terminate the dialogue and implicitly trigger the transmission of the reject component or to abort the dialogue.

On receipt of a TC-NOTICE indication the TC-USER is informed that a message cannot be delivered by the Network Layer. It occurs if the Return Option has been set (see subclause 18.1.1.3.7). It is for the application process to decide whether to terminate the dialogue or retry.

The receipt of a TC-U-ABORT-Ind or TC-P-ABORT-Ind on a dialogue terminates all request processing.

19.1.5.3 Dialogue handling

19.1.5.3.1 Dialogue establishment

19.1.5.3.2 Dialogue continuation

19.1.5.3.3 Dialogue termination

19.1.5.3.4 User abort

19.1.5.3.5 Provider abort

If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3])is included in the AC, and an SE-P-ABORT is required, then the SE-P-ABORT is mapped to the TC-U-ABORT primitive.

19.1.5.3.6 Mapping to TC dialogue primitives

The SCF-SCF IN services can be mapped onto TC services. This subclause defines the mapping of the SCF-SCF IN services onto the services of the TC dialogue handling services defined in ETS 300 287-1 [7].

a) The TC-BEGIN service is used to invoke the operations of the **scf-scfConnectionPackage** and the **dsspConnectionPackage** and **SETransfer** (Defined in [46] and [3])operations.

b) The TC-CONTINUE service is used to report the success of the operations invoked in a TC-BEGIN service and to invoke or respond to any other operations. If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, the TC-CONTINUE service is used for the second and third exchanges.

c) The TC-U-ABORT service is used to report the failure of operation of the **scf-scfConnectionPackage** and **dsspConnectionPackage** and **SETransfer** (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3])operations.

d) The TC-END service is used to invoke the **SCFUnbind** operations of the **scf-scfConnectionPackage**

The mapping of the parameters onto the TC-service is as follows:

The use of the parameters of the TC-BEGIN service is as defined in subclause 18.1.1.3.6 with the following qualifications:

- The AC Name parameter of the TC-BEGIN service shall take the value of the application-context-name field of the **scf-scfOperationsAC**, **distributedSCFSystemAC or distributedSCFWith3seAC** object.

- The use of the User Information parameter of the TC-BEGIN service shall contain an EXTERNAL ASN.1 Type with the direct-reference field, if present, set to **id-as-scfBindingAS** and the value to be encoded shall be of type **SCF-SCFBinding-PDUs.bind.bind-invoke (SCFBindRequestArgument)**. If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3])is included in the AC, the User Information parameter of the TC-BEGIN service shall contain a value of type **seItem**.

The use of the parameters of the TC-CONTINUE service is as defined in subclause 18.1.1.3.6 with the following qualifications:

- The use of the User Information parameter of the first TC-CONTINUE service shall contain an EXTERNAL ASN.1 Type with the direct-reference field, if present, set to **id-as-scfBindingAS** and the value to be encoded shall be of type **SCF-SCFBinding-PDUs.bind.bind-result (SCFBindResultArgument)**. If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, the User Information parameter of the TC-CONTINUE service shall contain a value of type **seItem**.

  In subsequent TC-CONTINUE services the use of the User Information field is network operator specific.

The use of the parameters of the TC-U-ABORT service is as defined in subclause 18.1.1.3.6 with the following qualifications:

- The Application-Context-Name parameter of the TC-U-ABORT service shall be used as specified in ETS 300 287-1 [7]. When the SCF refuses a dialogue because the application-context-name it receives is not supported, the value of the application-context-name it returns is selected as follows:
  - if the value has the same root as the **scf-scfOperationsAC**, the **scf-scfOperationsAC** is used,
  - if the value has the same root as the **distributedSCFSystemAC**, the **distributedSCFSystemAC** is used,
  - if the value has the same root as the **distributedSCFWith3seAC**, the **distributedSCFWith3seAC** is used,
  - otherwise the received value is returned.
- The User Information parameter of the TC-U-ABORT service shall contain an EXTERNAL ASN.1 Type with the direct-reference field, if present, set to **id-as-scfBindingAS** and the value to be encoded shall be of type **SCF-SCFBinding-PDUs.bind.bind-invoke (SCFBindErrorArgument)**.

The **DirectoryUnbind** service is mapped onto the TC-END service.

The use of the parameters of the TC-END service is as defined in subclause 18.1.1.3.6.

19.1.5.4 Component handling

19.1.5.4.1 Procedures for INAP operations

The INAP ASEs are users of the TC component handling services except for the TC-L-REJECT and TC-L-CANCEL services which are used by the application-process. Receipt of a TC-L-REJECT-Ind leads the application-process to abandon the dialogue (i.e. it issues a TC-U-ABORT-Request primitive).

The TC-U-CANCEL service is never used.

19.1.5.4.2 Mapping to TC component parameters

The SCF-SCF IN ASE services are mapped onto the TC component handling services. The mapping of operations and errors onto TC services is defined in subclause 18.1.1.4.2 with the following qualifications:

The timeout parameter of the TC-INVOKE-Req primitives is set according to clause 9. The SCFBind timer (T_{bi}) does not map to a TC timer.

19.1.6 SCF-SDF interface

All the services provided by the Directory ASEs are contained in a single AE. The Component Handler (CHA) of the TC supports the request/reply paradigm of the operation. The Directory ASEs provide the mapping function of the abstract-syntax notation of the directory operation packages onto the services provided by TC. The Dialogue handler (DHA) of the TC supports the establishment and release of an application-association called "dialogue" between a pair of AEs. Dialogues between a DUA and a DSA may be established only by the DUA.

19.1.6.1 Normal procedures

Dialogue handling services are used to support the DirectoryBind and DirectoryUnbind operations and to trigger the sending of the APDUs associated with the operations involved in the Directory packages.

Component grouping is performed under the control of the application-process through an appropriate usage of the TC-BEGIN and TC-CONTINUE service.

The TC-END service is solely used to support the unbind procedure (i.e it is never used to trigger the sending of components).

On receipt of an empty TC-CONTINUE.req primitive, the SDF should ignore the primitive.

On receipt of an TC-END.req with a database request, the SDF should not perform the database request and consider the requested TC-END service as an unbind procedure. The dialogue is then terminated .

19.1.6.2 Abnormal procedures

On receipt of a TC-U-REJECT.ind in the SDF, this primitive should be ignored.

On receipt of a TC-R-REJECT.ind in the SDF, the dialogue should be release with a TC-U-ABORT.req.

If reject situations are detected in the SDF, a TC-U-REJECT.req should be sent followed by a TC-CONTINUE.req.

The pre-arranged termination procedure is never used.

The application-process is the sole user of the TC-P-ABORT service and TC-NOTICE service.

The receipt of a TC-U-ABORT-Ind or TC-P-ABORT-Ind on a dialogue terminates all request processing. It is the application-process responsibility to confirm if requested modifications occurred.

It is an application-process responsibility to provide in the TC-BEGIN-req primitive a destination address which can be used by the underlying SCCP to route the message to the proper SDF.

19.1.6.3 Dialogue handling

19.1.6.3.1 Dialogue establishment

19.1.6.3.2 Dialogue continuation

19.1.6.3.3 Dialogue termination

19.1.6.3.4 User abort

If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, and an SE-U-ABORT is required, then the SE-U-ABORT is mapped to the TC-U-ABORT primitive.

19.1.6.3.5 Provider abort

If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, and an SE-P-ABORT is required, then the SE-P-ABORT is mapped to the TC-U-ABORT primitive.

19.1.6.3.6 Mapping to TC dialogue primitives

This subclause defines the mapping of the DirectoryBind and DirectoryUnbind services onto the services of the TC dialogue handling services defined in ETS 300 287-1 [7].

Bind

The DirectoryBind service is mapped onto TC-services as follows:

a) The TC-BEGIN service is used to invoke the **DirectoryBind** operation and **SETransfer** (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) operations.

b) The TC-CONTINUE service is used to report the success of the **DirectoryBind** operation. If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, the TC-CONTINUE service is used for the second and third exchanges.

c) The TC-U-ABORT service is used to report the failure of the **DirectoryBind** operation and **SETransfer** (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) operations.

The use of the parameter of these services is qualified in the following subclauses.

The use of parameters of the TC-BEGIN service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The AC Name parameter of the TC-BEGIN service shall take the value of the application-context-name field of the **iNdirectoryAccessAC** or **inExtendedDirectoryAccessAC** object.
- The Dialogue ID parameter of the TC-CONTINUE service shall be used as specified in subclause 18.1.1.3.7.

  The Authorized Relationship Id can be mapped onto the TC Dialogue ID parameter.
- The User Information parameter of the TC-BEGIN service shall contain an EXTERNAL ASN.1 Type with the direct-reference field, if present, set to **id-as-indirectoryBindingAS** and the value to be encoded shall be of type **DAPBinding-PDUs.bind.bind-invoke (DirectoryBindArgument).**

  If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, the User Information parameter of the TC-BEGIN service shall contain a value of type **seItem**.

The use of parameters of the TC-CONTINUE service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The Dialogue ID parameter of the TC-CONTINUE service shall be used as specified in subclause 18.1.1.3.7.

  The Authorized Relationship Id can be mapped onto the TC Dialogue ID parameter.
- The User Information parameter of the TC-CONTINUE service shall contain an EXTERNAL ASN.1 Type with the direct-reference field, if present, set to **id-as-indirectoryBindingAS** and the value to be encoded shall be of type **DAPBinding-PDUs.bind.bind-result (DirectoryBindResult)**.

  If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, the User Information parameter of the TC-CONTINUE service shall contain a value of type **seItem**.

The use of parameters of the TC-U-ABORT service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The Dialogue ID parameter of the TC-CONTINUE service shall be used as specified in subclause 18.1.1.3.7.

  The Authorized Relationship Id can be mapped onto the TC Dialogue ID parameter.
- The AC Name parameter of the TC-U-ABORT service shall be set as defined in subclause 18.1.1.3.6 unless the TC-U-ABORT is reporting the failure of the directoryBind operation. In this case it is set to the value of the application-context-name field of the **iNdirectoryAccessAC** or **inExtendedDirectoryAccessAC** object
- The User Information parameter of the TC-U-ABORT service shall be used as specified in subclause 18.1.1.3.7, unless the TC-U-ABORT is reporting the failure of a directoryBind operation. In this case this parameter shall contain an EXTERNAL ASN.1 Type with the direct-reference field, if present, set to **id-as-indirectoryBindingAS** and the value to be encoded shall be of type **DAPBinding-PDUs.bind.bind-error (DirectoryBindError)**.

Unbind

The **DirectoryUnbind** service is mapped onto the TC-END service.

The use of the parameters of the TC-END service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The Dialogue ID parameter of the TC-CONTINUE service shall be used as specified in subclause 18.1.1.3.7.
- The Authorized Relationship Id can be mapped onto the TC Dialogue ID parameter.

19.1.6.4 Component handling

19.1.6.4.1 Procedures for INAP Operations

The Directory ASEs are users of the TC component handling services except for the TC-L-REJECT and TC-L-CANCEL services which are used by the application-process. Receipt of a TC-L-REJECT-Ind leads the application-process to abandon the dialogue (i.e. it issues a TC-U-ABORT-Request primitive).

The TC-U-CANCEL service is never used.

19.1.6.4.2 Mapping to TC component parameters

The Directory ASE services are mapped onto the TC component handling services. The mapping of operations and errors onto TC services is defined in subclause 18.1.1.4.2 with the following qualifications: .

The timeout parameter of the TC-INVOKE-Req primitives is set according to table 18-2.

Table 18-2: TC timer values of DAP operations

Operation	Timeout
search	medium
modifyEntry	medium
addEntry	medium
removeEntry	medium
execute	medium

19.1.7 SDF-SDF interface

19.1.7.1 Normal procedures

19.1.7.2 Abnormal procedures

19.1.7.3 Dialogue handling

19.1.7.3.1 Dialogue establishment

19.1.7.3.2 Dialogue continuation

19.1.7.3.3 Dialogue termination

19.1.7.3.4 User abort

If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, and an SE-U-ABORT is required, then the SE-U-ABORT is mapped to the TC-U-ABORT primitive.

19.1.7.3.5 Provider abort

If the SESE (defined in ITU-T Recomendation X.832 [46] and ETS 300 009-1 [3]) is included in the AC, and an SE-P-ABORT is required, then the SE-P-ABORT is mapped to the TC-U-ABORT primitive.

19.1.7.3.6 Mapping to TC dialogue primitives

The DSP and DISP can be mapped onto TC services. This subclause defines the mapping of the **DSABind, DSAUnbind, DSAShadowBind** and **DSAShadowUnbind** , services onto the services of the TC dialogue handling services defined in ETS 300 287-1 [7].

The **DirectoryBind** service in mapped onto TC-services as follows:

a) The TC-BEGIN service is used to invoke the **DSAShadowBind** and **DSABind** operations.

b) The TC-CONTINUE service is used to report the success of the **DSAShadowBind** and **DSABind** operations.

c) The TC-U-ABORT service is used to report the failure of the **DSAShadowBind** and **DSABind** operations.

d) The TC-END service is used to invoke the **DSAShadowUnbind** operations of the **dispConnectionPackage**

The mapping of parameters onto the TC Dialogue services is as follows:

The use of parameters of the TC-BEGIN service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The AC Name parameter of the TC-BEGIN service shall take the value of the application-context-name field of the **inDirectorySystemAC** or **shadowSupplierInitiatedAC** or **shadowConsumerInitiatedAC** object.

- The User Information parameter of the TC-BEGIN service shall contain a value of type EXTERNAL which depends on the value of the AC being used.

  If the AC being used is **inDirectorySystemAC** then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to **id-as-indirectoryDSABindingAS** and the value to be encoded shall be of type **DSABinding-PDUs.bind.bind-invoke.**

  If the AC being used is **shadowSupplierInitiatedAC** or **shadowConsumerInitiatedAC** then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to **id-as-indsaShadowBindingAS** and the value to be encoded shall be of type **DISBinding-PDUs.bind.bind-invoke.**

The use of parameters of the TC-CONTINUE service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The Dialogue Id parameter of the TC-CONTINUE service shall be used as specified in ETS 300 287-1 [7]. The Authorized Relationship Id can be mapped onto the TCAP Dialogue Id.

- The User Information parameter of the first TC-CONTINUE service shall contain a value of type EXTERNAL which depends on the value of the AC being used.

  If the AC being used is **inDirectorySystemAC** then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to **id-as-indirectoryDSABindingAS** and the value to be encoded shall be of type **DSABinding-PDUs.bind.bind-result.**

  If the AC being used is **shadowSupplierInitiatedAC** or **shadowConsumerInitiatedAC** then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to **id-as-indsaShadowBindingAS** and the value to be encoded shall be of type **DISPBinding-PDUs.bind.bind-result.**

The use of parameters of the TC-U-ABORT service is defined in subclause 18.1.1.3.6 with the following qualifications:

- The Dialogue Id parameter of the TC-U-ABORT service shall be used as specified in subclause 18.1.1.3.6.. The Authorized Relationship Id can be mapped onto the TCAP Dialogue Id.

- The Application-Context-Name parameter of the TC-U-ABORT service shall be used as specified in ETS 300 287-1 [7]. When the SDF refuses a dialogue because the application-context-name it receives is not supported, this parameter shall have the value of the application-context-name field of the **inDirectorySystemAC** or **shadowSupplierInitiatedAC** or **shadowConsumerInitiatedAC** object.

- When the TC-U-ABORT is reporting the failure of a bind operation, ie. the abort reason parameter has the value "dialogue-refused", the User Information parameter of the TC-U-ABORT service shall contain a value of type EXTERNAL which depends on the value of the AC being used.

  If the AC being used is **inDirectorySystemAC** then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to **id-as-indirectoryDSABindingAS** and the value to be encoded shall be of type **DSABinding-PDUs.bind.bind-error.**

  If the AC being used is **shadowSupplierInitiatedAC** or **shadowConsumerInitiatedAC** then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to **id-as-indsaShadowBindingAS** and the value to be encoded shall be of type **DISPBinding-PDUs.bind.bind-error.**.

  Otherwise it shall be absent.

The **inDSAUnbind** and **inDSAShadowUnbind** services are mapped onto the TC-END service. The use of the parameters of the TC-END service is defined in subclause 18.1.1.3.6.

The **SETransfer** service is mapped onto TC-services is the same as for the other services except:

- If the SESE is included in the AC, the User Information parameter of the TC-CONTINUE service shall contain a value of type **seItem**.

19.1.7.4 Component handling

19.1.7.4.1 Procedures for INAP operations

19.1.7.4.2 Mapping to TC Component Parameters

The Directory ASE services are mapped onto the TC component handling services. The mapping of operations and errors onto TC services is defined in subclause 18.1.1.4.2 with the following qualifications.

The timeout parameter of the TC-INVOKE-Req primitives is set according to table 18-3.

Table 18-3: TC timer values of DSP/DISP operations

Operation	Timeout
chainedAddEntry	medium
chainedRemoveEntry	medium
chainedModifyEntry	medium
chainedExecute	medium
chainedSearch	medium
updateShadow	medium
coordinateShadowUpdate	medium
requestShadowUpdate	medium

19.2 Services assumed from SCCP

This clause describes the services required from the SCCP that may be used by the IN applications for the IN Application Protocol used between the SSF, SCF and SRF, SDF and CUSF.

19.2.1 Normal procedures

The SCCP forms the link between the TC and the MTP and provides (in conjunction with the MTP) the network services for the IN applications. The network services provided allow the signalling messages sent by the application to the lower layers to be successfully delivered to the peer application.

19.2.2 Service functions from SCCP

19.2.2.1 SCCP connectionless cervices

The services described are those given in the SCCP [2] and ETS 300 009-1 (SCCP User Guide) [3] should be consulted to identify possible interworking and compatibility issues between the different SCCP versions.

The following Connection-less services are expected from the SCCP:

(a) Network Addressing to enable signalling connections between SCCP users,

(b) Sequence Control to enable the SCCP users to invoke "sequence guaranteed" or "sequence not guaranteed" options for a given stream of messages to the same destination,

(c) Segmentation/reassembly of large user messages,

(d) Return Option to enable the SCCP users to invoke "discard message on error" or "return message on error" for a given message not able to be delivered by the SCCP to the destination SCCP user, due to routeing or segmentation/re-assembly failure.

(e) Congestion control

The primitives used for the above services are given below.

The N-UNITDATA request and N-UNITDATA indication primitives are used to send and receive data. The parameters of these primitives include the Called and Calling Addresses, Sequence Control, Return Option and User Data with the addressing parameters always mandatory.

The N-NOTICE indication primitive is used to return undelivered data if return option is set and a routeing/segmentation error occurs.

19.2.2.1.1 Addressing

The addressing elements consist of information contained within the Calling and the Called Party Addresses which are sent by the application to the lower layers.

The application expects the SCCP to route messages by either (a) the use of the Destination Point Code (DPC) plus the Subsystem Number (SSN), or (b) the use of the GT plus optionally the SSN. The application also specifies to the lower layer whether to route the message on the DPC or the GT.

Method (a) above may be used when the application is aware of the destination point code and the destination SSN located at that point code to which the message is to be delivered. Within a national network different SSNs, according to ETS 300 009-1 [3], may be allocated for the different network specific applications, e.g. a SSN may be allocated for a SCF functionality.

Method (b) above may be used when a message is to be delivered to a SCCP-user which can be identified by the combination of the elements within the GT. An example of the use of this method is when messages have to be delivered between different networks. This method may be used since the originating network is unaware of the point code and SSN's allocations within the destination network. The network that determines the end-node to which the message is to be delivered has to perform a GT Translation to derive the destination Point Code and the SSN. If optionally the original address contained the SSN, then this may be used as the destination SSN, or the translation may, if required, provide an appropriate new SSN. Where the destination node is in another network (and is not the gateway node) then the application populates the SSN field with either the SSN in use at the destination or zero.

When GT is used for addressing, the IN application expects that the SCCP supports the following elements as defined in ETS 300 009-1 [3]:

Address Indicator:

The application will set this indicator to indicate one or any combination of the elements "signalling point code, GT, subsystem number" in the address information octets.

GT Indicator:

This indicator specifies the method employed for the formatting of the address information. There are four values (1 to 4), for example, the value 4 indicates that the format includes the numbering plan, the nature of the address indicator and the translation type. The format with the indicator value 4 is always used for internetwork connections.

Translation Type:

The Translation Types are defined within ETS 300 009-1 [3].

Numbering Plan:

(1) The *proposed* "generic numbering plan" is described within the SCCP ETS 300 009-1 [3]. This numbering plan identifies the SCCP nodes or SCCP subsystems unambiguously such that messages may be efficiently routed within one or more networks, and is particularly useful when used in the Calling Address for the sending of a response message back to the originating node. This is achieved by having an international and a national part in the generic numbering plan. For response messages the responding node analyses the international part of the generic numbering plan to determine the gateway node to which the response is to be routed. Having routed to the gateway node, the national part (which was populated within the originating network) is analysed to determine the originating node within the originating network.

(2) A numbering plan which would define particular nodes based specifically on services is *outside the scope of IN CS2.*

(3) The SCCP caters for a number of other numbering plans (e.g. ISDN, Data, Telex, Mobile etc. numbering plans). The whole range catered for is shown in [2]. These may be used by IN applications if deemed suitable.

Encoding Scheme:

This identifies the encoding scheme employed by the application and is generally BCD encoded with odd or even number of digits.

GT Address Information:

These are the actual address digits supplied by the application and may be BCD digits or encoded as indicated by the encoding scheme.

The network provider must ensure that any change of GT value during translation preserves any INAP specific information contained in the initial GT value.

This requirement applies to all interfaces, not just those used for internetworking.

If *route on SSN* is to be supported from the originating node then a non-zero internationally standardized SSN is required for international internetworking.

In the absence of a standardized non-zero SSN for INAP services, the use of *route on GT* is mandatory from the origin node to the network containing the destination node.

The version of SCCP used to support INAP operations must be at least White Book 1992.

19.2.2.1.2 Sequence control

The application will specify whether SCCP protocol class 0 or 1 is required. Class 0 provides a basic connection-less service where the sequence of message delivery is not guaranteed. Class 1 connection-less service provides a guaranteed sequence delivery of messages (with the same called address) for a given stream of messages.

19.2.2.1.3 Return on error

Return on Error mechanism may be required by the IN applications such that the application is aware of messages that have not been delivered to the destination by the SCCP. The return option allows the return of the message that was not delivered due to routeing or segmentation/re-assembly failure back to the issuing user. This return option may be required in all segments of a long message or only in the first segment by the IN applications.

If the return option is invoked by the application and the message is not delivered then the SCCP specifies the "return reason" as specified in [3]. The N-NOTICE primitive is used to return the undelivered message to the originating user.

20.2.2.1.4 Segmentation/reassembly

The application expects that since the SCCP can send up to 260 octets of user data (including the address information and TC-message) in a UDT message (248 octets in a XUDT message performing segmentation and congestion control), segmentation is available for long user messages.

Also the SCCP is expected to perform the reassembly function on received segmented messages and deliver the reassembled message to the user.

However, it should be noted that even though the theoretical maximum size of SCCP-user data and addresses that can be segmented by the SCCP is 3 968 octets, the SCCP-user would limit the length to about 2 560 octets to allow for the largest known addresses. Note that the application must also allow for the octets used for the TC-message in the 2 560 octets.

The IN application does not expect the SCCP to segment the long message into more than 16 segments.

19.2.2.1.5 Congestion control

To help control of possible congestion that might occur in the lower layers the application may assign a value to indicate the importance of the message. The use of this parameter requires the use of SCCP (1997) ITU-T Recommendations.

Also there exist other congestion control mechanisms as indicated below in SCCP Management.

19.2.2.2 SCCP connection oriented services

The use by IN applications for the Connection-oriented services is outside the scope of IN CS2..

19.2.2.3 SCCP management

The subsystems used within the IN scenario expect the SCCP to provide management procedures to maintain network performance by re-routeing in the event of failure of a subsystem, and in case of network congestion by use of the congestion handling procedure. These procedures have appropriate interactions with the SCCP user as described in ETS 300 009-1 [3].

To achieve the above the SCCP is expected to perform the following procedures:

- Signalling point status management (which include the signalling point prohibited, signalling point allowed, signalling point congested, and local MTP availability sub procedures).
- Subsystem status management (which include the subsystem prohibited, subsystem allowed, and subsystem status test sub procedures).
- Co-ordinated state change (a procedure which allows a duplicated subsystem to be withdrawn from service without affecting the performance of the network).

20 IN generic interface security

Any interface within the IN functional architecture may have the need to apply security functions to the information flows passing across the interface. This section defines a generic set of security mechanisms and procedures based on the ITU-T Recommendation X.509 [38] Authentication Framework and the Simple Public Key GSS API Mechanism (SPKM) to enable any interfaces to provide suitable secured communications. This section defines a sub-profile of ITU-T Recommendation X.509 [38] and SPKM for use of IN CS2.

For IN CS2 the provision for security functions is required for the SCF-SDF, SDF-SDF, and SCF-SCF interfaces.

20.1 Interface security requirements

The security requirements on an IN system could be divided in the two following main families:

Requirements to offer security features (**Network access security requirements)**: this covers the various aspects relating to the protection of user access to services and of terminals to networks against attacks at the access interface (for instance user impersonation), by means of security features such as: user/terminal authentication (i.e. the result of a process by which a service user proves his or her identity to an IN system), user profile verification (i.e. the verification that a user is authorized to use a functionality), etc.

Requirements on the internetworking interfaces (Internetworking security requirements): this covers the protection of interactions between the various entities and agents involved in the provision of a telecommunication service against attacks at the internal interfaces of the system or even against data stored inside equipment, by means of security features such as: peer entity authentication (i.e. a process which allows a communicating entity to prove its identity to another entity in the network), signalling data or TMN data integrity, non repudiation, confidentiality, entity profile verification (i.e. the verification that an entity is authorized to use a functionality), etc.

For each of the above problems, the interfaces have to provide processes to enable data confidentiality, data origin authentication and data integrity. These capabilities are generally described in ITU-T Recommendation X.500 [36].

20.1.1 Data confidentiality

Data Confidentiality ensures that only the parties involved can examine the contents of the message. Encryption, which applies an algorithm to transform the message into an protected form, is sufficient to provide data confidentiality. The encryption methods for all communications with the directory shall be conformant to Directory Security Mechanisms or to directory entry methods.

20.1.2 Data Integrity and Data Origin authentication

Data Integrity ensures that the contents of the message have not been altered during transmission. Data Origin authentication ensures that the message came from the expected origin. Digital Signatures, which applies an encryption algorithm to the result of applying a hash function to the contents of message, are sufficient for providing data integrity and data origin authentication. Data integrity methods for all communications with the directory shall be conformant to Directory Security Mechanisms or to directory entry methods.

20.1.3 Key Management

Key Management refers to the generation, distribution and maintenance of private keys used in the cryptographic system. Key Management methods conformant with ITU-T Recommendation X.509 [38] shall be used.

20.2 Procedures and algorithms

Several assumptions are provided to the IN CS2 interface security procedures to focus the work, but which also provide acceptable levels of security:

- Generic Security Interface shall be based on SPKM as much as possible.
- The number of parameters in messages and procedures should be minimized, because each of them adds to the signaling traffic load and to some processing time.
- Only one type of security shall be applied to the interface. The order of selection will be none, User, Peer:

Table 19-1: Categories of Security Levels

	Signature	Encryption
None	None	None
User	None	User
	User	None
	User	User
Peer	None	Peer
	Peer	None
	Peer	Peer

- If both user and peer signature/encryption is requested, then user signature/encryption should be used.
- Only the mechanisms Encryption and Digital Signatures (including Message Authentication Code (MAC) techniques) shall be used.

20.2.1 Authentication procedures

The Authentication Procedure defines the process of establishing authentication and shall perform the following functions:

- Authenticate the Entity (User/FE) as being allowed to access an FE.
- Establish the level of security for all messages between the Entity and the FE.
- If required, establish a session key for encryption of messages between the Entity and the FE.

The current IN CS2 authentication uses possible procedures defined in ITU-T Recommendation X.509 [38], together with a new procedure, SPKM. The primary purpose of the ITU-T Recommendation X.509 [38] is to provide authentication for a directory service, which includes mechanisms on Simple Authentication, Strong Authentication and SPKM. This section indicates which aspects of the Directory Authentication should be considered and supported by implementors within the scope of IN CS2.

The current CS2 authentication procedures are:

1. Simple Authentication (ITU-T Recommendation X.509 [38] subclause 6.2), using Bind operations with passwords for IN CS1 (1995) or IN CS2.
2. Strong Authentication (ITU-T Recommendation X.509 [38] subclause 10), using Bind operations with strong credentials for IN CS2 and 1993 Directory; or using SPKM for IN CS2.
3. External Procedures, using Bind operations for IN CS1 (1995) or IN CS2.
4. SPKM:
   - The SPKM allows both unilateral and mutual authentication to be accomplished without the use of secure timestamps. This enables environments which do not have access to secure time to nevertheless have access to secure authentication.
   - The SPKM uses Algorithm IDs to specify various algorithms (for data confidentiality, data integrity, etc.) to be used by the communicating peers. This allows maximum flexibility for a variety of environments, for future enhancements, and for alternative algorithms.

20.2.2 SPKM algorithms and negotiation

Additional information on SPKM is provided in this section. For additional information on other authentication mechanisms, including Directory entry methods, please refer to the user guide and relevant Directory specifications.

A number of algorithm types are employed in SPKM. Examples of algorithms are presented in Appendix III. It should be noted that in the SPKM algorithms, the terms "initiator" and "target" are used. For IN CS2, the term "target" may be referred to as "consumer".

20.2.2.1 Data Confidentiality Algorithm (C-ALG):

The symmetric algorithm is used to generate the encrypted data. The DES-Connectionless Bearer Control (CBC) algorithm is an example for data confidentiality.

20.2.2.2 Data Integrity Algorithm (I-ALG)

The purpose of this algorithm is to ensure that a message has not been altered in any way after being constructed by the legitimate sender. Depending on the algorithm used, the application of this algorithm may also provide authenticity and support non-repudiation (including data origin authentication) for the message. Algorithms such as md5WithRSA, DES-MAC, md5-DES-CBC, and sum64-DES-CBC are examples for data integrity.

20.2.2.3 Key management ALGorithm (K-ALG)

The purpose of this algorithm is to establish a symmetric key (called context key) for use by both the initiator and the responder (called target) over the established context. The keys used for C-ALG and any keyed I-ALGs (for example, DES-MAC) are derived from this context key. Key establishment is done within the ITU-T Recommendation X.509 [38] authentication exchange and so the resulting shared symmetric key is authenticated. Algorithms such as RSAEncryption and dhKeyAgreement are examples for Key Management.

20.2.2.4 One-Way Function (O-ALG) for Subkey Derivation Algorithm

Having established a context key using the negotiated K-ALG, both initiator and target must be able to derive a set of subkeys for the various C-ALGs and keyed I-ALGs.

20.2.2.5 SPKM Negotiation

During context establishment in SPKM, the initiator offers a set of possible confidentiality algorithms and a set of possible integrity algorithms to the target (note that the term "integrity algorithms" includes digital signature algorithms). The confidentiality algorithms selected by the target become ones that may be used for C-ALG over the established context, and the integrity algorithms selected by the target become ones that may be used for I-ALG over the established context (the target "selects" algorithms by returning, in the same relative order, the subset of each offered list that it supports). Note that any C-ALG and I-ALG may be used for any message over the context and that the first confidentiality algorithm and the first integrity algorithm in the agreed sets become the default algorithms for that context.

The agreed confidentiality and integrity algorithms for a specific context define the valid values of the Quality of Protection (QOP) parameter. If no response is expected from the target in unilateral authentication, then the algorithms offered by the initiator are the ones that may be used over the context (if this is unacceptable to the target, then the association with this context will not be successfully established).

Furthermore, in the first context establishment token the initiator offers a set of possible K-ALGs, along with the key (or key half) corresponding to the first algorithm in the set (its preferred algorithm). If this K-ALG is unacceptable to the target then the target must choose one of the other K-ALGs in the set and send this choice along with the key (or key half) corresponding to this choice in its response (otherwise the association with this context will not be established). If necessary (that is, if the target chooses a 2-pass K-ALG such as dhKeyAgreement), the initiator will send its key half in a response to the target.

Finally, in the first context establishment token the initiator offers a set of possible O-ALGs (subkey derivation algorithms) (only a single O-ALG if no response is expected). The (single) O-ALG chosen by the target becomes the subkey derivation algorithm to be used over the context.

20.2.3 Three-way mutual authentication

The use of three-way mutual authentication procedures (including SPKM) in the establishment of a security association between two FEs implies the use of the Generic Upper Layers Security (GULS) Security Exchange Service Element (SESE) protocol (defined in ITU-T Recommendations X.831 [45] and X.832 [46]) as described in ITU-T Recommendation X.519 [41].

20.2.4. Assignment of credentials

Procedures for assigning credentials to users or FEs are generally defined in ITU-T Recommendation X.500 [36].

20.3. Mapping of Security Information Flow Definitions to Tokens

In the case of the SCF - SDF interface, the Context Establishment Tokens will be conveyed in Bind procedure and the Per-Message Tokens will be used in DAP operations.

In the case of the SDF-SDF interface, the Context Establishment Tokens will be conveyed in Bind procedure and the Per-Message Tokens will be used in DSP and DISP operations.

In the case of the SCF-SCF interface, the Context Establishment Tokens will be conveyed in the SCF Bind procedure and the Per-Message Tokens will be used in SCF-SCF operations and Distributed SCF System operations.

20.4 Security FSM definitions

The Security FSMs described here are generic machines. They illustrate how the security procedures are handled across the interface between two FEs.

20.4.1 Two-way mutual authentication FSMs

20.4.1.1 Outgoing FSM

The outgoing FSM controls the originating side of the interface. It is shown in figure 19-1.

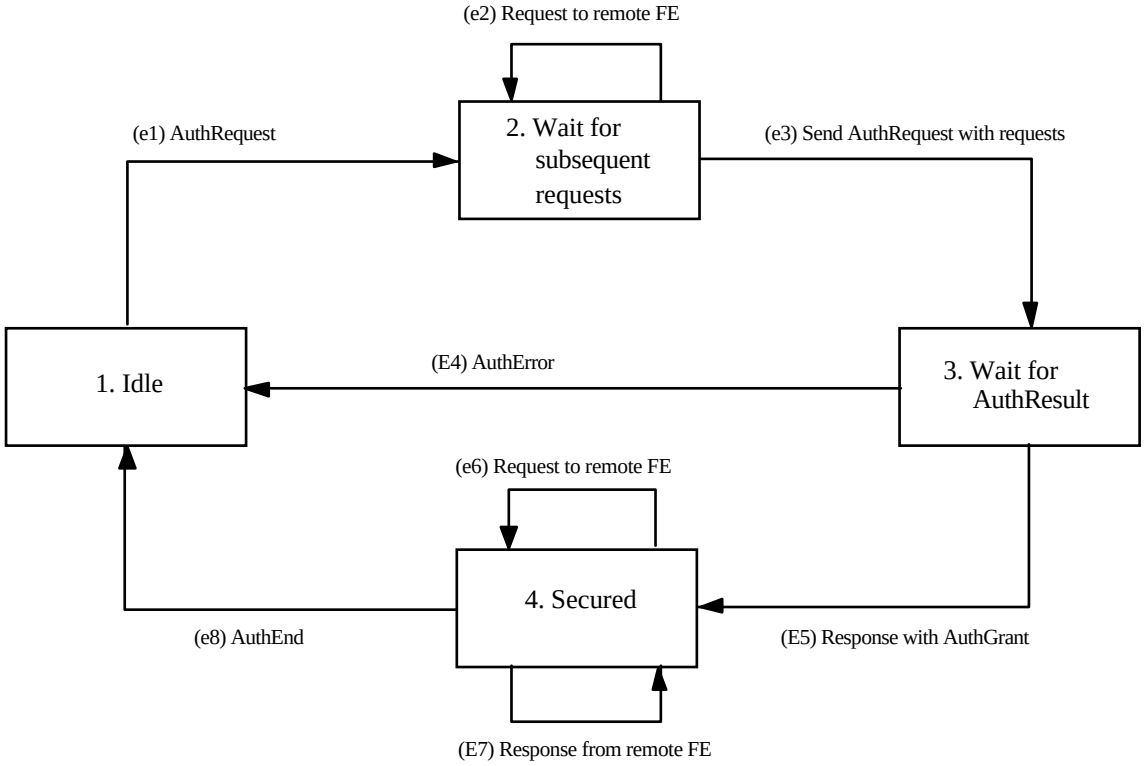


Figure 19-1: Originating FSM states

20.4.1.1.1 State 1: "Idle"

The following event is considered in this state:

(e1)AuthRequest
This is an internal event, caused by the need of the SL to create an association with an Remote FE in order to begin accessing a service function. This event causes a transition to the State 2, **Wait for subsequent requests**.

20.4.1.1.2 State 2: "Wait for subsequent requests"

In this state, subsequent operations to be sent with the **AuthRequest** operation (in the same message) to the remote FE are expected. The following two events are considered in this state:

(e2)Request _to_remote_FE
This is an internal event caused by the reception of an operation. The operation is buffered until the reception of a delimiter (or a timer expiration). The Originating FSM remains in the same state; and

(e3)Send_AuthRequest_with_requests
This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing the argument of the **AuthRequest** operation and other operations" arguments, if any, is sent to the remote FE. This event causes a transition out of this state to the State 3, **Wait for Auth Result**.

20.4.1.1.3 State 3: "Wait for Auth Result"

In this state, the local FE is waiting for the response from the remote FE. Two events are considered in this state:

(E4)Auth_Error
This is an external event, caused by the reception of a error to the **AuthRequest** operation previously issued to the remote FE. This event causes a transition out of this state to State 1, **Idle**; and

(E5)Response_with_AuthGrant
This is an external event, caused by the reception of a **AuthRequest** result combined with the responses to other operations previously issued to the remote FE (if any). This event causes a transition to State 4, **Secured.**

20.4.1.1.4 State 4: "Secured"

In this state, the local FE has established an authenticated access to the remote FE, and is waiting for requests to the remote FE from the SL or is waiting for responses to the operations previously issued to the remote FE. Three events are considered in this state:

(e6)Request_to_remote_FE
This is an internal event, caused by the need of the SL to access the remote FE. The Originating FSM remains in the same state;

(E7)Response_from_remote_FE
This is an external event, caused by the reception of responses to the operations previously issued to the remote FE. The Originating FSM remains in the same state; and

(e8)AuthEnd
This is an internal event, caused by the need of the SL to terminate the authenticated access to the Remote FE. This event causes a transition state to State 1, **Idle**.

20.4.1.2 Incoming FSM

The incoming FSM controls the terminating side of the interface. It is shown in figure 19-2.

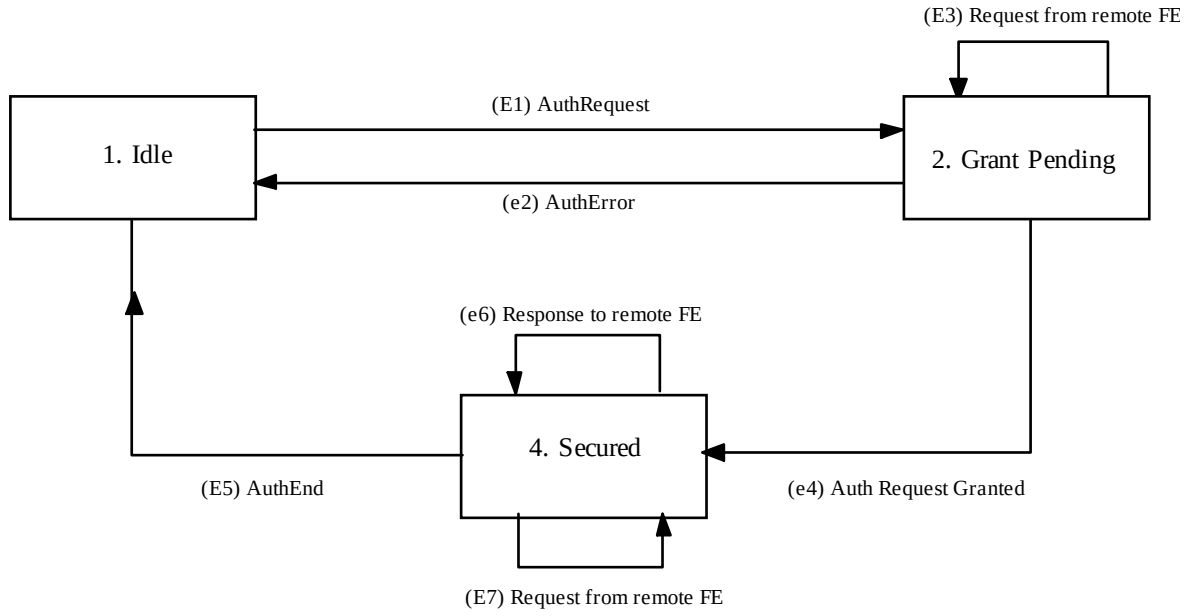


Figure 19-2: Terminating FSM states

Each state is discussed in one of the following subclauses.

General rules applicable to more than one state are as follows:

In any state, if the dialogue with the Remote FE is terminated, then the Terminating FSM returns to **Idle** state after ensuring that any resources allocated to the call have been de-allocated.

20.4.1.2.1 State 1: "Idle"

The only event accepted in this state is:

(E1)AuthRequest
This is an external event caused by the reception of AuthRequest operation from the remote FE. This event causes a transition out of this state to State 2, **Grant Pending**.

20.4.1.2.2 State 2: "Grant Pending"

In this state, an Auth Request has been received from the remote FE. The local FE is performing the authentication procedures. Three events are considered in this state:

(e2)AuthError
This is an internal event, caused by the failure of the AuthRequest operation previously issued to the local FE. This event causes a transition out of this state to State 1, **Idle** and a AuthError is returned to the invoking SCF;

(E3)Request_from_remote_FE
This is an external event, caused by the reception of operations before the result from the AuthRequest operation is determined. The SDSM remains in the same state; and

(e4)AuthRequest_Granted
This is an internal event, caused by the successful completion of the AuthRequest operation previously issued to the local FE. This event causes a transition out of this state to State 3, **Secured**.

20.4.1.2.3 State 3: "Secured"

In this state, the access of the remote FE to the local FE was authorized and operations coming from the remote FE are accepted. Besides waiting for requests from the remote FE, the local FE can send responses to previously issued operations. Three events are considered in this state:

(E5)AuthEnd
This is an external event, caused by the reception of the AuthEnd operation from the remote FE. The authorized association is ended and all associated resources are released. This event causes a transition out of this state to State 1, **Idle**;

(e6)Response_to_remote_FE
This is an internal event, caused either by the completion of the operations previously issued by the remote FE. The results or errors of these operations are returned to the remote FE. The Terminating FSM remains in the same state; and

(E7)Request_from_remote_FE
This is an external event, caused by the reception of a request from the remote FE. The Terminating FSM remains in the same state.

20.4.2 Three-way mutual authentication FSMs

An outgoing and incoming generic security FSMs for three-way mutual authentication are described in this subclause. The Security FSMs described here are generic machines. They illustrate how the security procedures are handled across the interface between two FEs.

20.4.2.1 Outgoing FSM for Three-Way Mutual Authentication

The outgoing FSM controls the originating side of the interface. It is shown in figure 19-3.

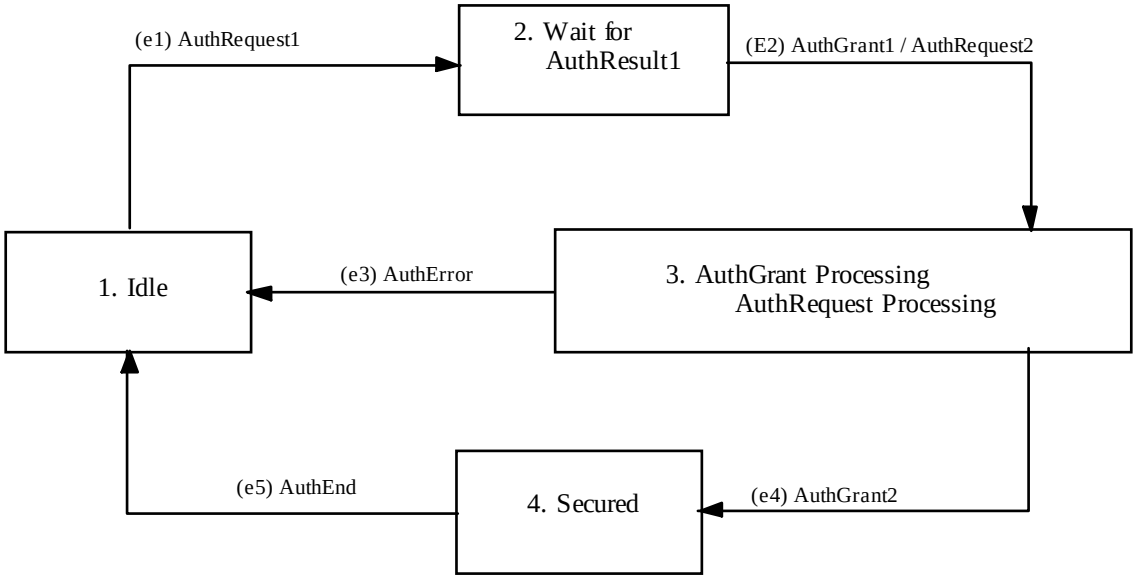


Figure 19-3: Originating FSM States for three-way mutual authentication

20.4.2.1.1 State 1: "Idle"

The following event is considered in this state:

(e1)AuthRequest1
This is an internal event, caused by the need of the SL to create an association with an Remote FE in order to begin accessing a service function. This event causes a transition to the State 2, **Wait for AuthResult1**.

20.4.2.1.2 State 2: "Wait for AuthResult1"

In this state, a response to the initial challenge (sent in the AuthRequest1) from the remote FE is expected. The following event is considered in this state:

(e2)AuthGrant1/AuthRequest2
This is an external event caused by the reception of a response to the initial challenge and a subsequent challenge from the remote FE. This event causes a transition out of this state to the State 3, **AuthGrant Processing/ AuthRequest Processing**.

20.4.2.1.3 State 3: "AuthGrant Processing/AuthRequest Processing"

In this state, the local FE confirms the correct response to the AuthRequest1. In addition, an appropriate response to AuthRequest2 is created. Two events are considered in this state:

(e3)AuthError
This is an internal event, caused by the an error in processing AuthGrant1 received from the remote FE in response to **AuthRequest1**. This event causes a transition out of this state to State 1, **Idle**; and

(e4)AuthGrant2
This is an internal event, caused by the processing of an appropriate response to AuthRequest2 which is issued to the remote FE. This event causes a transition to State 4, **Secured.**

20.4.2.1.4 State 4: "Secured"

In this state, the local FE has established a unilateral authentication with the remote FE. Once the remote FE also progresses to a Secured state requests to and from the remote FE may be processed (these events are not shown for clarification). One event is considered in this state:

(e5)AuthEnd
This is an internal event, caused by the need of the SL to terminate the authenticated access to the Remote FE. This event causes a transition state to State 1, **Idle**.

20.4.2.2 Incoming FSM for Three-Way Mutual Authentication

The incoming FSM controls the terminating side of the interface. It is shown in figure 19-4.

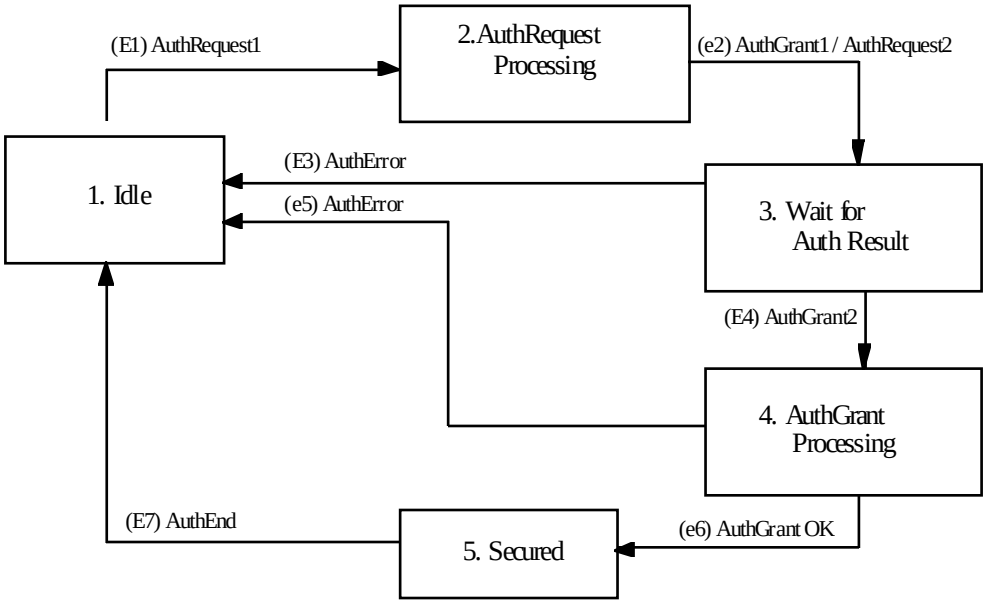


Figure 19-4: Terminating FSM states for three-way mutual authentication

20.4.2.2.1 State 1: "Idle"

The only event accepted in this state is:

(e1)AuthRequest1
This is an external event caused by the reception of AuthRequest1 information from the remote FE. This event causes a transition out of this state to State 2, **AuthRequest Processing**.

20.4.2.2.2 State 2: "AuthRequest Processing"

In this state, an Auth Request has been received from the remote FE. The local FE is performing the authentication procedures. One event is considered in this state:

(e2)AuthGrant1/AuthRequest2
This is an internal event, caused by the successful completion of the AuthRequest1 operation previously issued to the local FE. In addition, the local FE authentication challenge (AuthRequest2) is generated. This event causes a transition out of this state to State 3, **Wait for Auth Result**.

20.4.2.2.3 State 3: "Wait for Auth Result"

In this state, the local FE awaits confirmation from the remote FE that either the AuthGrant1 response to AuthRequest1 challenge was accepted, or a response (AuthGrant2) to the AuthRequest2 challenge is received. Two external events are considered in this state:

(e3)AuthError
This is an external error event, generated by the remote FE, on receipt of an invalid response to its challenge. This event causes a transition out of this state to State 1, **Idle.**

(e4)AuthGrant2
This is an external event, generated by the remote FE, containing the response (AuthGrant2) to the local FE challenge (AuthRequest2). This event causes a transition out of this state to State 4, **AuthGrant Processing**.

20.4.2.2.4 State 4: "AuthGrant Processing"

In this state, the local FE confirms the received response (AuthGrant2) to the challenge (AuthRequest2). Two internal events are considered in this state:

(e5)AuthError
This is an internal error event, generated by the local FE, on receipt of an invalid response to its challenge. This event causes a transition out of this state to State 1, **Idle.**

(e6)AuthGrant OK
This is an internal event, generated by the local FE, on receipt of a valid response to its challenge. This event causes a transition to State 5, **Secured.**

20.4.2.2.5 State 5: "Secured"

In this state, the local FE has established a unilateral authentication with the remote FE. Since the remote FE is also in a Secured state, requests to and from the remote FE may be processed (these events are not shown for clarification). One event is considered in this state:

(e7)AuthEnd
This is an external event, caused by the need of the SL to terminate the authenticated access to the Remote FE. This event causes a transition state to State 1, **Idle**.

Annex A (normative): INAP SDLs

A.1 Introduction to the INAP CS1 and CS2 SDL models

The three clauses of this annex (A.1, A.2 and A.3) are considered an integral part of this EN. The SDL diagrams contained in A.3 are normative; the normative information provided by the SDL diagrams is the dynamic behaviour at the external interfaces. The internal behaviour and structure have been introduced for modelling purposes. No SDL specification of the SCF is provided, the behaviour of the SCF is specified in clauses 12 and 17 of the present document.

The INAP IN CS1 and IN CS2 models are specified with SDL 96 in an object-oriented way. The IN CS2 specification inherits the IN CS1 and only specifies the difference between IN CS1 and IN CS2.

The SDL model specifies precisely and unambiguously the external behaviour and the interworking between the different functional entities: CS, CSA, SSF-FSM, BCSM. However, the internal behaviour is only for information and the SDL model does not impose any requirements on the internal structure of an implementation. The data structures used are imported from clauses 4, 5, 6 and 10 of this EN (ASN.1 definitions).

The conformance test suite for Core INAP CS2 has been automatically derived from the model. The model can also be used as a platform for service emulation etc.

Figure A.1-1 shows the IN CS1/CS2 information model. There are zero, one or many Call Segments in one Call Segment Association. There is a one to one correspondence between Call Segment and Connection Point, and so on. There are two objects of type BCSM, the OriginatingBCSM and the TerminatingBCSM.

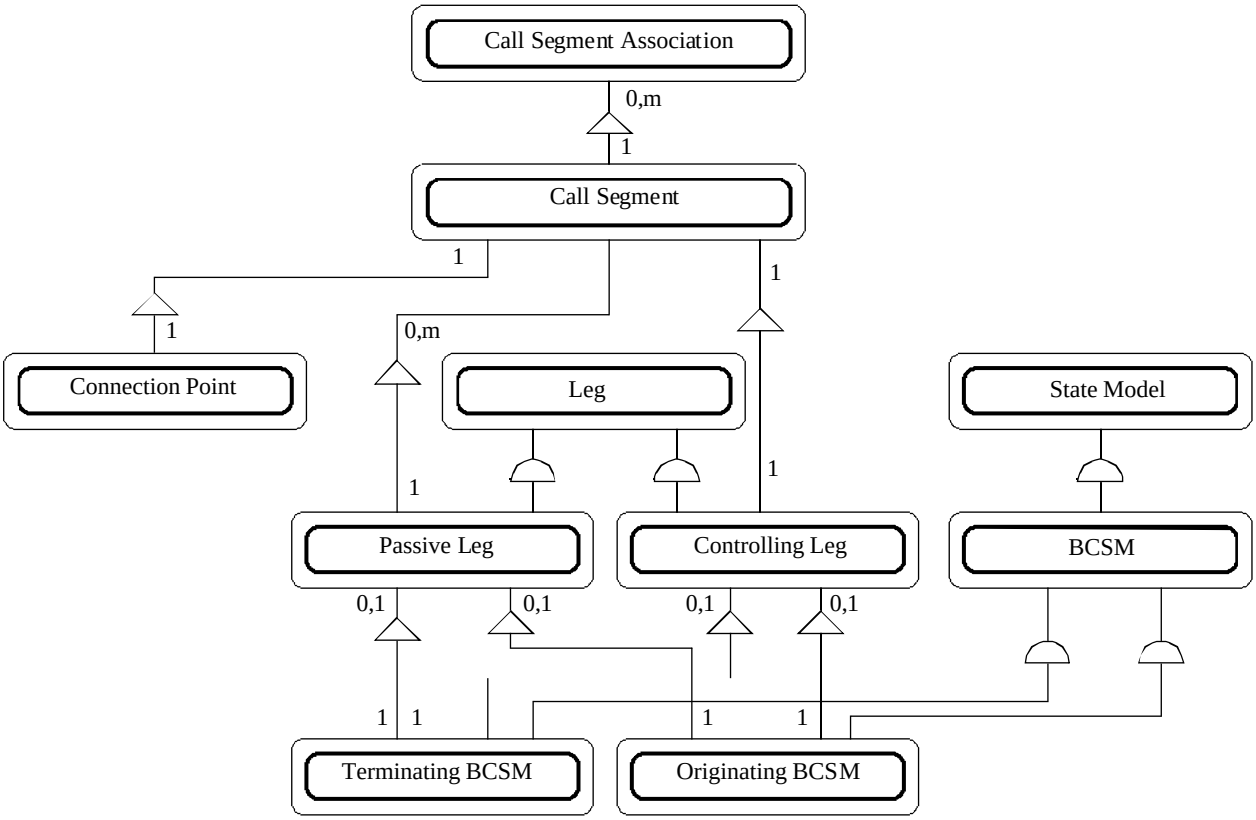


Figure A.1-1: IN CS1/CS2 information model

Figure A.1-2 shows the SDL model of CS1 SSF/CCF. The IN CS2 equivalent is shown in figure A.1-5. It inherits most of the items of the IN CS1 model and refines the behaviour of them. The objects in the information model of figure A.1-1 map to the SDL model of figure A.1-2 in the following way.

The Call Segment Association object is modeled by a process type in the SDL model. The CallSegmentAssociation manages:

- the creation of CallSegments; and
- the dialogue with the SCF (via the Interrupt Handler (IH) and TCAP).

The Call Segment object is modeled using two process types in the SDL model: CallSegment and SSF-FSM.

The SSF-FSM process type corresponds with the FSM for Call Segment described in subclause 11.5.2, and not with the SSF FSM described in subclause 11.3.

The process type CallSegment manages the:

- IDs of the connected legs (connection view). This data structure models the Connection Point object;
- creation of the O-BCSMs and T-BCSMs;
- creation of the SSF-FSM;
- filtering of DPs;
- processing of IN operations changing the connection view (CPH-operations, CON, ICA).

The CallSegment process defines states for the CS. This approach is different from the CVS approach in ITU-T Recommendation Q.1224 [26]: the CVS approach explicitly names combinations of associated CSs in a particular state. The number of such combinations is infinite, since the number of CSs in a CSA has no upper bound.. Instead, in the SDLs the Call Segment states are given a name (e.g. Stable-2-Party) and the CSA is represented as a set of associated CSs, possibly in different states.

The process type SSF-FSM manages the:

- processing of IN operations;
- handling of DPs (EDPs and TDPs).

In order to centralize the DP processing and thus simplify the model, the SSF-FSM is created even though no TDP may be armed for the call.

The Connection Point object is modelled by a data structure as described above.

The Leg object is also a data structure within the process type CallSegment. It is identical to the data structures of Passive Leg and Controlling Leg. The data structure contains the status of the leg and a pointer to the BCSM connected to the leg.

T-BCSM and O-BCSM are both objects of the class BCSM in the information model. The SDL model is not exactly constructed this way. Since the O-BCSM and T-BCSM significantly differ, they are modelled in two completely independent process types.

The SDL model contains one additional process type: InterfaceHandler. The InterfaceHandler is the permanent manager of the CCF. When the interpretation of the SDL specification begins, the IH is the only process that exists. Then during the course of a call setup the IH, after having received the appropriated messages from the environment, creates the call segment association. It also handles the dialogue with the SCF and passes the primitives between the Signalling Control Interface and the CSAs and between the SCF interface and the CSAs.

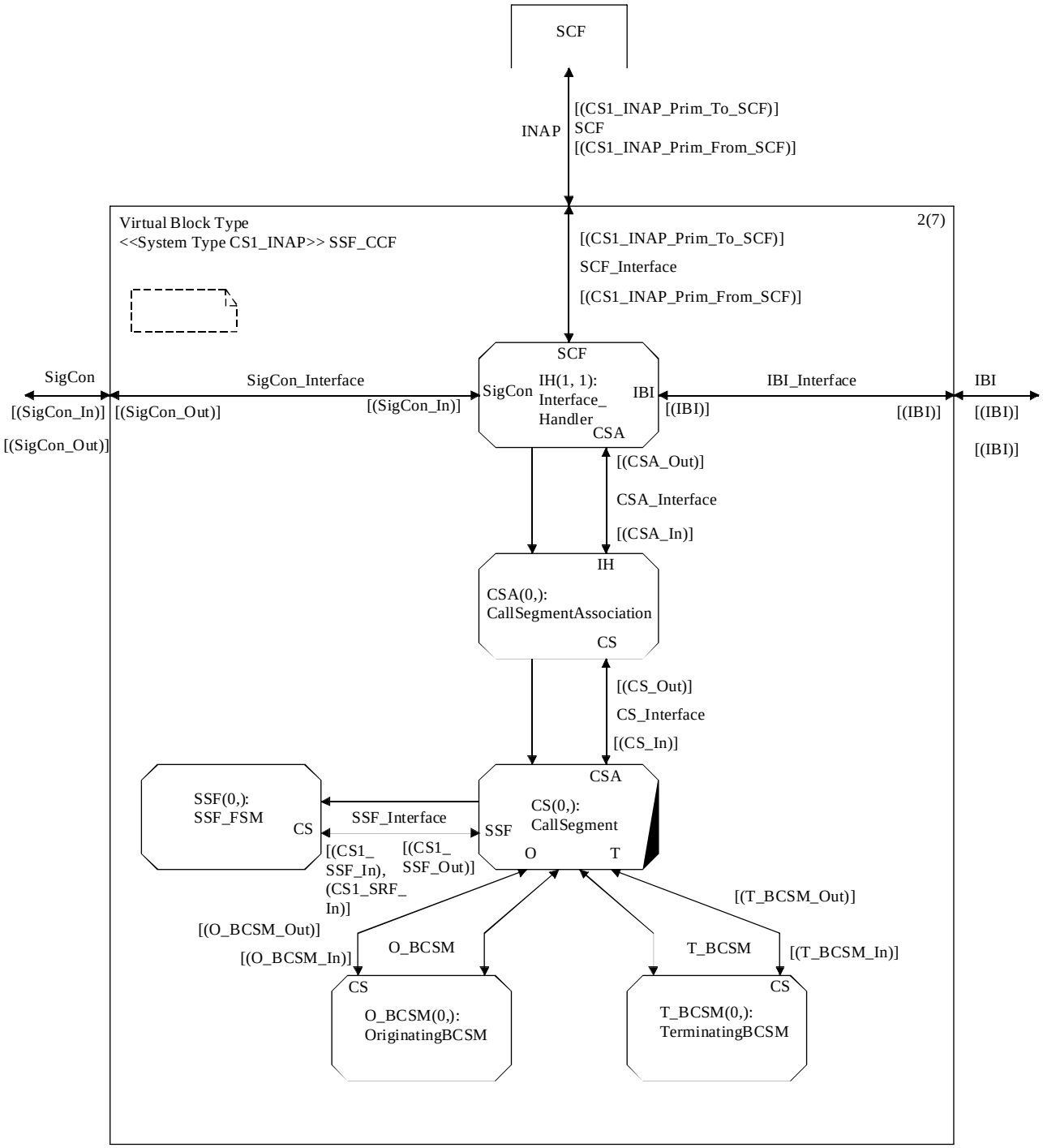


Figure A.1-2: The INAP IN CS1 SDL model, half-call view

NOTE: The figure A.1-2 will be updated to include the SSME-FSM.

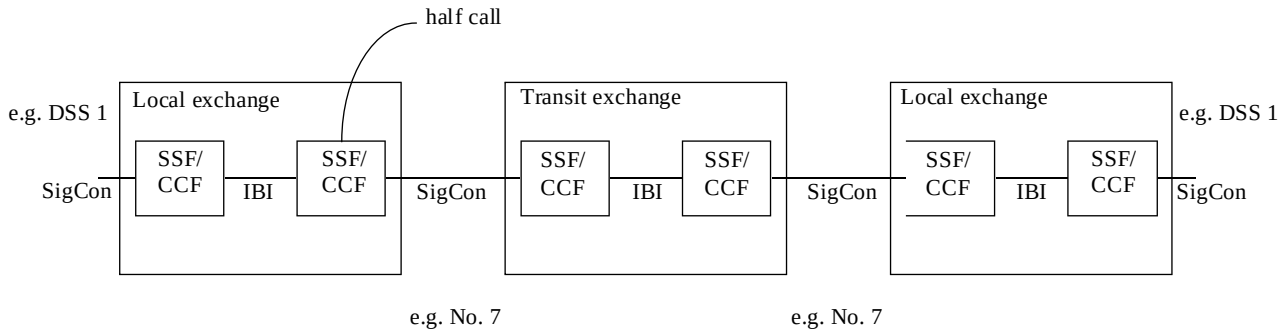


Figure A.1-3: General example scenario for INAP showing the SSF/CCF only

Figure A.1-3 shows a general example scenerio for IN related to the SDL model. The Intra-BCSM Interface (IBI) interface is an internal intra BCSM interface between two half calls.

The full SDL model consists of two half calls as indicated in figure A.1-4. Figure A.1-2 is a refinement of the SSF/CCF block. It shows one half call under control of the SCF, including the service switching and CCFs. In order to operate the model, the Block SSF_CCF needs to be doubled to get two half calls. In this way the full functionality of the interworking between the O-BCSM and T-BCSM can be simulated.

The model has three interfaces. The SigCon interface could, for example, be a DSS1 interface. The Messages on SigCon are abstract and have to be mapped to a real protocol, e.g. DSS1, in a concrete case. IBI is completely internal to a switch. The INAP interface to the SCF is the one with the standardized INAP messages.

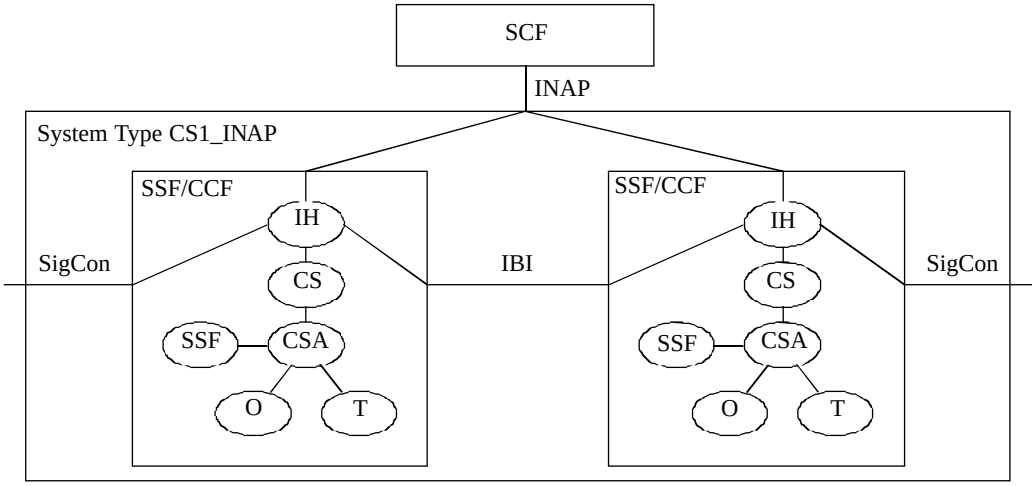


Figure A.1-4: Two half calls

NOTE: The figure A.1-4 will be updated to include the TCAP.

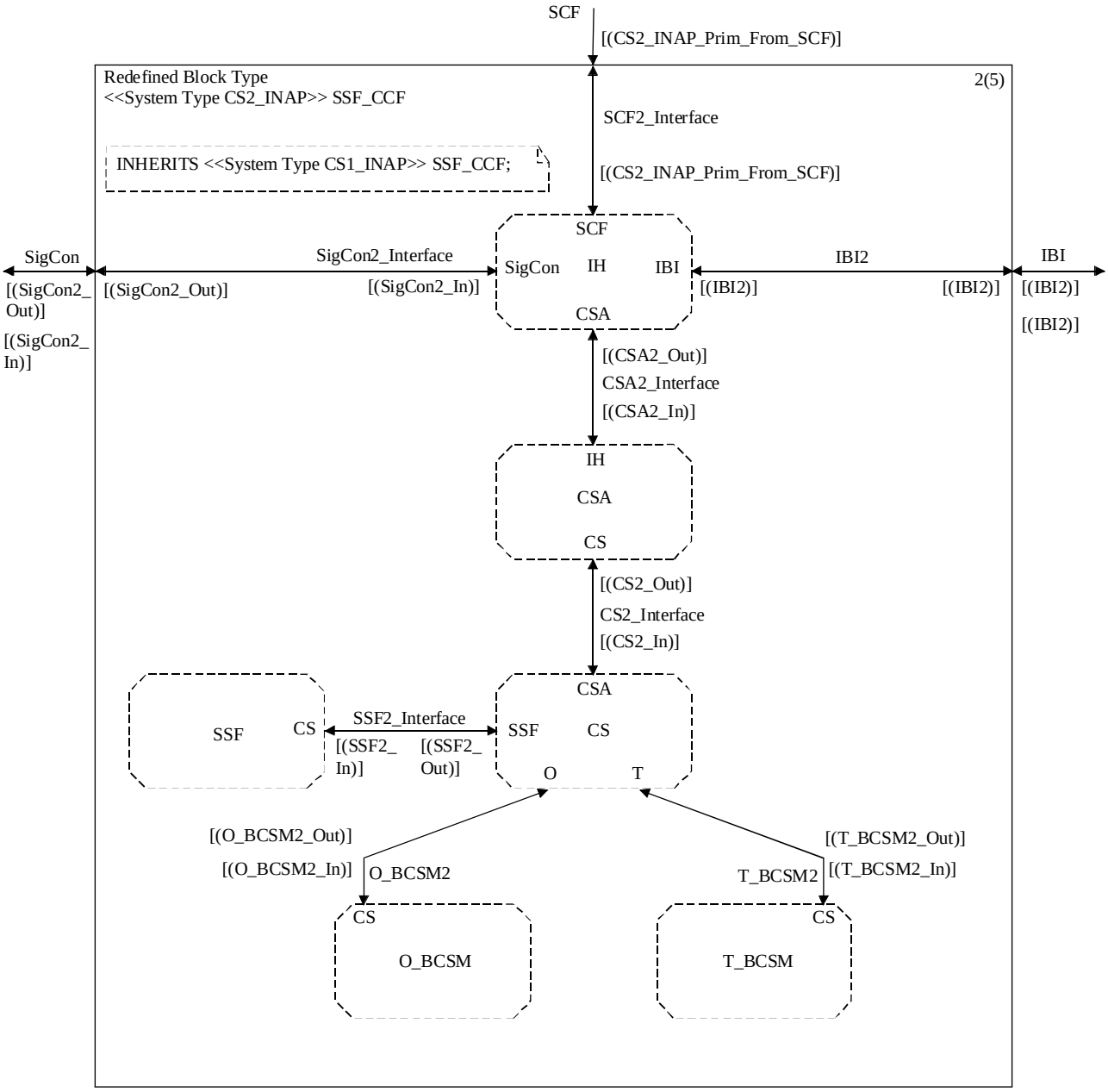


Figure A.1-5: INAP IN CS2 model which inherits the IN CS1

NOTE: The figure above will be updated to include the SSME-FSM.

Example for the interworking of the SSF/CCF SDL processes

The following message sequence chart (MSC) shows the interworking between the environment, IH, CSA, CS, SSF and O_BCSM after a SetupInd is received through the SigCon interface. For brevity, only the first few signal exchanges are shown.

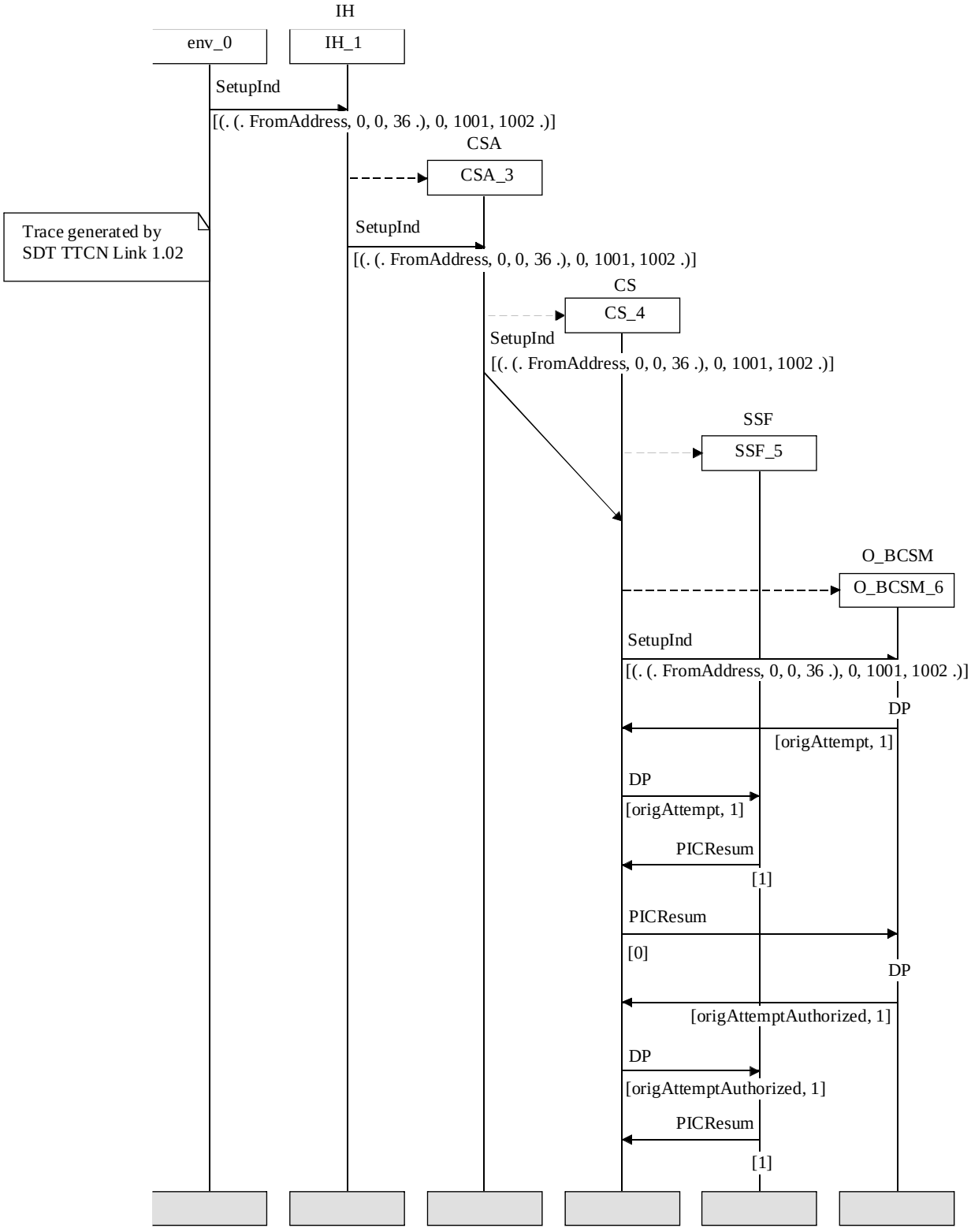


Figure A.1-6: MSC for the interworking of the SSF/CCF SDL processes

NOTE: The MSC above will be updated with a newly generated one.

Example for the Three Party Call setup as seen from the environment

The following MSC shows an example of the interworking between the SDL model and the environment during a Three Party Call. The MSC is developed according to the architecture of figure A.1-4. The internal message exchange of the two half call CS2_INAP model is not shown in the MSC. There are three local exchanges involved: A, B and C. From the point of view of the MSC they are all environments, together with the SCF. For the sake of clarity these environments are each put on a different process instance axis, each one corresponding to one Point of Control and Observation (PCO).

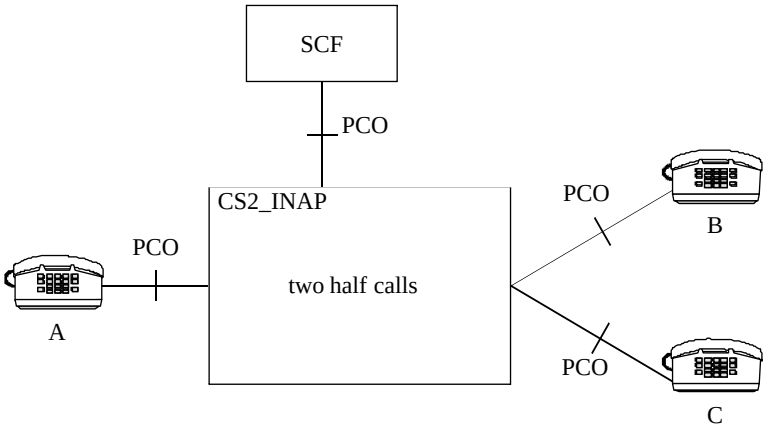


Figure A.1-7: Three Party Call setup - two half calls

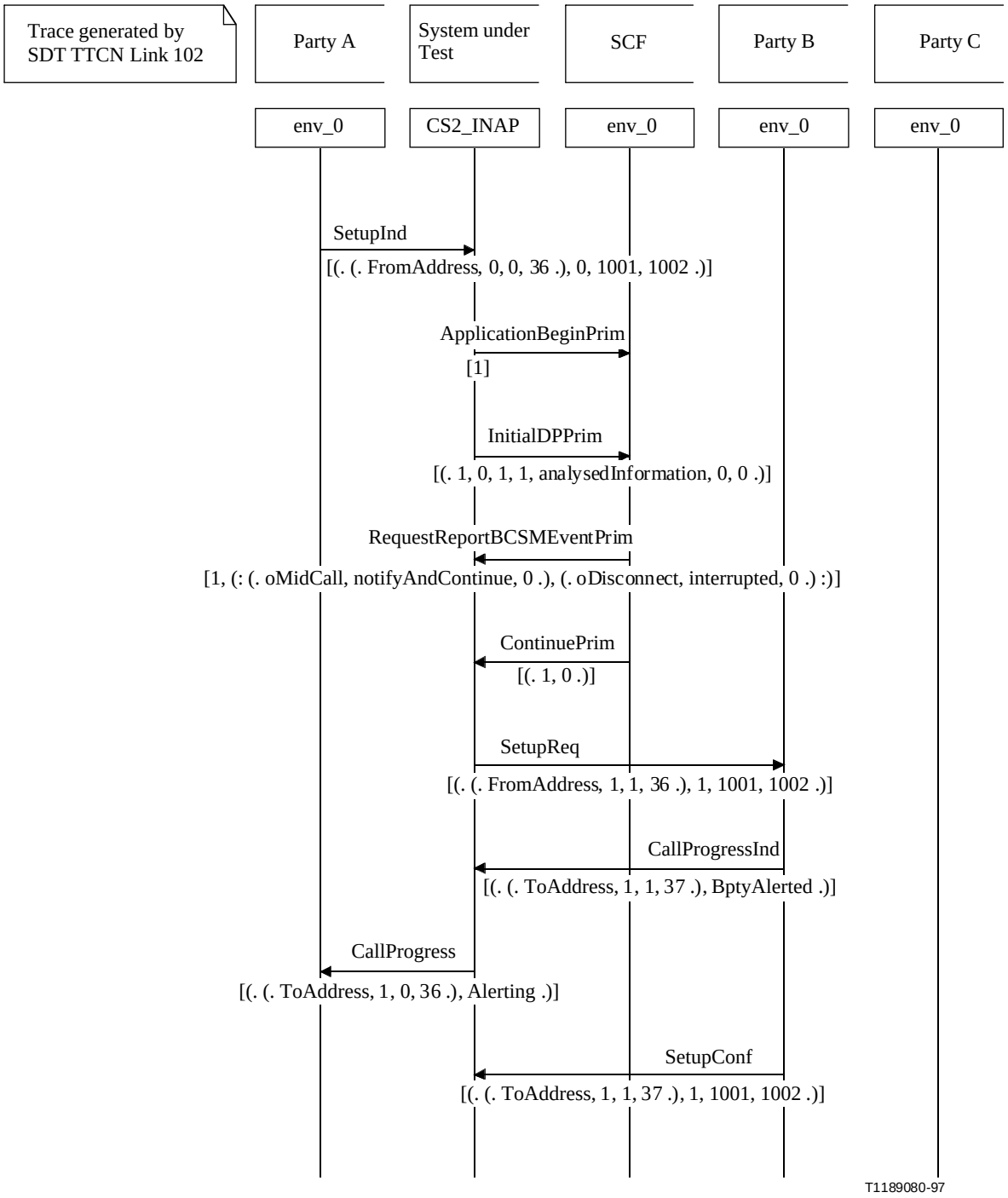


Figure A.1-8: MSC - Three Party Call (sheet 1 of 2)

NOTE: The MSC above will be updated with a newly generated one.

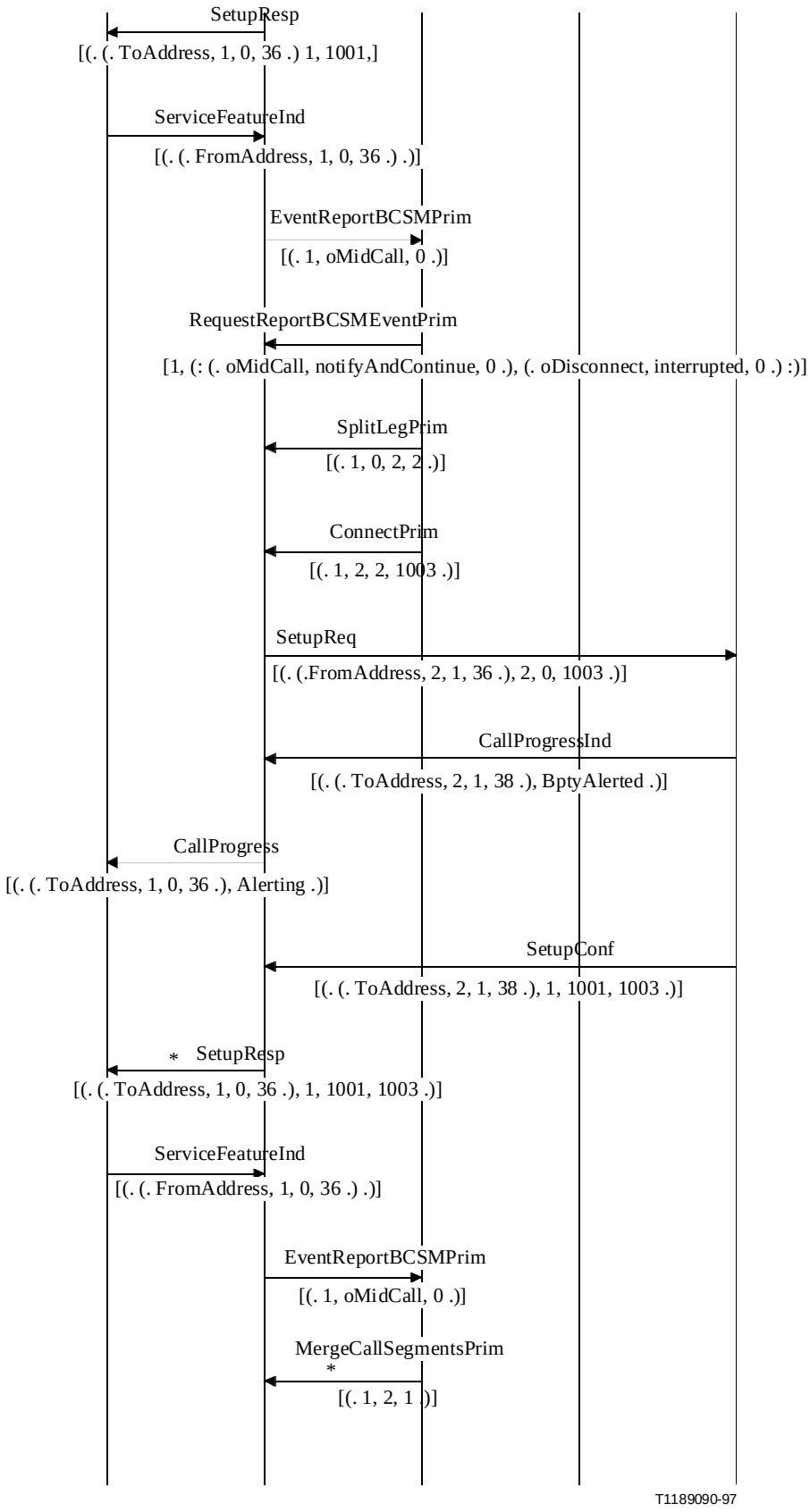


Figure A.1-8: MSC - Three Party Call (sheet 2 of 2)

NOTE: The MSC above will be updated with a newly generated one.

A.2 Transition diagrams

A.2.1 Call segment association transition diagram

A.2.1.1 Definition of states and transitions

The states identified for the CSACV transition diagram are:

Idle/Null: The state Idle indicates that no CSACV instance exists but may be created; this state is e.g. entered when a particular CSACV instance is deleted. The state Null is entered when a call segment association is created without any call segments.

One Segment: This state represents a call segment association containing one call segment.

Multi-Segment: This state represents a call segment association containing multiple call segments.

A.2.1.2 General functional procedures

A.2.1.2.1 Queuing of BCSM events on receipt of a Radio Regulations Board (RRB) operation in the CSACV

In case a RRB operation is received by the CSACV with legIDs for which no corresponding passive leg yet exist, the "to be armed" BCSM events shall be stored. When a CON or ICA operation subsequently creates a new passive leg with a legID equal to a legID for which BCSM events have been stored, the stored BCSM events shall be retrieved and transferred to the exiting or newly created CS instance. The armed events have to be stored at the CSA level and not in the initial created CS because it can for example happen that this CS is deleted and with it the stored events.

A.2.1.3 Transition diagram for Call Segment Association (CSACV)

The transition diagram for the CSACV is given in figure A.2.1.

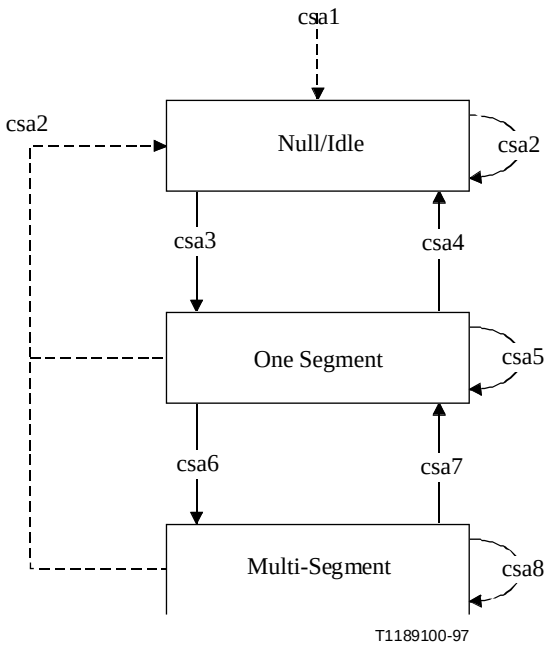


Figure A.2-1: CSACV transition diagram

The CSACV transition diagram contains the following transitions:

csa1: Create Call Segment Association

csa2: Delete Call segment Association

csa3: SetupInd, SetupReqInd, InitiateCallAttempt

csa4: ReleaseCall

csa5: DisconnectLeg

csa6: SplitLeg, InitiateCallAttempt

csa7: ReleaseCall, MergeCallSegments

csa8: SplitLeg, InitiateCallAttempt, MergeCallSegments, DisconnectLeg, MoveLeg

A.2.2 Call Segment Connection View (CSCV) transition diagram

A.2.2.1 Definitions of States and transitions

The states identified for CSCV transition diagram are:

Idle/Null: The state Idle indicates that no CSCV instance exists but may be created; this state is e.g. entered when a particular CSCV instance is deleted. The state Null represents that no call processing is active and there is no controlling leg or passive legs connected to the connection point.

Originating Setup: This state represents an originating two-party in the setup phase with a joined controlling leg and a pending passive leg with an associated O_BCSM.

Terminating Setup: This state represents a terminating two-party call in the setup phase with a "pending" controlling leg and a joined passive leg with an associated T_BCSM.

Stable 2-Party: This state represents a stable or clearing two-party call, and is either an originating or a terminating call from the perspective of the controlling user with a "joined" controlling leg and a "joined" passive leg with either an O_BCSM or T_BCSM associated with it.

1-Party: This state represents a 1-party call being originated with a "joined" controlling leg status and no passive legs. The BCSM is associated with the controlling leg since no passive leg is connected to the CP (Connection Point).

Originating 1-Party Setup: This state represents a 1-party call being originated on behalf of the network with a "surrogate" controlling leg and a "pending" passive leg with which an O_BCSM is associated.

Terminating 1-Party Setup: This state represents a 1-party call being terminated on behalf of the network with a "surrogate" controlling leg and a "joined" passive leg to which an T_BCSM is associated.

Stable 1-Party: This state represents a 1-party call originated on behalf of the network; with a "surrogate" controlling leg and a joined passive leg with either an O_BCSM or T_BCSM associated with it , that is in a stable or clearing phase.

Forward: This state represents a forwarded call. Call processing for the originally calling party passive legs are either in an originating or terminating call setup/stable phase, whereas call processing for the forwarded-to passive leg is in an originating call setup phase. The call segment has a surrogate controlling leg with one or two calling party joined passive legs with either an associated O_BCSM or T_BCSM, and a pending "forwarded to" passive leg with an associated O_BCSM.

Transfer: This state represents a transferred call. The call between the involved passive legs is in the stable or clearing phase.

On Hold: This state represents a held call, where the controlling user has put the involved remote parties on hold and is participating in another call.

Stable Multi-Party: This state represents a stable or clearing multi-party call.

A schematic representation of the states identified for the CSCV is given in the following figure A.2-2.

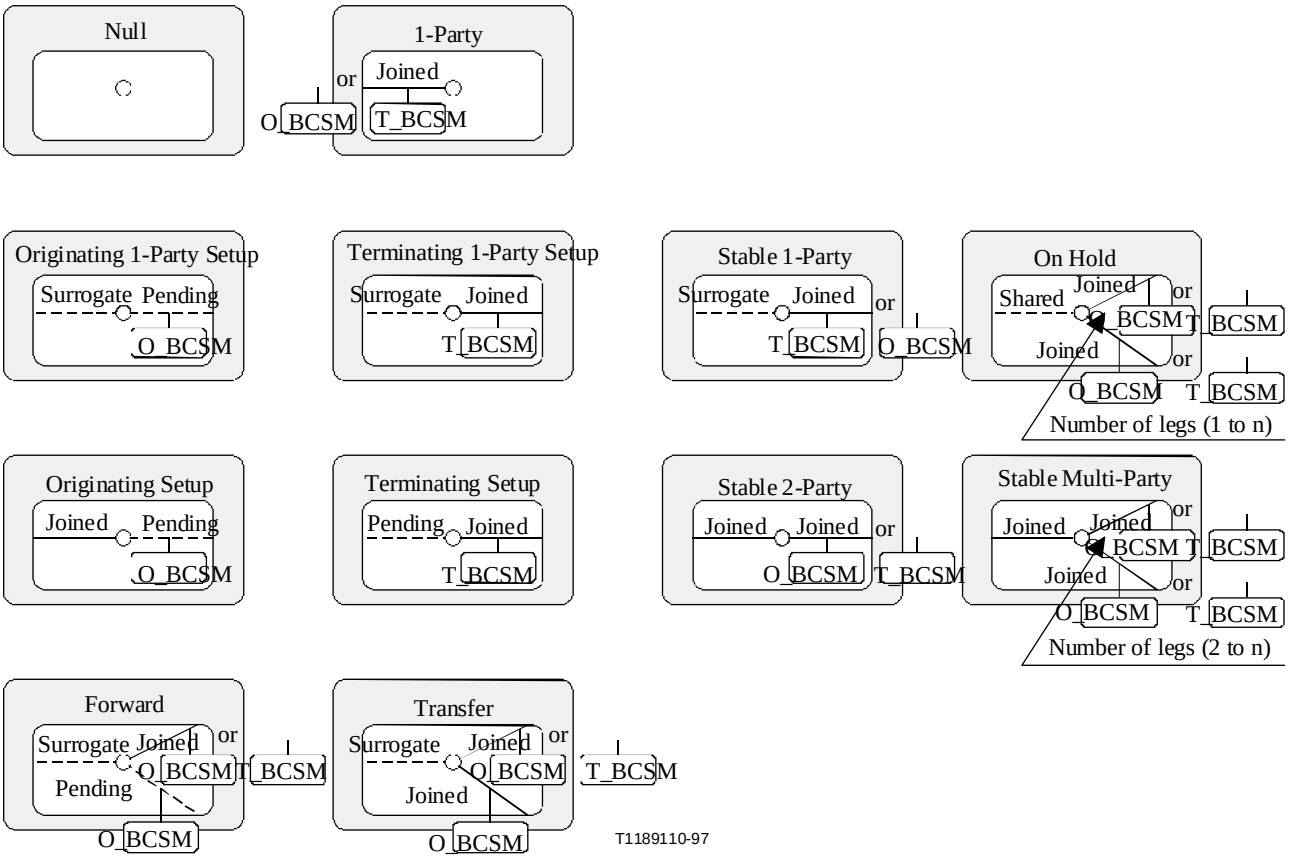


Figure A.2-2: States of the CSCV transition diagram

A.2.2.2 General functional procedures for the CSCV

The following general principles shall apply:

A.2.2.2.1 BCSM type indication

A "BCSM type" attribute shall be allocated with the controlling leg at the moment this leg is created, indicating if the controlling leg belongs to an Originating or a T-BCSM, i.e. an indication for the type of BCSM in which call triggering occurred.

When a controlling leg is left in a CS, the "BCSM type" attribute shall determine which BCSM shall apply. The conditions of the leg i.e. the armed EDPs, the ApplyChargingReport pending, the EventNotificationCharging pending, and the CallInformationReport pending are also applied to the same leg after creation of an BCSM instance (e.g. after a SplitLeg). The latter entails that if no EDPs and/or reports were armed for the leg in relation to the created BCSM type, then an explicit arming is to be made using e.g. the RequestReportBCSMEvent operation, if needed.

In order to obtain full alignment with the SL design in the SCF no mapping of events from O_BCSM to T_BCSM events and vice versa shall occur. This means that the DP at which call processing is to be suspended shall select in the associated BCSM instance the actual state of the call as seen from the legs viewpoint; e.g. if an "active call PIC" exists then the BCSM instance shall be put in MidCall DP of the active BCSM state when the leg is left in a CS after e.g. a DisconnectLeg or SplitLeg operation. The following is noted for the O_BCSM and T_BCSM instances:

- When the O_BCSM instance is put in the O_Mid_Call of the O_Active state then:
  - a transition to the Collect_Information, Analyse_Information or Select_Route is possible by means of sending the appropriate operations from the SCF. This allows to support follow-on calls in accordance with the extended BCSM transitions as described in ITU-T Recommendation Q.1224 [26].
- When the T_BCSM instance is put in the T_Mid_Call of the T_Active state then:
  - in order to allow to set-up an originating call connection to another terminating called party the ContinueWithArgument followed by the InitiateCallAttempt and the MergeCallSegments operations may be sent from the SCF in order to perform a "transfer call" (see figure A.2-3). It is noted that the transfer of a call in this case where the controlling leg has the T-BCSM type attribute is only possible in a serving node exchange (not in a transit exchange).

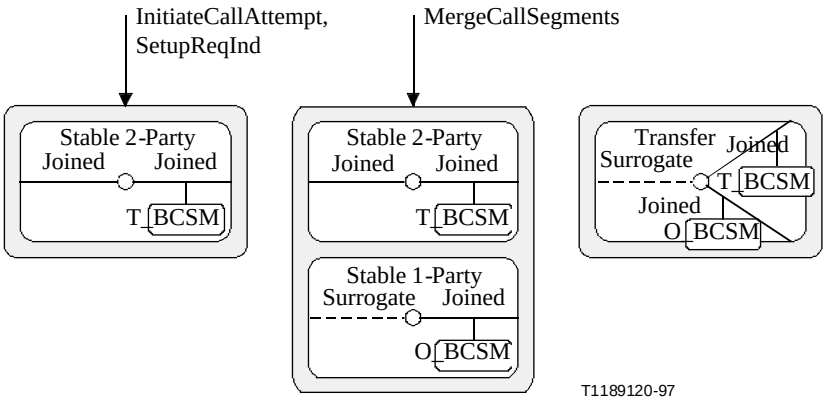


Figure A.2-3: Transfer Call at T_BCSM

If in the above-mentioned cases, the controlling joined leg is the only leg left in a CS due to a SCF operation (DisconnectLeg, SplitLeg, MoveLeg etc.), the CS shall be put in the 1-Party state with the associated SSF_FSM state in "Waiting For Instructions". In order to resume from the O_Mid_Call and T_Mid_Call DPs of the associated BCSM O_Active and T_Active PIC states a ContinueWithArgument operation can be sent by the SL.

It should be noted that the problem of "stranding" a "surrogate" controlling leg does not exist as it does for a "Joined" controlling leg, because it is assumed that a "stranded" surrogate leg will be destroyed due to the fact that no voice path is associated with this controlling leg.

A.2.2.2.2 Event handling rules

The following inter-BCSM precedence rules for the reporting to the SCF of events to be signalled for the controlling leg shall apply:

- An event shall only be reported once to the SCF irrelevant of the number of passive legs per CS connected to the controlling leg (via CP).
- A "BCSM type" attribute on the controlling leg indicates if the controlling leg belongs to an Originating or a T-BCSM. The "BCSM type" attribute is set to reflect the type of BCSM (originating or terminating) in which the trigger or event occurred at the moment the CS was created.
- If the controlling leg belongs to an O-BCSM it is reported as an originating event.
- If the controlling leg belongs to a T-BCSM it is reported as a terminating event.

A.2.2.2.3 Creation of O/T_BCSM based on "BCSM type" attribute

A CS with only a controlling leg (leg status "joined") left is assigned a BCSM instance to supervise the leg. The "BCSM type" attribute set on the controlling leg is used to reflect the type of BCSM (originating or terminating) to be assigned. The corresponding SSF_FSM for the CS shall be put in "Waiting_For_Instructions" state where a leg is left in a CS due to a CPH (e.g. DisconnectLeg) operation and call processing is suspended at the Mid Call DP of the Active state in the associated BCSM.

When the controlling leg becomes connected to the passive leg, no BCSM instance shall be connected to the controlling leg. The BCSM instance which was connected to the controlling leg disappears in case a passive leg is moved (imported) to the Call Segment (e.g. MergeCallSegments operation). If a new passive leg is created (e.g. Connect operation) within the CS then the existing BCSM instance becomes connected to the passive leg.

An export of a leg due to the execution of a MoveLeg, SplitLeg or MergeCallSegments operation for a passive leg can only be executed when the O_BCSM is at least in the Send_Call PIC and for the T_BCSM in the T_Active PIC. This entails that these operations cannot be executed when the passive leg is in the pending status.

A.2.2.2.4 Release of a Call/Connection

The release of a Call/Connection for a normal call where both call segments are in the "Stable_2_Party" state (with the associated O_BCSM and T_BCSM in the active state) is given in the figure A.2-4.

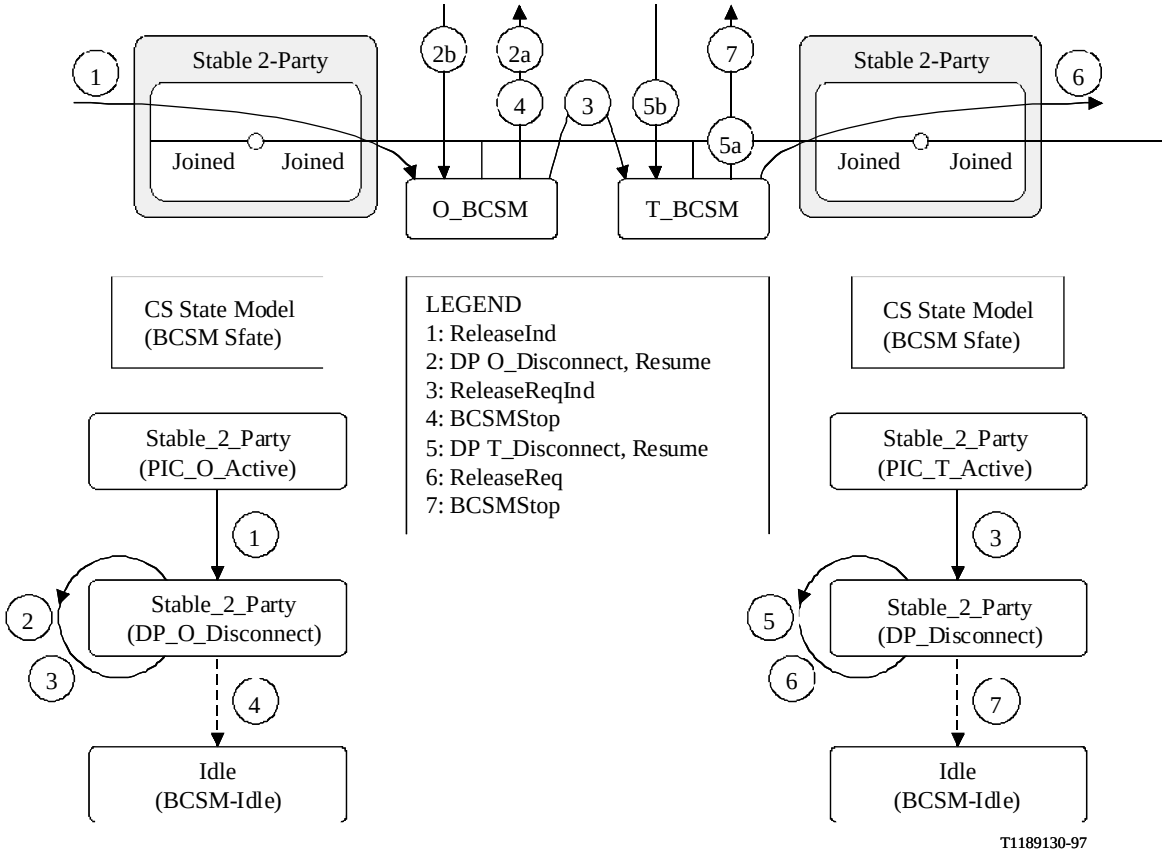


Figure A.2-4: Release of Call/Connection where call segments are both in "Stable_2_Party"

The release of a Call/Connection for a redirected call where the call segments are in the "Stable_2_Party" state and in the "Forward" states is given in the figure A.2-5.

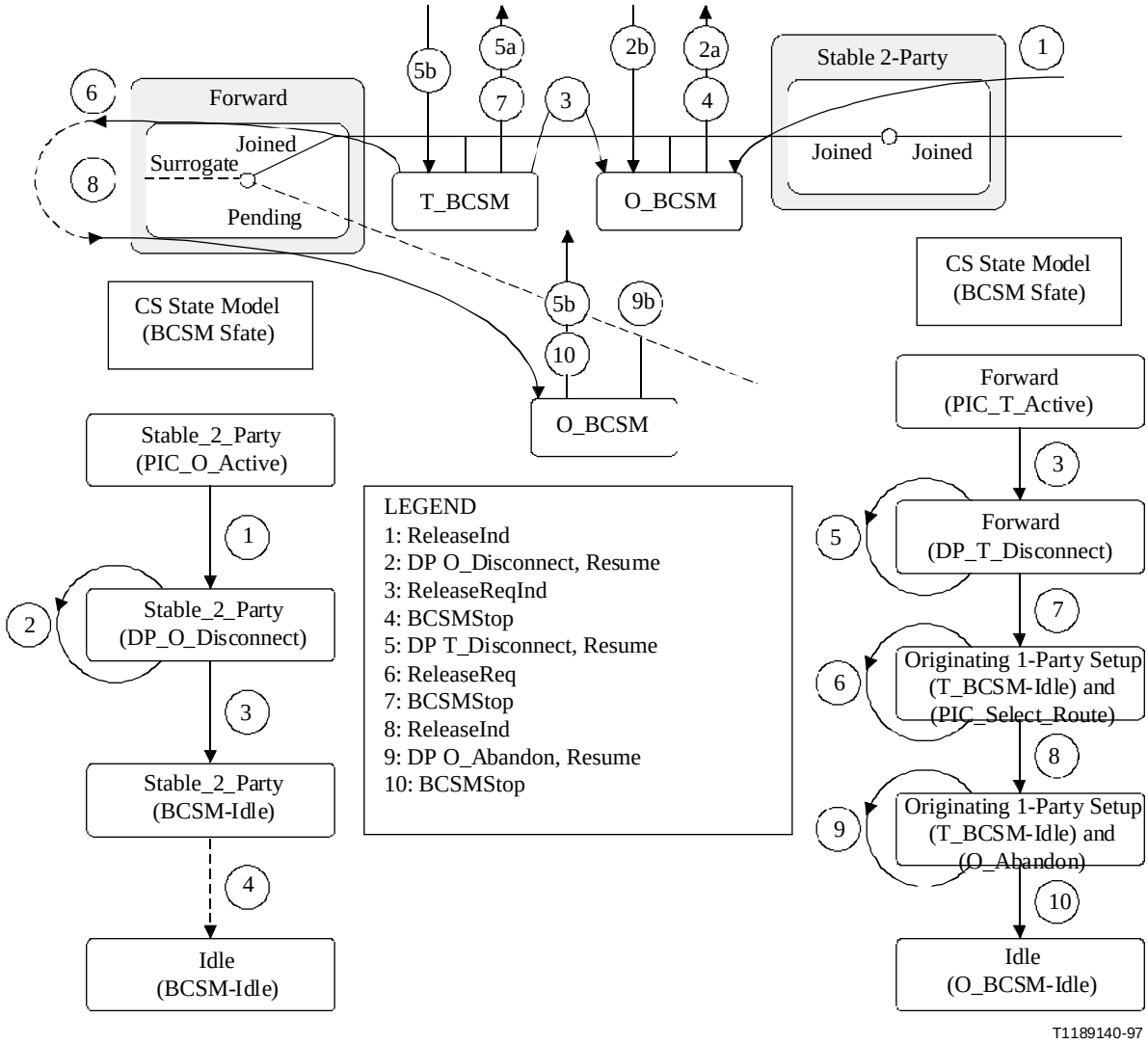


Figure A.2-5: Release of Call/Connection where the call segments are in the "Stable_2_Party" state and in the "Forward" state

The release of a Call/Connection for a redirected call where the call segments are in the "Stable_2_Party" state and in the "Transfer" states is given in the figures A.2-6a and b.

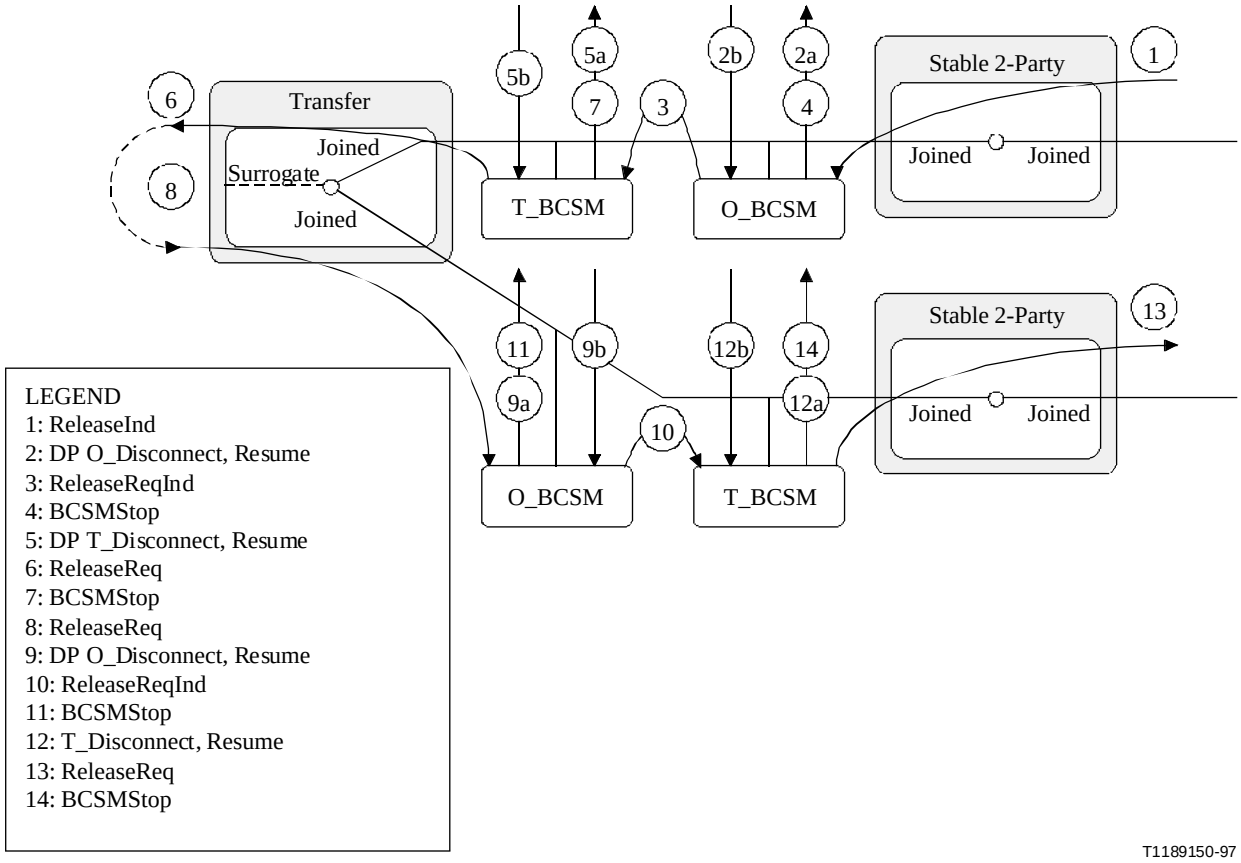


Figure A.2-6a: Release of Call/Connection where the call segments are in the "Stable_2_Party" state and in the "Transfer" state (information flow)

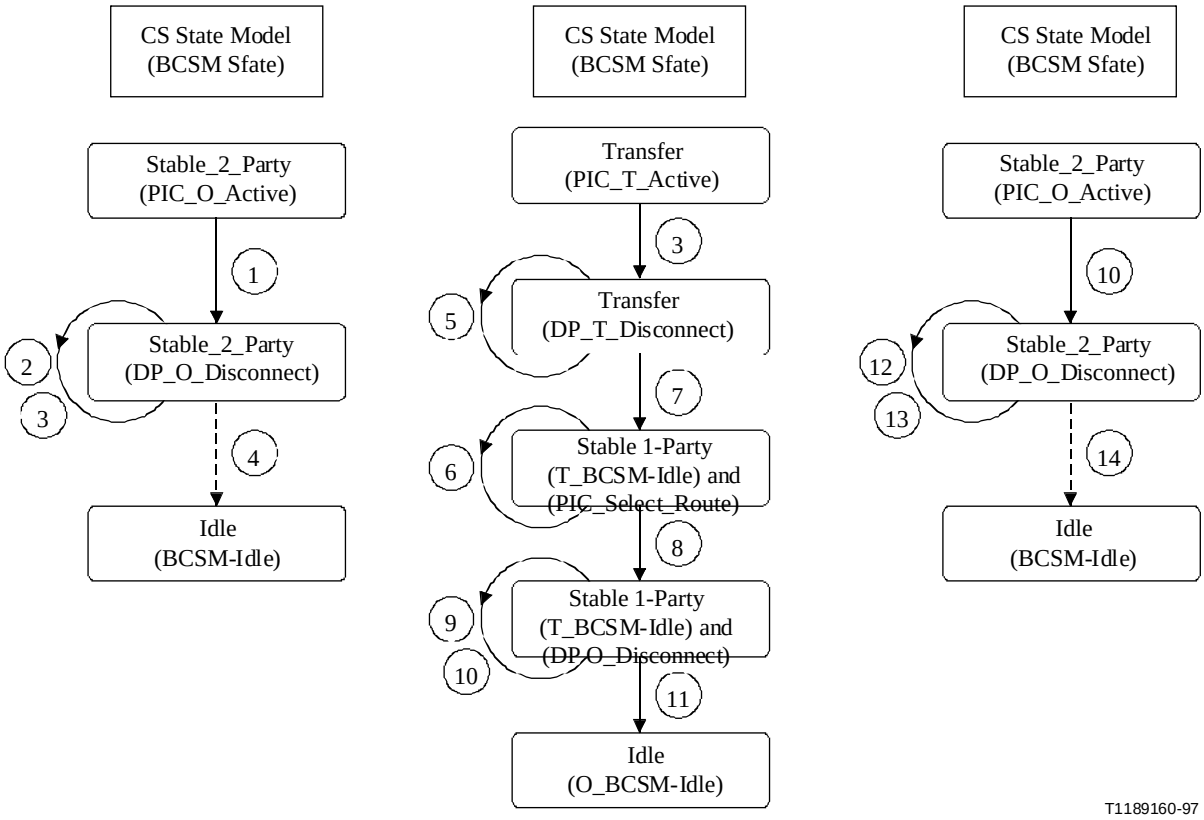


Figure A.2-6b: Release of Call/Connection where the call segments are in the "Stable_2_Party" state and in the "Transfer" state (state transitions)

A.2.2.2.5 Disconnect Leg Operation

A DisconnectLeg operation for a controlling and passive leg shall physically release the specified leg from the connection point towards the remote user. The information flows depicting the release sequences as a result of a DisconnectLeg operation for the controlling (c) and passive legs of a Call/Connection for a normal call where both call segments are in the "Stable_2_Party" state (with the associated O_BCSM and T_BCSM in the active state) is given in the figure A.2-7.

The information flows for the receipt of the following operations are illustrated:

- The following actions occur when a DisconnectLeg operation (for the controlling leg) is received from the SL:
  - a ReleaseReq signal is sent to the CCFA with a cause value "disconnect controlling leg" in the PIC_O_Disconnect (or PIC_T_Disconnect), and;
  - the BCSM shall transit to the appropriate MidCall_Active PIC substate depending on the fact whether a passive leg or not is still associated with the controlling leg.
- The following actions occur when a DisconnectLeg operation (for the passive leg) is received from the SL:
  - a ReleaseReqInd signal is sent to remote BCSM CCFA with a cause value "disconnect passive leg" in the DP_O_Disconnect (or DP_T_Disconnect), and;
  - the BCSM shall transit to the appropriate Disconnect PIC substate depending on the fact whether a controlling leg or not is still associated with the passive leg.

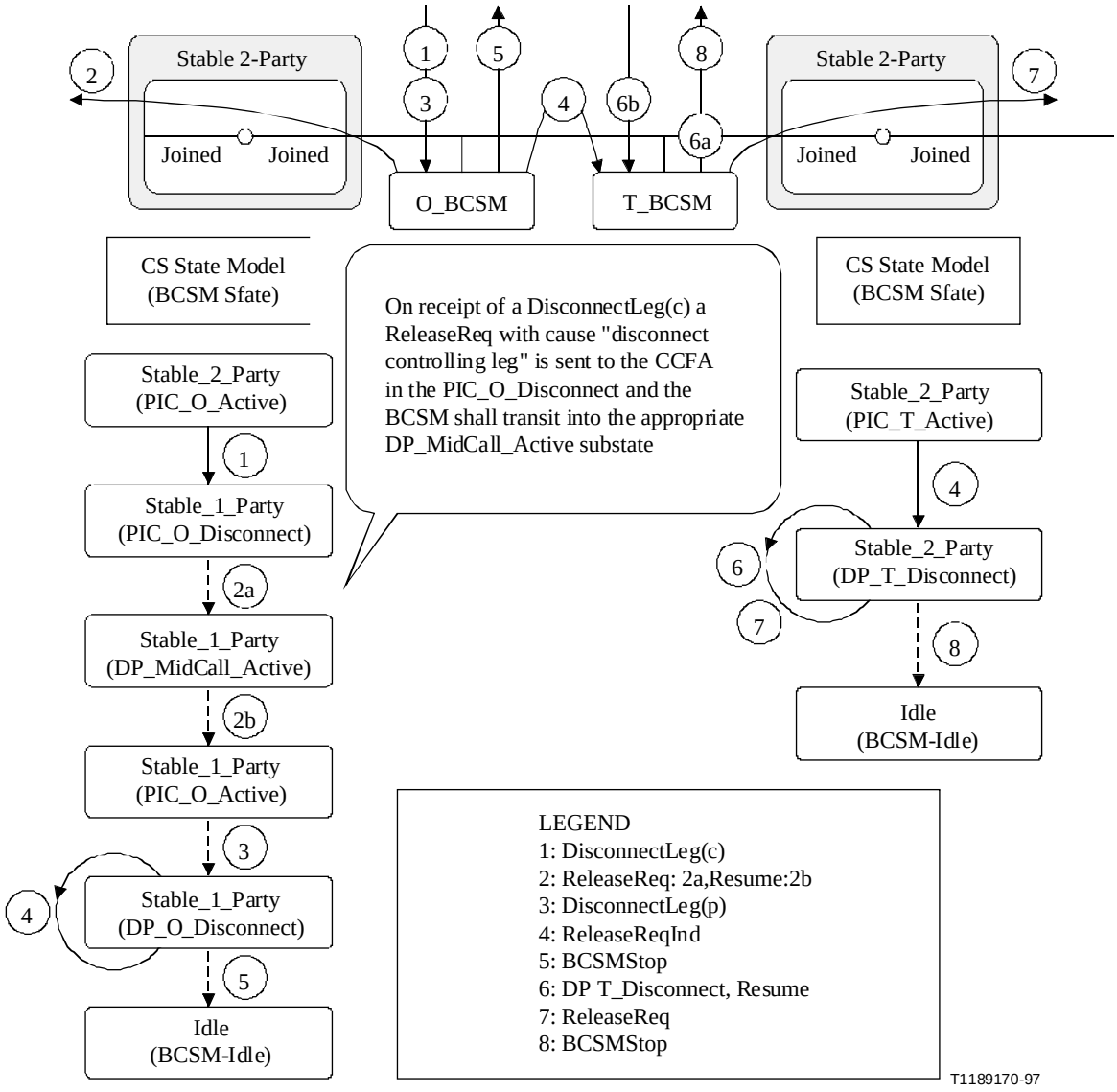


Figure A.2-7: Disconnect Leg of Call/Connection where call segments are both in "Stable_2_Party"

A.2.2.2.6 User interactions during the SSF-FSM "Monitoring" state

During the "Monitoring" state of the SSF-FSM it shall be possible to perform user interactions in order to send tones, announcements and display information.

A.2.2.2.7 Call Segment and associated BCSM states for CPH operations

The following principles apply for the Call Segment and associated BCSM states for CPH operations:

- If a CPH operation (SplitLeg , DisconnectLeg, MergeCallSegments, MoveCallSegment or MoveLeg) is received in the Monitoring state, the FSM's for the involved Call Segments shall first transit to the "Waiting for Instruction" state while the associated BCSM instances within the involved Call Segments shall move from the O/T PIC of the corresponding O/T MidCallDP in order to handle subsequent EDP rearming. It shall be noted that when the BCSM instance is in a DP it shall stay in this DP after processing of the CPH operation. This involves that only some PIC transitions to MidCall DP's are possible as described in the templates of the operations.

- The receipt of a CPH operation received and/or processed in the state Waiting for Instructions shall not cause the change of the state Waiting for Instructions.

- For these CPH operations which create for the controlling leg a new BCSM, (stranded leg) only this BCSM shall be put into the DP-O/T-MidCall of the active state, the BCSM states of the other involved CS's should transit from the PIC into the corresponding DP state or remain in the DP wait state , the associated CS SSF_FSM for the newly created BCSM and the other involved CSs shall be put into the "Waiting for Instruction" state so that the EDPs can be rearmed.

All CPH operation sequences shall be finalized by an operation which changes the state into Monitoring (e.g. Continue, ContinueWithArgument, Connect).

A.2.2.2.8 "Forward" and "Transfer" connection view behaviour principles

The connection view for Forward and Transfer state behaves differently for the case where an operation is received from Terminating Setup (e.g. Call forward service) or Stable_1_Party (e.g. meet me conference service). Depending on the situation the signalling events received from one party (one leg) may have to be relayed to the other party. The method of processing the signaling event received from parties is outside the scope of CS2.

A.2.2.2.9 Conventions at the IBI interface

A call flag has to be allocated at he IBI interface for those signals which are bidirectional like ReleaseReqInd, DataReqInd. For the signals which are either from the O_BCSM to T_BCSM (like e.g. SetupReqInd) or from T_BCSM to O_BCSM (like e.g. ProgressReqInd) no call flag must be inserted.

The following conventions apply for the call flag at the IBI interface:

The call flag is set to <sender> for the signals generated from the O_BCSM

The call flag is set to <receiver> for the signals generated from the T_BCSM

A.2.2.3 Transition Diagrams for CSCV

The CSCV transition diagram for the IN CS1 subset of IN CS2 is shown in figure A.2-8.

The CSCV transition diagram for IN CS2 is shown in figure A.2-9.

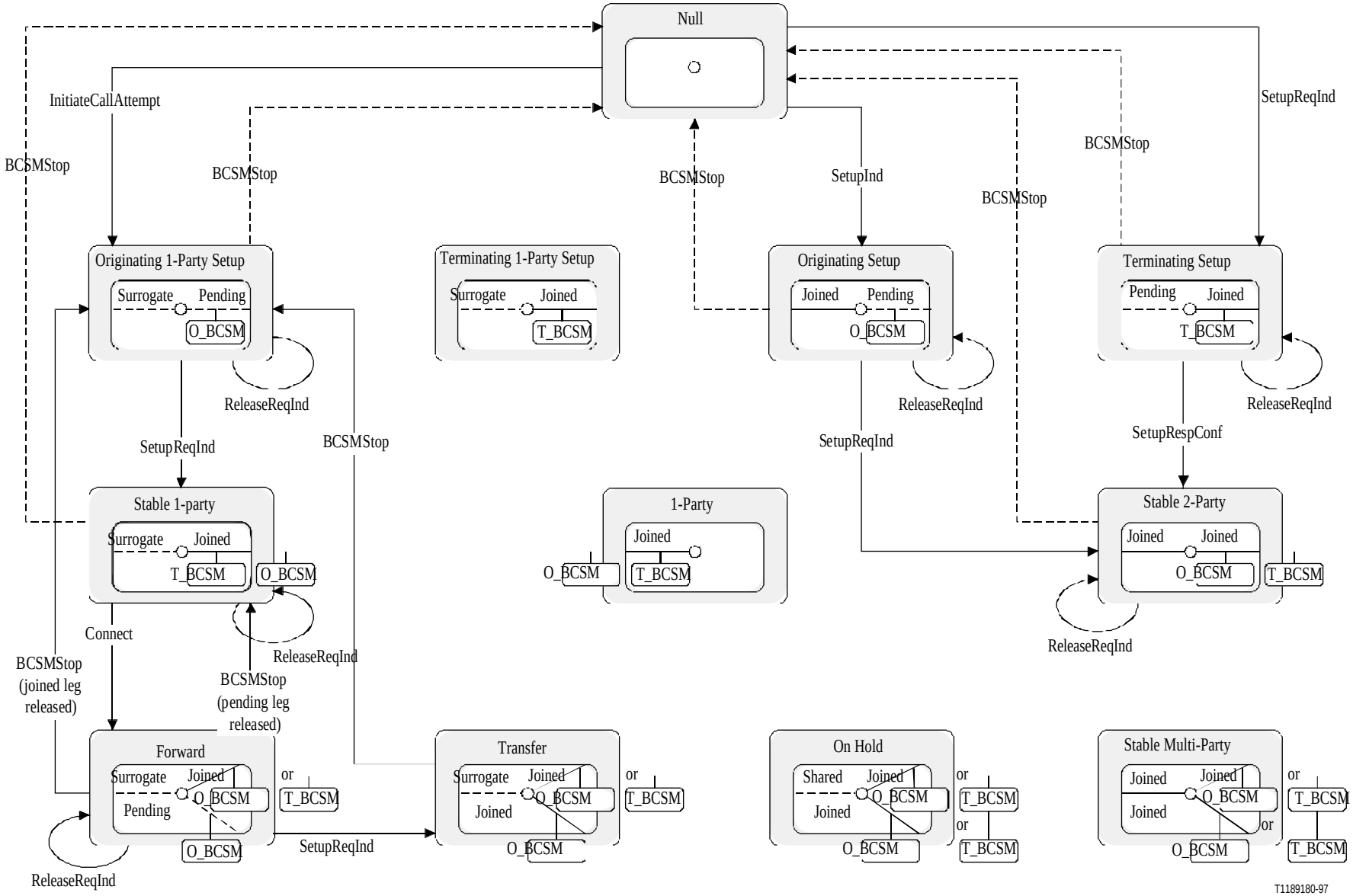


Figure A.2-8: Call Segment (CSCV) Transition diagram for the IN CS1 Subset of IN CS2

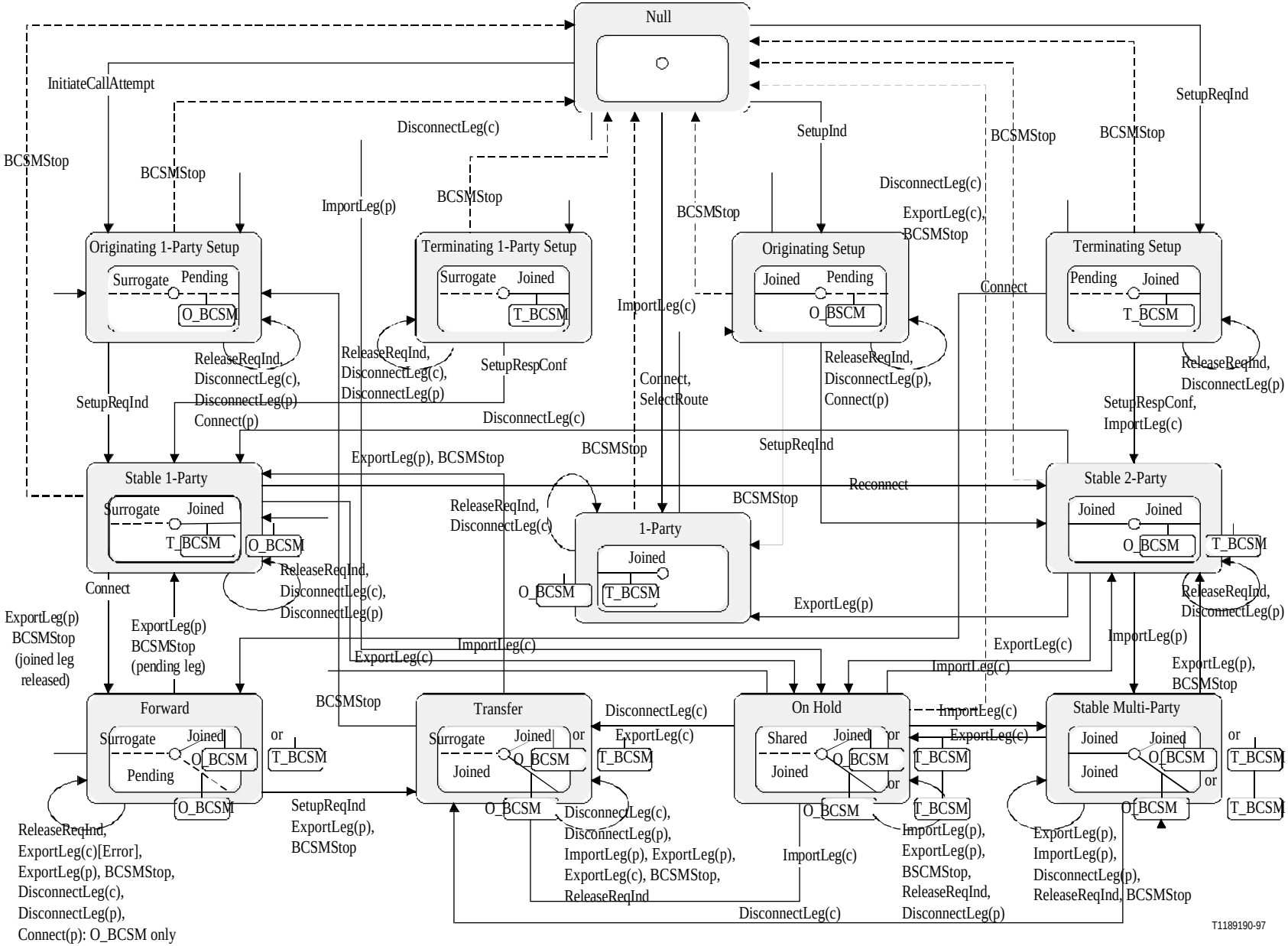


Figure A.2-9: Call Segment (CSCV) transition diagram for IN CS2

A.2.3 SSF_CCF basic generic primitive signal interface model

A.2.3.1 Introduction

A SSF_CCF Basic Primitive Interface Model (BPIM) is provided. The definition of the applied primitive signals is found in the INAP SDL model. The SSF_CCF BPIM allows to describe the applied signalling primitive interfaces and their possible mappings to applied signalling protocols The INAP SDL model consists of two half calls, an originating (SSF_CCF_A) and a terminating (SSF_CCF_B) one. In order operate the model both half calls are needed.

The BCSM is supposed to model existing switch processing of a basic two-party call and should reflect the functional separation between the originating and terminating portions of calls. The SSF-CCF BPIM includes a half call (SSF_CCF_A) with an O-BCSM and a half call (SSF_CCF-B) with a T-BCSM.

In this way the full functionality of the interworking between the O_BCSM and the T_BCSM is catered for.

Since the BCSM is generic, it may describe events that do not apply to certain access arrangements (e.g. analogue signalling systems).

The SSF_CCF Generic Primitive Interface Model supports four different interface types: The SigCon interface to/from NNI/UNI (e.g. ISUP/DSS1, the IBI Interface between half calls and the INAP interface to/from the standardized INAP messages(operations). The signalling control interface is a generic interface that can be mapped to different signalling protocols.

Between the two call halves a switch internal IBI carrying abstract generic primitive signals is applied.

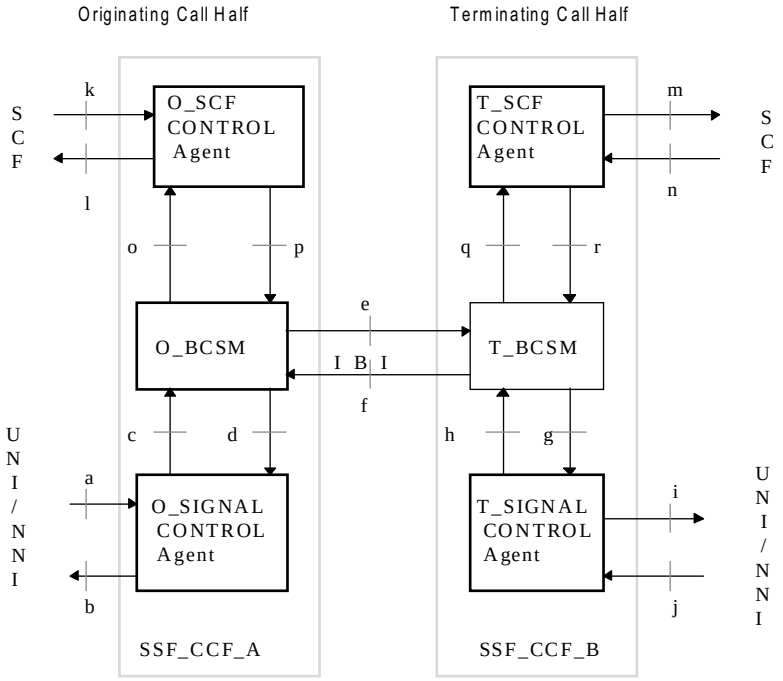


Figure A.2-10: SSF_CCF basic generic primitive signal interface model

A.2.3.2 Primitive signal definitions

The generic primitive signals used in the SDLs are aligned with the information flows described in ITU-T Recommendation Q.71 [17]. The primitive signals defined supporting the UNI/NNI interface are included here in order to ease the readability of the SDLs. For the SCF-SSF primitives supporting the INAP interface are not listed here as they map to corresponding INAP operations defined in clause 17.

A.2.3.3 Description of UNI/NNI Related Primitives

Setup

The Setup primitive is used to request establishment of a call connection. This is a confirmed signal, i.e. a response confirmation Setup primitive is used to confirm that the connection has been established.

The request for establishment of a connection can be originated by either the user or the network (i.e. SCF).

Release

The Release primitive is used to notify that a user has disconnected from the connection and cannot be connected and to request disconnection of a call connection. This is an unconfirmed signal.

SubsequentAddress

The SubsequentAddress primitive is a called number (address) signal for conveying subsequent address information during the digit-by-digit methods of call setup. and for conveying information about last digit received, i.e. address end during the digit-by-digit methods of call setup. This is an unconfirmed signal.

CallProgress

The CallProgress primitive is a signal that is used to report status and/or other types of call information across the network. The type of information is indicated (e.g. "no indication", "alerting" , "remote call hold" etc.). This is an unconfirmed signal.

NetworkSuspend

The NetworkSuspend primitive is a signal used to suspend the call on behalf of the called party upon receipt of an on-hook indication from the terminating line or upon receipt of a network suspend message indication from terminating side. This is an unconfirmed signal.

NetworkResume

NetworkResume. primitive is a signal used to resume the call on behalf of the called party upon receipt of a re-answer indication from the terminating line as the subscriber goes off-hook or upon receipt of a network resume message indication from terminating side. This is an unconfirmed signal.

Failure

Failure primitive is a signal used to report the occurrence of a failure in the network

ServiceFeatureIndication

ServiceFeatureIndication primitive is a signal used to report the occurrence of a service feature activation request from user

ChargingEvent

ChargingEvent primitive is a signal used to report the occurrence of a charging event

A.2.3.4 Primitive Signal Conventions

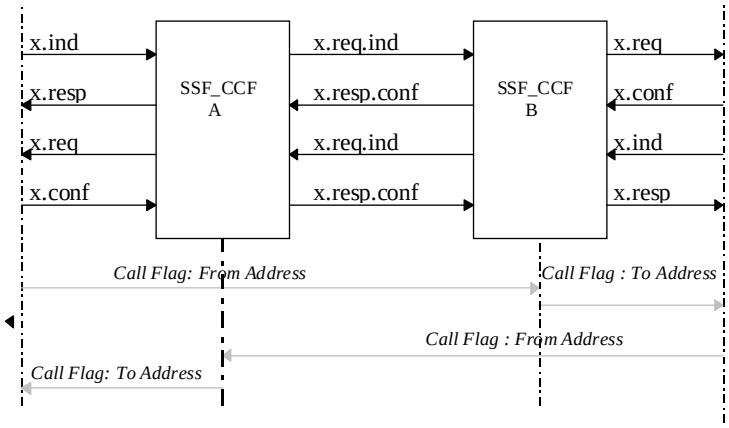


Figure A.2-11: Primitive conventions

Each Primitive signal will as a mandatory parameter include a CallRef parameter consisting of a CallFlag and a CallID (call instance ID). The Call Flag idicates the direction of the primitive signal as indicated in the figure A.2-11.

Primitive signal types:

Confirmed Example: B party answer message received in response to a setup request messages (e.g. ANM, CON in ISUP).

Unconfirmed Example: B party alerted, Alert message sent backward to notify calling party(e.g. ACM (free subscr.) or CPG(Alert) in ISUP).

History

Document history				
V1.1.1	March 1998	Public Enquiry	PE 9829:	1998-03-20 to 1998-07-17