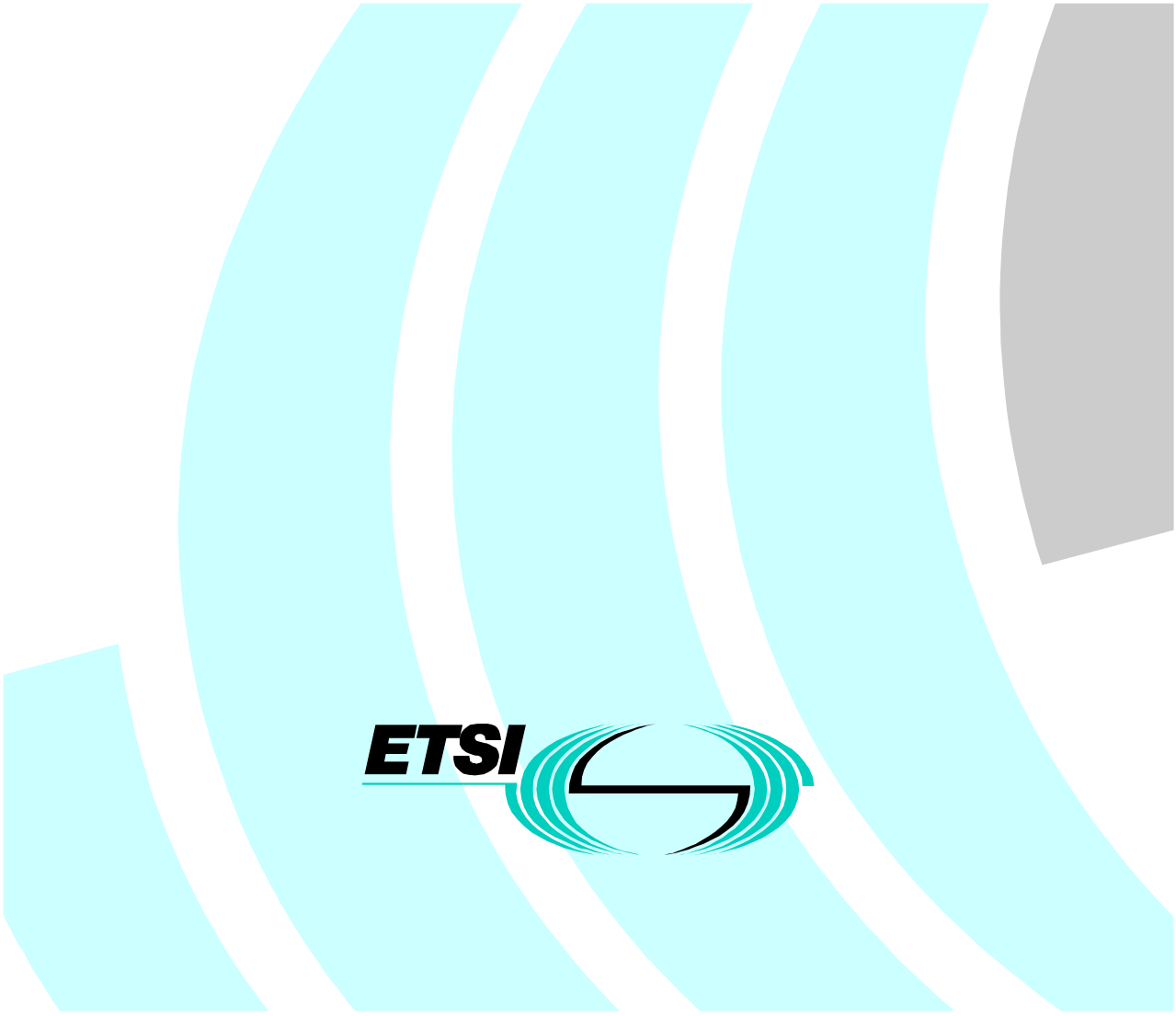
Draft ETSI EN 301 931-1 V1.1.1 (2000-08)

European Standard (Telecommunications series)

Intelligent Network (IN);
Intelligent Network Capability Set 3 (CS3);
Intelligent Network Application Protocol (INAP);
Protocol specification;
Part 1: Common aspects



Reference
DEN/SPAN-03063/1-1

Keywords
CS3, CTM, IN, INAP, Protocol, UPT

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - 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

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 ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI 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 ETSI SR 000 314 (or the updates on the ETSI Web server) 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 Services and Protocol for Advanced Networks (SPAN), and is now submitted for the Public Enquiry phase of the ETSI Standards Two-step Approval Procedure.

The present document describes the common aspects of IN including an overview of each FE. It also describes aspects of the protocol that are common to all interfaces, including services assumed from lower layers, IN security information and the common data types used on all the interfaces.

The present document is part 1 of a multi-part standard covering Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 3 (CS3); Protocol Specification, as described below:

Part 1: **"Common aspects"**;

Part 2: "SCF-SSF interface";

Part 3: "SCF-SRF interface";

Part 4: "SDLs for SCF-SSF interface".

The present document and parts 2 to 4 define the Intelligent Network (IN) Application Protocol (INAP) for IN Capability Set 3 (IN CS-3). The present document and parts 2 to 4 define the INAP for IN CS-3 based upon ETSI Core INAP CS-2 (EN 301 140-1 [11]) and ITU-T IN CS3 Recommendation Q.1238 (1999).

In addition to the features supporting IN CS-1 and IN CS-2 functionalities, the present document and parts 2 to 4 provide:

- general extensions to the CS-2 INAP in support of IN CS-3 target services;
- protocol support for Call Party Handling capabilities;
- protocol support for the Service Control Function (SCF) to SCF and Service Data Function (SDF) to SDF functional relationships to support distributed service logic execution and distributed data functions;
- protocol support for the Call Unrelated Service Function (CUSF) to SCF to support non-call related interactions between users and the SCF;
- additional details on services assumed from lower layers and generic interface security;
- SDLs for SSF related procedure handling based upon ITU-T Recommendation Z.100 [84] object oriented Specification and Description Language (SDL).

The present document and parts 2 to 4 describe the protocol realizing the ITU-T Recommendation Q.1231 [55] Distributed Functional Plane (DFP) in a service and vendor implementation independent manner, as constrained by the capabilities of the embedded base of network technology. This provides the flexibility to allocate distributed functionality into multiple physical network configurations and to evolve IN from IN CS-3 to some future CS-N.

The present document and parts 2 to 4 describe the Intelligent Network modelling of each Functional Entity (FE) and their relationships as defined as part of the DFP. It also defines the INAP (Intelligent Network Application Protocol) required for the communication of functional entities when they are located in different physical entities.

Intelligent Network Capability Set 3 (IN CS-3) supports the following functional entities:

- Service switching function (SSF)
- Service control function (SCF)
- Specialized resource function (SRF)
- Service data function (SDF)
- Call unrelated service function (CUSF)
- Service Management Function (SMF)

The structure of the present document and parts 2 to 4 follows the ITU-T Recommendation Q.1238 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.1238 was not publicly available.

Future ENs 301 932, 301 933 and 301 934 will contain the PICS, the TSS&TP as well as the ATS and partial PIXIT.

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 provides a brief introduction to the modelling of IN CS-3 Functional Entities (FE). It also describes aspects of the protocol that are common to all interfaces. This includes physical examples of interface interconnection, services assumed from lower layers (e.g. TCAP), IN security information and the common data types used on all the interfaces.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

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

[1] ANSI T1.113-1995: "Signalling System No. 7(SS7) – Integrated Services Digital Network (ISDN) User Part".

[2] ETSI 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".

[3] ETSI 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]".

[4] ETSI 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]".

[5] ETSI 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)".

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

[9] ETSI EN 300 356-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; ISDN User Part (ISUP) version 4 for the international interface; Part 1: Basic services [ITU-T Recommendations Q.761 to Q.764 modified]".

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

[11] ETSI ETS 301 140-1: "Intelligent Network (IN); Intelligent Network Capability Set 2 (CS2); Core Intelligent Network Application Protocol (INAP); Part 1: Protocol specification".

[12] ETSI ETS 300 140-5: "Intelligent Network (IN); Intelligent Network Capability Set 2 (CS2); Core Intelligent Network Application Protocol (INAP); Part 5: Distributed Functional Plane (DFP)".

[13] ETSI 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]".

[14] ETSI 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]".

[15] ETSI ES 201 296: "Integrated Services Digital Network (ISDN); Signalling System No.7; ISDN User Part (ISUP); Signalling aspects of Charging".

[16] ETSI TS 124 008: "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Mobile radio interface layer 3 specification; Core Network Protocols - Stage 3 (3G TS 24.008 version 3.3.1 Release 1999)".

[17] ETSI TS 129 002:"Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Mobile Application Part (MAP) specification (3G TS 29.002 version 3.4.0 Release 1999)".

[18] ETSI TS 122 024: "Digital telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Description of Charge Advice Information (CAI) (3G TS 22.024 version 3.0.1 Release 1999)".

[19] ETSI TS 123 078: "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Customized Applications for Mobile network Enhanced Logic (CAMEL) phase 3 - Stage 2 (3G TS 23.078 version 3.3.0 Release 1999)".

[20] ETSI TS 123 040: "Digital cellular etc (GSM); Universal Mobile Telecommunications System (UMTS); Technical realization of the Short Message Service (SMS); (3G TS 23.040 version 3.4.1 Release 1999)".

[21] ETSI TS 123 079: "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Support of Optimal Routeing (SOR); Technical realisation (3G TS 23.079 version 3.3.0 Release 1999)".

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

[23] ISO 639 (1988 Ed. I): "Standardization of methods for creating, compiling and coordinating terminologies - Code for representation of names of languages".

[24] ITU-T Recommendation E.164/I.331 (05/97): "The international public telecommunication numbering plan".

[25] ITU-T Recommendation M.3010 (05/96): "Principles for Telecommunications Management Networks".

[26] ITU-T Recommendation M.3320 (04/97): "Management Requirements Framework for the TMN X-Interface".

[27] ITU-T Recommendation M.3400 (04/97): "TMN management functions".

[28] ITU-T Recommendation Q.29 (11/88): "Causes of noise and ways of reducing noise in telephone exchanges".

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

[30] ITU-T Recommendation Q.700 (11/93): "Introduction to CCITT Signalling System No. 7".

[31] ITU-T Recommendation Q.710 (11/88): "Simplified MTP version for small systems".

[32] ITU-T Recommendation Q.711 (07/96): "Signalling System No. 7 - Signalling Connection Control Part: Service definition".

[33] ITU-T Recommendation Q.713 (07/96): "Signalling System No. 7 -Formats and Codings".

[34] ITU-T Recommendation Q.714 (07/96): "Signalling System No. 7 - Signalling connection control part procedures".

[35] ITU-T Recommendation Q.715 (07/96): "Signalling System No. 7 - SCCP User Guide".

[36] ITU-T Recommendation Q.735 (03/93): "Stage 3 description for community of interest supplementary services using SSNo.7".

[37] ITU-T Recommendation Q.762 (09/97): "Signalling System No. 7 - General Function of Messages and Signals of the ISDN User Part of Signalling System No. 7".

[38] ITU-T Recommendation Q.763: "Signalling System No. 7 - Formats and Codes of the ISDN User Part of Signalling System No. 7".

[39] ITU-T Recommendation Q.765: "Signalling System No. 7 - Application Transport Mechanism".

[40] ITU-T Recommendation Q.767 (02/91): "Application of the ISDN user part of CCITT signalling system No. 7 for international ISDN interconnections".

[41] ITU-T Recommendation Q.771 (06/97): "Signalling System No. 7 - Functional Description of Transaction Capabilities".

[42] ITU-T Recommendation Q.772 (06/97): "Signalling System No. 7 - Transaction capabilities: information elements definitions".

[43] ITU-T Recommendation Q.773 (06/97): "Signalling System No. 7 - Transaction capabilities: messages format and codes".

[44] ITU-T Recommendation Q.774 (06/97): "Signalling System No. 7 - Transaction capabilities: procedures".

[45] ITU-T Recommendation Q.775 (06/97): "Signalling System No. 7 - Guidelines for using Transaction Capabilities".

[46] ITU-T Recommendation Q.822 (04/94): "Stage 1, Stage 2 and Stage 3 description for the Q3 interface - Performance management".

[47] ITU-T Recommendation Q.850 (05/98): "Usage of cause and location in the Digital Subscriber Signalling System No. 1 and the Signalling System No. 7 ISDN User Part".

[48] ITU-T Recommendation Q.931: "Digital Subscriber Signalling System No. 1".

[49] ITU-T Recommendation Q.932 (05/98): "Digital Subscriber Signalling System No. 1 (DSS 1) - Generic Procedures for the Control of ISDN Supplementary Services".

[50] ITU-T Recommendation Q.1208 (09/97): "General aspects of the Intelligent Network Application Protocol".

[51] ITU-T Recommendation Q.1218 (10/95): "Interface recommendation for intelligent network CS-1".

[52] ITU-T Recommendation Q.1224 (09/97): "Distributed functional plane for intelligent network Capability Set 2".

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

[54] ITU-T Recommendation Q.1228 (09/97): "Interface recommendation for intelligent network Capability Set 2".

[55] ITU-T Recommendation Q.1231: "Introduction to Intelligent Network Capability Set-3".

[56] ITU-T Recommendation Q.1236: "Intelligent Network Capability Set 3 Management Information Model Requirements and Methodology".

[57] ITU-T Recommendation Q.1290: "Intelligent Network: Glossary of terms used in the definition of Intelligent Networks".

[58] ITU-T Recommendation Q.1400 (03/93): "Architecture framework for the development of signalling and OA&M protocols using OSI concepts".

[59] ITU-T Recommendation Q.1601: "Signalling System No.7 ISUP Interaction between N-ISDN and INAP CS-2".--ISDN) – Digital subscriber signalling system no.2 (DSS2) – additional traffic parameters".

[60] ITU-T Recommendation X.208: "Specification of Abstract Syntax Notation One (ASN.1)".

[61] ITU-T Recommendation X.209: "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)".

[62] ITU-T Recommendation X.219: "Remote operations: Model, notation and service definition".

[63] ITU-T Recommendation X.229: "Remote operations: Protocol specification".

[64] ITU-T Recommendation X.500 (08/97) | ISO/IEC 9594-1 (1997): "Information technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Services".

[65] ITU-T Recommendation X.501 (08/97) | ISO/IEC 9594-2 (1997): "Information technology - Open Systems Interconnection -The Directory: The Models".

[66] ITU-T Recommendation X.509 (08/97): "The Directory - Authentication Framework".

[67] ITU-T Recommendation X.511 (1997) | ISO/IEC 9594-3 (1997): "Information technology - Open Systems Interconnection -The Directory: Abstract Service Definition".

[68] ITU-T Recommendation X.518 (1997) | ISO/IEC 9594-4 (1997): "Information technology - Open Systems Interconnection -Distributed Procedures".

[69] ITU-T Recommendation X.519 (1997) | ISO/IEC 9594-5 (1997): "Information technology - Open Systems Interconnection -The Directory: Protocol Specifications".

[70] ITU-T Recommendation X.525 (08/97): "Information technology - Open Systems Interconnection - The directory: Replication".

[71] ITU-T Recommendation X.680 (12/97): "Information technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[72] ITU-T Recommendation X.681 (12/97): "Information technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Information object specification".

[73] ITU-T Recommendation X.682 (12/97): "Information technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Constraint specification".

[74] ITU-T Recommendation X.683 (12/97): "Information technology - Open Systems Interconnection -Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications".

[75] ITU-T Recommendation X.690 (12/97): "Information technology - Open Systems Interconnection - Specification of ASN.1 encoding rules: Basic, Canonical, and Distinguished Encoding Rules".

[76] ITU-T Recommendation X.733 (02/92): "Information technology - Open Systems Interconnection - Systems management: Alarm reporting function".

[77] ITU-T Recommendation X.734 (09/92): "Information technology - Open Systems Interconnection - Systems management: Event report management function".

[78] ITU-T Recommendation X.738 (11/93): "Information Technology - Open Systems Interconnection - Systems Management: Summarization Function".

[79] ITU-T Recommendation X.739 (11/93): "Information Technology - Open Systems Interconnection - Systems Management: Metric objects and attributes".

[80] ITU-T Recommendation X.880 (07/94) | ISO/IEC 13712-1 (1994): "Information technology - Remote Operations: Concepts, model and notation".

[81] ITU-T Recommendation X.830 (04/95) | ISO/IEC 11586-1 (1995): "Information technology - Open Systems Interconnection - Generic Upper Layers Security: Overview, Models and Notation".

[82] ITU-T Recommendation X.831 (04/95) | ISO/IEC 11586-2 (1995): "Information technology - Open Systems Interconnection - Generic Upper Layers Security: Security Exchange Service Element (SESE) Service Definition".

[83] ITU-T Recommendation X.832 (04/95) | ISO/IEC 11586-3 (1995): "Information technology - Open Systems Interconnection - Generic Upper Layers Security: Security Exchange Service Element (SESE) Protocol Specification".

[84] ITU-T Recommendation Z.100 (11/99): "CCITT specification and description language (SDL)".

[85] ETSI EN 301 931-2: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification; Part 2: SCF - SSF interface".

[86] ETSI EN 301 931-3: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification Part 3: SCF-SRF interface".

[87] ETSI EN 301 931-4: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification; Part 4: SDLs for SCF-SSF interface".

[88] ITU-T Recommendation Q.1204: "Intelligent network distributed functional plane architecture".

[89] ITU-T Recommendation Q.2931: "Broadband Integrated Services Digital Network (B-ISDN) - Digital Subscriber Signalling System No. 2 (DSS 2) - User-Network Interface (UNI) - Layer 3 specification for basic call/connection control".

[90] ITU-T Recommendation E.164: "The international public telecommunication numbering plan".

[91] ITU-T Recommendation Q.1214: "Distributed functional plane for intelligent network CS-1".

[92] ETSI I-ETS 300 819: "Telecommunications Management Network (TMN); Functional specification of usage metering information management on the Operations System/Network Element (OS/NE) interface".

[93] ITU-T Recommendation Q.2763: "Broadband Integrated Services Digital Network (B-ISDN) - Signalling System No. 7 B-ISDN User Part (B-ISUP) - Formats and codes".

3 Abbreviations

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

AC	Application Context
ACN	Application Context Negotiation
ACSE	Application Control Service Element
AD	Adjunct
ADSI	Analogue Display Service Interface Server
AE	Application Entity
AEI	Application Entity Invocation

AOC	Advice Of Charge
APCI	Application Protocol Control Information
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASE	Application Service Element
ASR	Automatic Speech Recognition
ATS	Abstract Test Suite
BCM	Basic Call Manager
BCP	Basic Call Process
BCSM	Basic Call State Model
BCUP	Basic Call Unrelated Process
BCUSM	Basic Call Unrelated State Model
BGID	Business Group Identity
BNCM	Basic Non Call Manager
BRI	Basic Rate Interface
CAC	Carrier Access Code
CCAF	Call Control Agent Function
CCF	Call Control Function
CDP	Customized Dialling Plan
CHA	Component Handler
CID	Call Instance Data
CM	Call Manager
CMIS	Common Management Information System
CPH	Call Party Handling
CS	Capability Set
CSA	Call Segment Association
CSCV	Call Segment Connection View
CSM	Call Segment Model
CUSF	Call Unrelated Service Function
CUSP	Call Unrelated Service Point
CVS	Connection View State
DAP	Directory Access Protocol
DET	Determination (charging)
DFP	Distributed Functional Plane
DHA	Dialogue Handler
DLE	Destination Local Exchange
DN	Directory Number
DP	Detection Point
DPC	Destination Point Code
DSA	Directory System Agent
DSL	Distributed Service Logic
DSP	Directory System Protocol
DSS 1	Digital Subscriber Signalling No. 1 Protocol
DSS 2	Digital Subscriber Signalling No. 2 Protocol (Broadband)
DTMF	Dual Tone Multi Frequency
DUA	Directory User Agent
EDP	Event Detection Point
EDP-N	Event Detection Point-Notification
EDP-R	Event Detection Point-Request
EUI	Extended User Interface Server
FE	Functional Entity
FEA	Functional Entity Action
FEAM	Functional Entity Access Manager
FIM	Feature Interactions Manager
FPLMTS	Future Public Land Mobile Telecommunications Services
FRL	Facility Restriction Level
FSM	Finite State Machine
GEN	Generation(charging)
GFP	Global Functional Plane
GSL	Global Service Logic

GSS	Generic Security Service
GT	Global Title
GULS	Generic Upper Layer Security
GVNS	Global Virtual Network Services
HLSIB	High Level Service Independent Block
IAF	Intelligent Access Function
ICA	InitiateCallAttempt
IEC	International Electrotechnical Commission
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
INCM	IN Conceptual Model
INDB	IN Data base
INDBMS	IN Data Base Management System
IN-SM	IN Switching Manager
IN-SSM	IN Switching State Model
IP	Intelligent Peripheral
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ISUP	Integrated Services Digital Network-User Part
ITU-T	International Telecommunication Union - Telecommunication Standardization
LE	Local Exchange
MACF	Multiple Association Control Function
MSR	Message Storage and Retrieval
MTP	Message Transfer Part
NAP	Network Access Point
NCM	Non Call Manager
NEF	Network Element Function
NFA	Network Functional Architecture
NM	Network Manager
NSAP	Network Service Access Point
OCCRUI	Out Channel Call Related User Interaction
OCCUUI	Out Channel Call Unrelated User Interaction
OFC	Off-line Charging (billing/accounting information)
OLE	Originating Local Exchange
OLI	Originating Line Information
ONC	On-line Charging (user access information)
OSF	Operator System Function
OSI	Open Systems Interconnection
OUT	Output
PE	Physical Entity
PIC	Point In Call
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol Implementation eXtra Information for Testing
PM	Personal Mobility
POC	Point Of Control
POI	Point Of Initiation
POR	Point Of Return
POS	Point Of Synchronization
PRI	Primary Rate Interface
PSTN	Public Switched Telephony Network
PTNX	Private Telecommunications Network Exchange
RCP	Resource Control Part
RDN	Relative Distinguished Name
REG	Registration(charging)
RFP	Resource Function Part
RLF	Radio Link Function
ROS	Remote Operations
ROSE	Remote Operations Service Element
RPOA	Recognized Private Operating Agency
RRB	RequestReportBCSMEvent

SACF	Single Association Control Function
SAO	Single Association Object
SAP	Service Access Point
SCCP	Signalling Connection Control Part
SCE	Service Creation Environment
SCEF	Service Creation Environment Function
SCEP	Service Creation Environment Point
SCF FSM	Service Control Function Finite State Machine
SCF	Service Control Function
SCFID	Service Control Function Identifier
SCME FSM	Service Control Function Management Entity Finite State Machine
SCME	Service Control Function Management Entity
SCP	Service Control Point
SCSM	Service Control Function Call State Model.
SDF FSM	Service Data Function Finite State Machine
SDF	Service Data Function
SDL	Specification and Description Language
SDME	Service Data Function Management Entity
SDP	Service Data Point
SDSM	Service Data Function Call State Model
SDSS	Server Display and Script Services
SESE	Security Exchange Service Element
SF	Service Feature
SIB	Service Independent Building Block
SL	Service Logic
SLCP	Service Logic Control Program
SLEE	Service Logic Execution Environment
SLEM	Service Logic Execution Manager
SLMP	Service Logic Management Program
SLP	Service Logic Processing Program
SLPI	Service Logic Processing Program Instance
SM	Service Manager
SMAF	Service Management Access Function
SMF	Service Management Function
SMP	Service Management Point
SMS	Service Management System
SN	Service Node
SRF FSM	Specialized Resource Function Finite State Machine
SRF	Specialized Resource Function
SRME	Specialized Resource Function Management Entity
SRSM	Specialized Resource Function Call State Model
SS	Service Subscriber
SS7	Signalling System No. 7
SSCP	Service Switching and Control Point
SSD	Service Support Data
SSF FSM	Service Switching Function Finite State Machine
SSF	Service Switching Function
SSME FSM	Service Switching Function Management Entity Finite State Machine
SSME	Service Switching Function Management Entity
SSN	SCCP Subsystem Number
SSP	Service Switching Point
STI	Service Trigger Information
STUI	Service To User Information
SU	Service User
TC	Transaction Capabilities
TCAP	Transaction Capabilities Application Part
TDP	Trigger Detection Point
TDP-N	Trigger Detection Point - Notification
TDP-R	Trigger Detection Point-Request
TMN	Telecommunication Management Network

TSS&TP	Test Suite Structure & Test Purposes
TTS	Text to Speech Synthesis
UPT	Universal Personal Telecommunication
USI	User Service Information
UTSI	User To Service Information
VPN	Virtual private Network

4 Scope of IN Distributed Functional Plane for capability set 3

The scope of the IN Distributed Functional Plane (DFP) architecture and supporting protocol for IN capability set 3 (IN CS-3) is driven by the requirements of the services desired for IN CS-3, and constrained by the capabilities of the embedded base of network technology. The functionality required to support IN CS-3 services includes functionality to provide:

– end user access to call/service processing;

– call-related service invocation and control;

– end user interaction with service control;

– service management;

– Call Party Handling;

– Internetworking;

– Security;

– Out-Channel User Interaction;

– call unrelated service invocation and control; and

– Feature Interactions.

Each of these aspects is addressed below.

4.1 End user access

End user access to call/service processing is provided via the following access arrangements:

NOTE: This does not preclude the use of these interfaces to support access from private or mobile networks.

- analogue line interfaces;
- ISDN BRI and PRI; and
- traditional trunk and SS No. 7 interfaces.

4.2 Call related service invocation and control

Call/service processing builds upon the current call processing infrastructure of existing digital exchanges. It does so by using a generic model of existing call control functions to process basic two-party calls, then adding service switching functions to invoke and manage IN service logic. This generic model is known as the Basic Call State Model (BCSM). Once invoked, IN service logic is executed under the control of service control functions, in conjunction with service data functions. With this distributed approach to call/service processing, the existing call control function retains ultimate responsibility for the integrity of calls, as well as for the control of call processing resources. The following call/service processing constraints apply:

a) Call control and service switching functionality are tightly coupled, thus the relationship between SSF and CCF is not standardized.

b) A call is either between two or more end users that are external to the network and addressable via a directory number or combination of directory number and bearer capability, or a call is between one or more end users and the network itself.

c) A call may be initiated by an end user, or by an SCF within the network on behalf of an end user. To supplement a call, IN service logic may either be invoked by an end user served by an IN exchange, or by the network on behalf of an end user.

d) A call may span multiple exchanges. As such, each exchange only controls the portion of the call in that exchange – call processing is functionally separated between exchanges. IN service logic invoked on IN exchanges in such inter-exchange calls are managed independently by each IN exchange.

e) Existing exchanges can be viewed as having two functionally separate sets of call processing logic that co-ordinate call processing activities to create and maintain a basic two-party call. This functional separation is provided between the originating portion of the call and the terminating portion of the call. This functional separation should be maintained in an IN exchange to allow IN service logic invoked on the originating portion of the call (i.e. on behalf of the calling party) to be managed independently of IN service logic invoked on the terminating portion of the call (i.e. on behalf of the called party).

f) It is desirable to allow multiple IN-supported service logic instances to be simultaneously active for a given end user. It is also recognized that non-IN service logic will continue to exist in the network. As such, service feature logic instances mechanisms should:

- determine which service logic to invoke for a given service request. This mechanism should select the appropriate IN-supported service logic or non-IN-supported service logic, and block the invocation of any other service logic for that particular service request;
- ensure that simultaneously active IN-supported service logic instances adhere to the single-ended restriction on service processing.

g) The distributed approach and added complexity of call/service processing requires mechanisms for fault detection and recovery, allowing graceful termination of calls and appropriate treatments for end users.

4.3 End user interaction

End user interaction with the network to send and receive information is provided by service switching and call control resources, augmented by specialized resources. These specialized resources are controlled by service control functionality, and are connected to end users via call control and service switching functionality.

4.4 IN service management functionality

IN service management functionality is used to provision and manage the service control functionality, service data functionality, specialized resource functionality and the combined service switching/call control functionality in the network, outside of the context of call/service processing. The IN management functions are modelled according to the TMN functional architecture.

ITU-T Recommendation M.3010 [25] specifies the functional information and physical architectures that support the TMN management services. A TMN management function is the smallest part of the TMN management service as perceived by the user of the service. It will generally consist of a sequence of actions on a defined managed object or objects. The TMN management functions are described in ITU-T Recommendation M.3400 [27].

Many of the IN management functions are not IN specific but can be mapped onto TMN management functions and reused for IN purposes.

Furthermore, the TMN uses OSI System Management Functions. These functions provide generic management controls and capabilities which can be used by specific TMN management services and functions. As a consequence of what is described above, a set of managed objects which are defined in the different system management functions, can be identified and reused for IN purposes.

In particular one can point out: "Summarization Function" (ITU-T Recommendation X.738 [78]), Workload Monitoring Function" (ITU-T Recommendation X.739 [79]), "Event Management Function" (ITU-T Recommendation X.734 [77]), "Alarm Reporting Function" (ITU-T Recommendation X.733 [76]) and "Performance Management Function" (ITU-T Recommendation Q.822 [46]).

4.5 Call Party Handling

Call Party Handling (CPH) is the ability to manage various parties involvement in a call. CPH capabilities rely on an abstract model of call and connection processing activities, maintained by the SSF, in terms of call and connection states. They provide IN services with the ability to add, delete, join and/or separate parties bearer channels from the other parties involved in a call.

4.6 Internetworking

This capability set identifies the SCF - SDF, SCF-SCF, SMF-SMF and SDF-SDF relationships for internetworking purposes. Distributed service logic, but not distributed service control, is supported. Additionally, internetwork management interactions and distributed data handling processes are supported.

4.7 Security

Security is supported through the provision of a number of security assisting functions. A security assisting function requires the co-operation of two or more functional entities. The security assisting functions support both internal network operations and interworking between two or more networks. Security features do not automatically guarantee network integrity, but rather they provide some of the tools in order to allow secured systems to be built.

4.8 Out-Channel User Interaction

The "Out-Channel Call Related User Interaction" (OCCRUI) and the "Out Channel Call Unrelated User Interaction" (OCCUUI) network aspects are used to provide IN services with the ability to request the transparent transfer of information between a User and a Service Logic.

The term "user" refers to any application process which may reside in physical entity connected to the network (e.g. an ISDN terminal, a SCP or any equipment which may be connected to the network through a PRI or trunk line).

4.9 Call unrelated service invocation and control

This capability enables IN-supported service logic instances to be invoked outside the context of a call. It may be required for services/service features such as message waiting indication and registration of the user location.

Call unrelated service processing builds upon the current call unrelated processing infrastructure of existing digital exchanges. It does so by using a generic model of existing call unrelated activities, then adding call unrelated service functions to invoke and manage IN service logic. This generic model is known as the Basic Call Unrelated State Model (BCUSM).

Call unrelated processing and call unrelated service functionality are tightly coupled, thus the relationship between CUSF and CCF is not standardized.

4.10 Feature Interactions

Feature Interaction mechanisms are means to manage feature/service interference situations which have been recognized prior to service deployment. Use of these mechanisms is a network operator option.

4.11 Distributed Functional Model

Figure 1 identifies the IN DFP model for IN CS-3. This figure is provided here as a reference, and is used to show the relevant interfaces on which the protocol in the present document Part 1 is provided. For detailed descriptions of the functional entities, one should refer to ITU-T Recommendation Q.1231 [55].

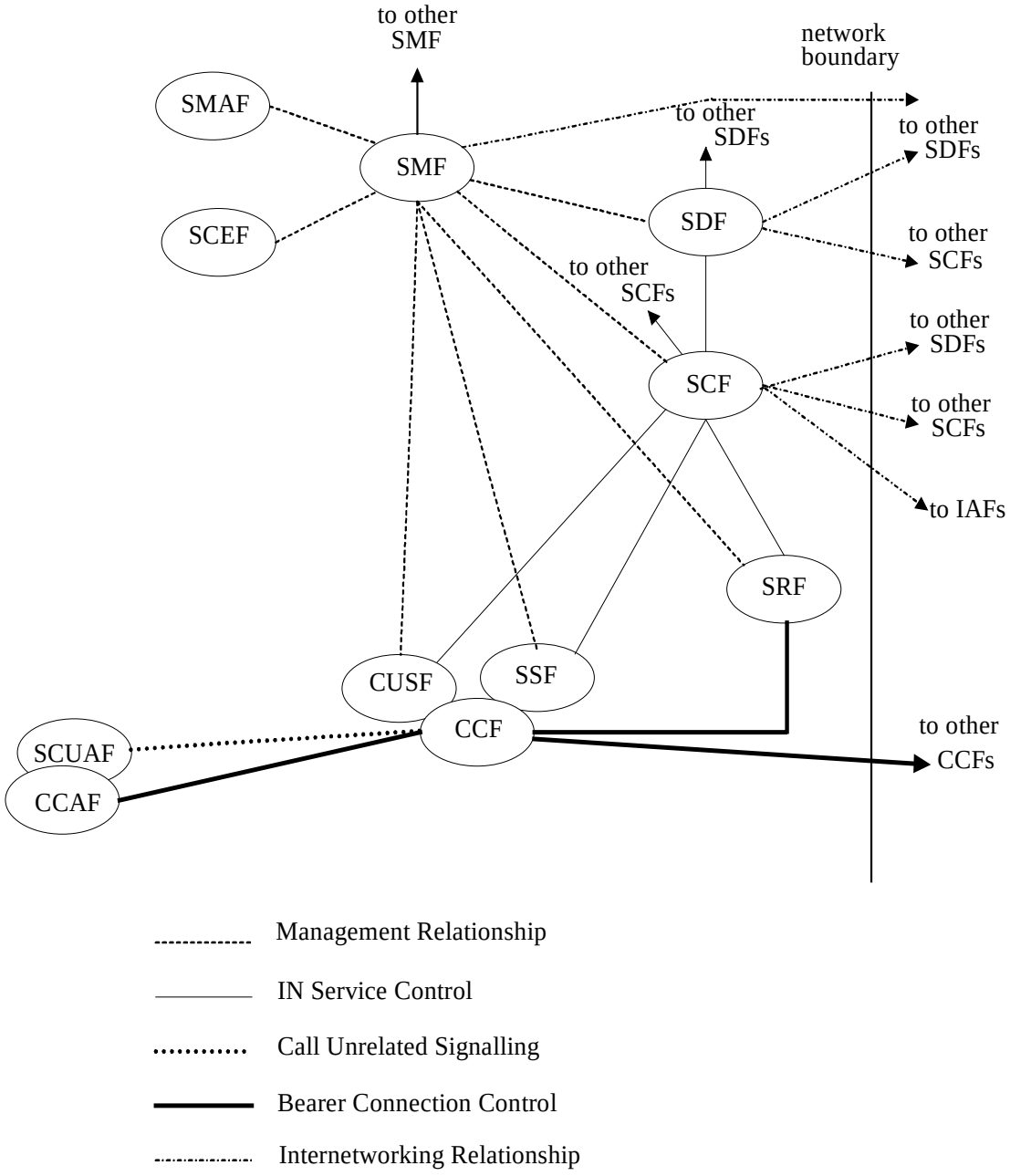


Figure 1: IN CS-3 DFP Architecture

4.12 Communication model

For the purpose of communication, each Functional Entity (FE) includes a Functional Entity Access Manager (FEAM) which provides the necessary functionality to support the exchange of information with other FEs. This includes:

a) establishing and maintaining the interfaces to other FEs using the appropriate underlying communication mechanism;

b) passing and queuing (when necessary) the messages received from other FEs to other components of the FE; and

c) formatting, queuing (when necessary), and sending to other FEs the messages received from other components of the FE.

On the following IN interfaces, interactions between a pair of communicating FEs are defined and specified using the Remote Operations (ROS) paradigm defined in ITU-T Recommendation X.880 [80] .

In such cases, the FEAM is modelled as a *ROS-Object* whose class definition includes the list of *contracts* used over each of the IN interfaces it supports. Each *contract* is in turn defined as a set of operation *packages*, each of which includes one or more *operations* supporting the elementary interactions between the involved FEs.

The internal behaviour of each FE is modelled using one or more Finite State Machines (FSMs) which are interfaced with the FEAM.

This communication model does not assume any particular physical realization. It remains valid when two FEs are co-located in the same equipment or communicate through a network. At the Physical Plane, when two communicating FEs are located in different physical entities (PE), each contract is realized by an application-context using the Transaction Capabilities (TC) of signalling system No7 or using the ROSE capabilities of the DSS.1 protocol. In addition to TC, the application-context includes a set of Application Service Element (ASE) each of which embodies the knowledge of an operation package. This is further explained in clause 5.

The communication model used on interfaces to the SMF is not subject to standardization.

5 FE Models

This clause provides a general description of each of the functional entities identified in the Distributed Functional Plane. Detailed descriptions of the functional entities are provided in EN 301 931-2 to EN 301 931-4. In case of discrepancy between this clause and these EN subparts, the later take precedence.

5.1 Call Control Function / Service Switching Function (CCF/SSF) Models

5.1.1 General

A model of the CCF/SSF is shown in Figure 2 and Figure 3. Figure 2 shows the CCF/SSF model for a single-ended service logic instance related to a calling or called party. Figure 3 shows the CCF/SSF model for separate single-ended service logic instances related to the calling and called parties on the same call. The functional separation serves to isolate single-ended service logic instances related to the calling party from single-ended service logic instances related to the called party for the same call. The purpose of the models is to provide a framework for the understanding of call modelling with respect to the CCF/SSF.

5.1.2 CCF/SSF Components

The CCF/SSF model consists of several entities. Each of the entities are briefly described below, with additional details in EN 301 931-2. It is noted that this shows a conceptual model of CCF/SSF and is not intended to imply an actual implementation of the CCF/SSF.

5.1.2.1 Basic Call Manager (BCM)

The BCM is not a functional entity. It provides an abstraction of the part of a switch which implements basic call and connection control to establish communication paths for users, and to interconnect such communication paths. It detects basic call and connection control events that can lead to the invocation of IN service logic instances or should be reported to active IN service logic instances, and manages CCF/SSF resources required to support basic call and connection control.

The BCM also implements the BCSM and the DP processing logic.

The DP processing logic is the entity of the BCM that interacts with the FIM/CM as described in the FIM/CM description below.

5.1.2.2 Feature Interaction Manager/Call Manager (FIM/CM*)*

This entity in the SSF provides mechanisms to support multiple concurrent instances of IN and non-IN service logic instances on a single call. In particular, the FIM/CM can prevent multiple instances of IN and non-IN service logic instances from being invoked. As well, there is no functionality in the SSF for handling service feature interactions between the separate SSF calling party processes and SSF called party processes in this capability set. The FIM/CM integrates these interaction mechanisms with the BCM and IN-SM to provide the SSF with a unified view of call/service processing internal to the SSF for a single call.

In this capability set, the FIM/CM supports:

- Prevention of non-IN service logic instance invocation according to the service interactions indicators received from an IN service logic instance. This IN-based – switch-based feature/service interaction provides to an IN service logic the ability to allow/deny or modify switch-based service logic execution via call-related signalling. This mechanism is extensible, such that new service interactions may be covered as required. New service interactions may arise due to the introduction of new basic network services or new IN services.

- Prevention of multiple IN service logic invocations according to the service compatibility indicators which are assigned, via administration, to each IN service by the network operator.. This IN-based – IN-based feature/service interaction consist of two different approaches, which may be applied either singly or in combination.

a) SSF-and/or single SCF-oriented approach:

The triggering mechanisms are used and enhanced to avoid feature interference situations. Trigger precedence rules and service compatibility checks are already in place, which may be used to avoid feature interference. During triggering, the FIM performs a generic compatibility check procedure using an administered exclusion matrix for these indicators.

For each triggering of an IN Service during a call the according indicator is derived and shall be propagated through the network. For the case a service compatibility indicator was already received from the network, the new indicator is added to the received one. The ISDN User Part (ISUP) is required to convey this indication.

When received by the IN Service Logic, the ability to overwrite it is a network operator specific option.

b) Two SCF-oriented approach:

This approach is based on the exchange of information between the two involved SCFs (bi-directional), using the OCCRUI mechanism.

In case such mechanisms are used across network boundaries, the relevant signalling and the relevant procedures need to be standardized or bilaterally agreed.

Global management of feature interactions also requires conveying these indicators via basic network signalling to the originating and terminating instances.

5.1.2.3 IN – Switching Manager (IN-SM)

This entity in the SSF interacts with the SCF in the course of providing IN service features to users. It provides the SCF with an observable view of CCF/SSF call/connection processing activities, and provides the SCF with access to control CCF/SSF capabilities and resources. It manages SSF resources required to support IN service logic instances. The IN-SM interacts with the FIM/CM as described above. It also includes the necessary functionality to exchange information with other functional entities (i.e. the FEAM).

Other aspects shown in Figure 2 are not addressed in this capability set, but are assumed to exist.

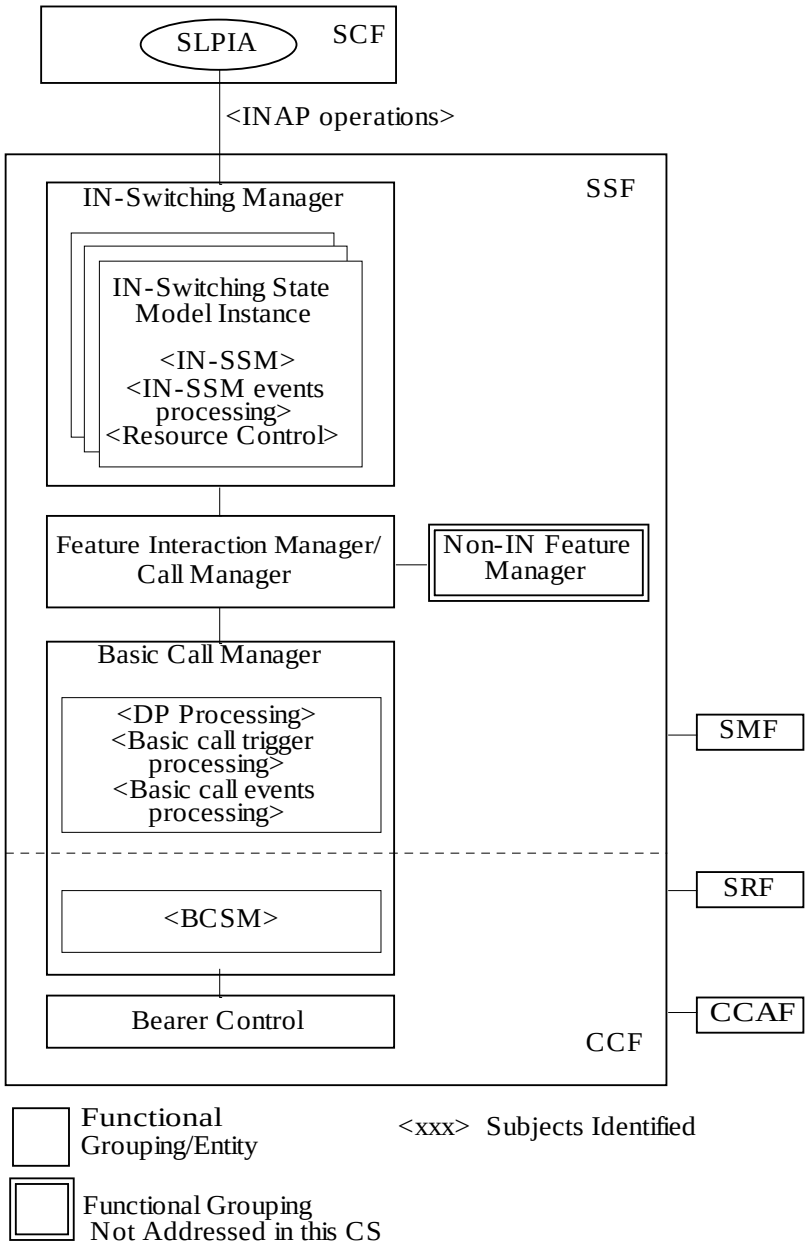


Figure 2: CCF/SSF model - Single-ended SLPI related to calling or called party

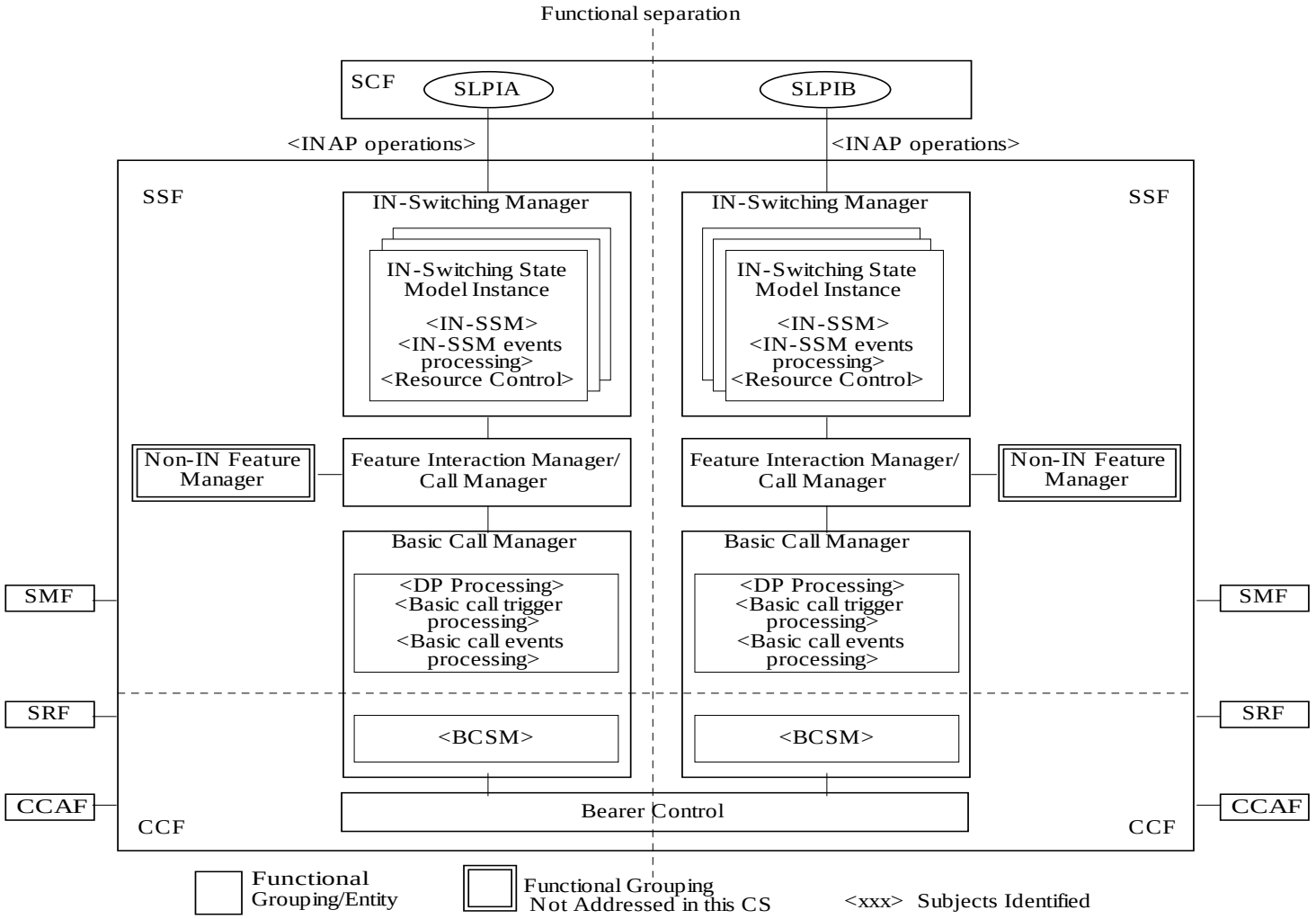


Figure 3: CCF/SSF model - Separated single-ended SLPIs related to calling and called parties

5.1.3 CCF/SSF Trigger Information Objects

This subclause provides an overview of the trigger information objects required for the CCF/SSF. Detailed descriptions of the trigger information can be found in EN 301 931-2.

The entities/components that are involved in triggering include:

a) the DP processing entity in the SSF;

b) the Feature Interactions Manager/Call Manager in the SSF; and

c) the IN-Switching Manager entity in the SSF.

The entities/components shall collectively:

a) perform the DP processing actions specified in EN 301 931-2;

b) check if traffic mechanisms are active;

c) check for SCF accessibility;

d) handle service feature interactions.

They shall hand back control to the CCF for at least the following cases:

a) if call gapping is in effect: the SSF logic instructs the CCF to continue the call processing with the appropriate treatment;

b) if service filtering is in effect: the call is counted (if required) and the SSF logic instructs the CCF to continue the call processing with the appropriate treatment;

c) if a trigger criteria match is not found (e.g., insufficient information to proceed): the SSF logic returns call control to the CCF;

d) if the call is abandoned: the SSF logic returns call control to the CCF and continues processing;

e) if the destination SCF is not accessible: the SSF logic instructs the CCF to route the call if possible (e.g. default routing to a terminating announcement);

f) if there is an existing control relationship for the call and a DP is encountered which is armed as an TDP-R: the SSF returns call control to the CCF in case Multiple Point of Control (MPC) is not supported.

Triggering relies on information stored in the trigger tables in the form of a set of trigger items. For each item, the trigger information model includes:

a) Trigger mode (i.e. an indication on whether triggering should cause call processing suspension);

b) Detection Point in BCSM;

c) Trigger criteria;

d) Information required for SCF addressing;

e) Service key;

f) Application Context Name;

g) Fault handling (to indicate call handling on an error being encountered):

- complete call as dialled;
- release call;
- play specified announcement and release call;
- route call to specified alternate destination.

h) Association to CCF/SSF user/network resource;

i) Service Compatibility Indicators;

j) Trigger precedence.

The following Configuration Management Functions are possible:

- Trigger Table Configuration:

The following capabilities should be possible on the CCF/SSF:

1) Introduction of new trigger data along with the corresponding service key and relevant information.

2) Remove a trigger data identified by its service key.

3) Activate or deactivate existing trigger data within the CCF/SSF.

The SSF should be able to predict at service deployment time unwanted service interactions. If this occurs, new trigger data should be disallowed and an error handling mechanism invoked with the SMF.

- Configuration verification:

The SMF should be able to determine the current configuration of the CCF/SSF at any time. This may be accomplished by having the SMF request the CCF/SSF to provide to the SMF a copy of the trigger information for an CCF/SSF user resource.

5.1.4 SSF FSM Structure

The SSF FSM provides the SCF an observable view of the current state of the SSF involved in a SSF/SCF relationship. The SSF FSM is a finite state machine that is part of the IN SSM, instantiated by the IN-SM. The state of the SSF FSM depends on information exchanged between the SSF FSM and the SCF via the FEAM.

NOTE: One of the three possible FSM's (either IN SSM; Assisting SSF or Handed-off SSF) is selected

Figure 4 shows the SSF FSM structure. The interfaces shown in this figure are internal, and are not subject to standardization.

As shown in Figure 4, an instance of an SSF FSM is either:

a) an IN Switching State Model (IN SSM) FSM;

b) an Assisting SSF FSM;

c) a Handed-off SSF FSM.

The relationship between the BCSM and the SSF FSM is described as follows:

a) 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. If a DP is armed as a Trigger DP (TDP) an instance of an SSF FSM is required.

b) If an InitiateCallAttempt is received from the SCF outside the context of an existing relationship, an instance of a BCSM is created, as well as an instance of an SSF FSM.

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 to the various SSF FSMs and SSME FSMs, and the Functional Entity Access Manager (FEAM).

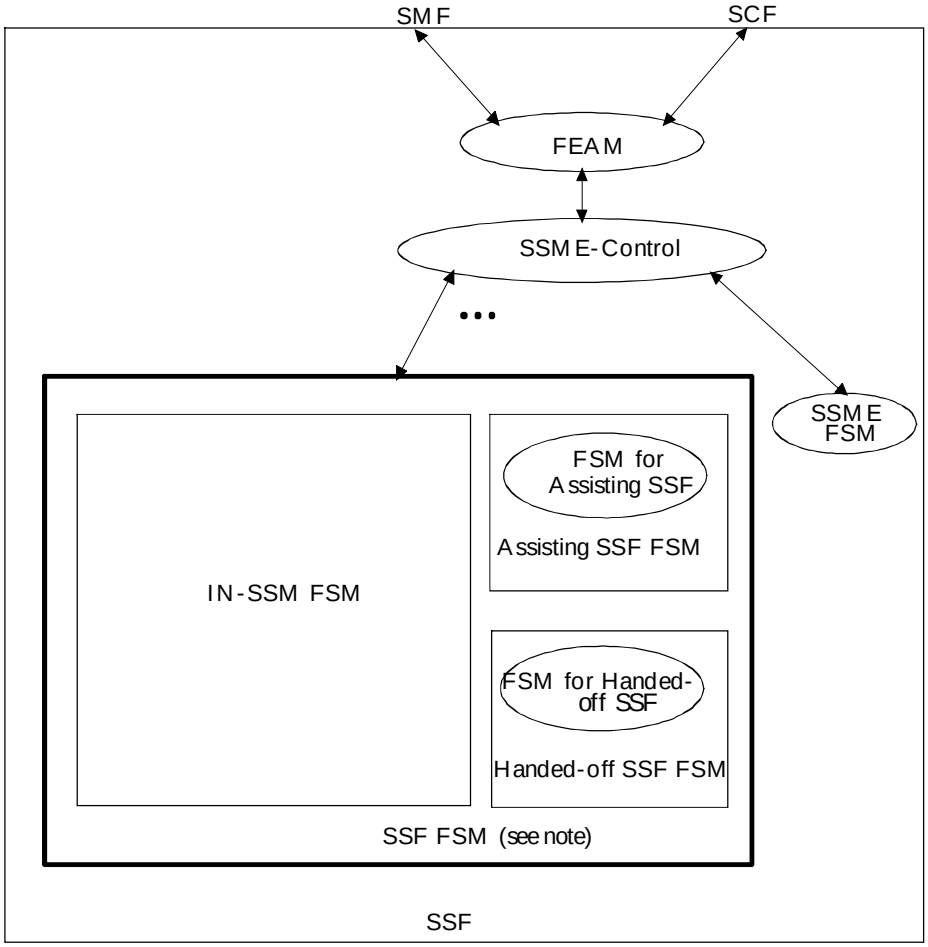
The FEAM provides the lower level maintenance functions on the interface as described in clause 5.

The SSME-control maintains the dialogues with the SCF, and SRF on behalf of all instances of the SSF Finite State Model (FSM). These instances of the SSF FSM occur concurrently and asynchronously as calls occur, which is why there is only 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:

a) Interprets the input messages from other FEs and translates them into corresponding SSF FSM events.

b) Translates the SSF FSM outputs into corresponding messages to other FEs.

c) 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 non-call associated 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.



NOTE: One of the three possible FSM's (either IN SSM; Assisting SSF or Handed-off SSF) is selected

Figure 4: SSF FSM Structure

5.2 Specialized Resource Function (SRF) Models

5.2.1 General

A model of the SRF is shown in Figure 5. The purpose of this model is to provide a framework for the understanding specialized resource functions with respect to the SRF.

Management of the SRF is required for placing resources in or out of service, e.g. for provisioning, administration and maintenance purpose. The SRF management is activated by a request from another functional entity, and never takes action by itself.

For call/service processing, the SRF has a logical relationship with the CCF/SSF and the SCF. The SCF controls the connection between the CCF/SSF and the SRF, and sends instructions to the SRF.

As part of the process of formulating a response to the SSF, the SCF may need to enter into a dialogue with a calling or a called party. This could, for example, take the form of a prompt and collect digits sequence. In this case the SCF will instruct the SRF to start a dialogue with a user after setting up a bearer path between the CCF/SSF and the SRF. The dialogue between the SRF and the user allows the SRF to play an announcement, and if appropriate, collect digits. If digits have been collected, the SRF will pass the digit information to the SCF. When the service logic in the SCF does have further need the resources, the SCF requests the CCF/SSF to release the connection to the SRF, and the SRF resource will be released.

5.2.2 SRF Components

This section describes the various components found within an SRF. It is noted that this shows a conceptual model of SRF and is not intended to imply an actual implementation of the SRF.

Main SRF components are:

a) Functional Entity Access Manager (FEAM) as described in clause 5;

b) Resource Control Part (RCP);

c) Resources.

The RCP mainly comprises a block called the SRF Resource Manager along with the Resource Logic Library and the Resource Logic Instances, each of which control two types of resources:

a) Resource Function Part (RFP);

b) Data Part (DP).

Details of the components are described in the following subclauses:

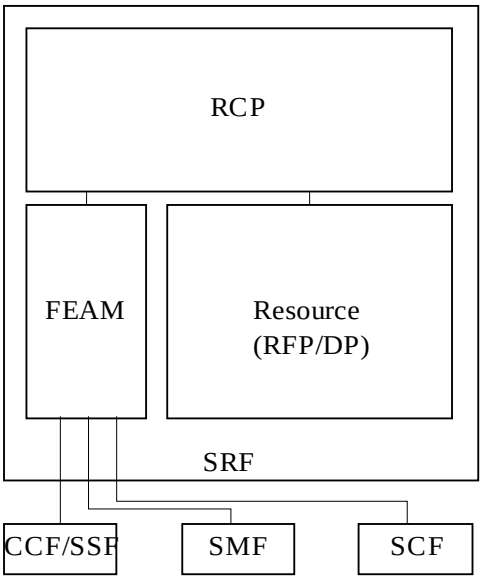


Figure 5: SRF Model

5.2.2.1 Functional Entity Access Manager (FEAM)

The FEAM provides the necessary functionality to exchange information with other functional entities as described in clause 5.

5.2.2.2 Resource Control Part (RCP)

The RCP contains SRF service logic, and controls the service procedure using the capabilities of other blocks. To offer a specialized resource, RCP uses resource-function pair in the RFP and data in the Data Part (DP).

The RCP contains the SRF Resource Manager, the Transaction Module, the User-Interaction script (UI script), the Resource Logic Library and the Resource Logic Instances. Detailed descriptions can be found in EN 301 931-3.

5.2.2.3 Resource Function Part (RFP)

The RFP is a collection of resource-function pairs or functional resource elements. For a service procedure, resources in a resource-function pair are allocated and released together. It should be noted that new resources could be identified as new services/service features are considered.

5.2.2.4 Data Part (DP)

The DP is composed of a database manager and a database containing recorded voice, sound, image, text, etc.

5.2.3 SRF Functional Resources

This subclause provides an overview of the SRF functional resources. Further information can be found in EN 301 931-3.

The following functional resources are supported in this capability set and are defined as follows:

a) DTMF receiver:

This resource receives dual tone multi-frequency (DTMF) from a linked resource.

b) Tone generator/announcements:

This resource provides in-channel information to the specified virtual resource.

c) Message sender/receiver:

This resource sends or receives messages, such as electronic messages, voice messages, etc., to/from users.

d) Synthesized voice/speech recognition device with interactive prompting facilities:

This resource receives in-channel speech information from a linked virtual resource. When the information is input from a user, it is recognized by this resource and this resource converts it to IN operations. When this resource receives an instruction to send a voice message with source-information, it is converted to voice message. Usually, such action is performed with interactive prompting.

e) Text to speech synthesis:

This resource provides the ability to convert a text on an inband information.

f) Modem detection:

This resource provides the ability to detect a modem

g) SDSS capability:

This resource provides the ability to interact with an analogue user by mean of the SDSS (Server Display and Script Service) protocol on the analogue interface (V23).

Examples of other kind of functional resources not supported in this capability set are provided below:

a) Audio conference bridge:

Receiving in-channel audio information from any other linked virtual resources, this resource mixes this information and sends the mixed information to all the linked virtual resources. Another new virtual resource can be joined to or any virtual resources linked to it can be split from this connection resource. It is used as an audio conference bridge.

b) Information distribution bridge:

Receiving in-channel information from a linked virtual resource, this resource distributes the information to all the other linked virtual resources. New virtual resources can be joined to existing connection resource, or existing virtual resources can be split from the connection resource. It is used as a broadcasting device.

c) Protocol converters

5.2.4 SRF FSM Structure

Figure 6 shows the SRF FSM structure. The interfaces shown in this figure are internal, and are not for standardization.

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 to the various SRF call State Models (SRSM) and the functional entity access manager (FEAM).

The model associates a Finite State Machine (FSM) with each initial interaction request from the SCF. Thus, multiple initial requests may be executed concurrently and asynchronously by the SRF, which is why there is only a single entity (SRME) that performs the tasks of creation, invocation, and maintenance of the SRSM objects. 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.

a) Interprets the input messages from other FEs and translates them into corresponding SRSM events; and

b) Translates the SRSM outputs into corresponding messages to other FEs.

Finally, the FEAM provides the lower level maintenance functions on the interface to SSF and SCF, as described in clause 5.

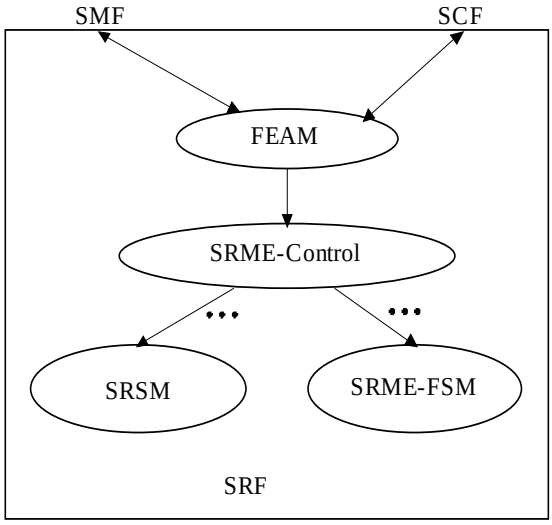


Figure 6: SRF FSM Structure

5.3 Service Control Function (SCF) Models

5.3.1 General

A model of the SCF is shown in Figure 7. The purpose of this model is to provide a framework for the understanding of service logic processing with respect to the SCF.

The prime function of the SCF is the execution of service logic, which is provided in the form of Service Logic Processing programs (SLPs). The SCF also includes the SLP execution supporting functions, such as service logic selection/interaction management, functional entity access management, and SLP provisioning management.

5.3.2 SCF Components

The SCF model providing the functionality defined above is shown in Figure 7. It is noted that this shows a conceptual model of the SCF and is not intended to imply an actual implementation of the SCF.

The SCF platform provides a Service Logic Execution Environment (SLEE) on which a Service Logic Processing program (SLP) runs to provide service processing. An SLP is a service application program invoked by the SLEE and is used to realize service processing under the control of the SLEE. The simultaneous invocation and execution of multiple SLPs are also managed by the SLEE.

Each of the entities shown in Figure 7 will be described in the following subclauses.

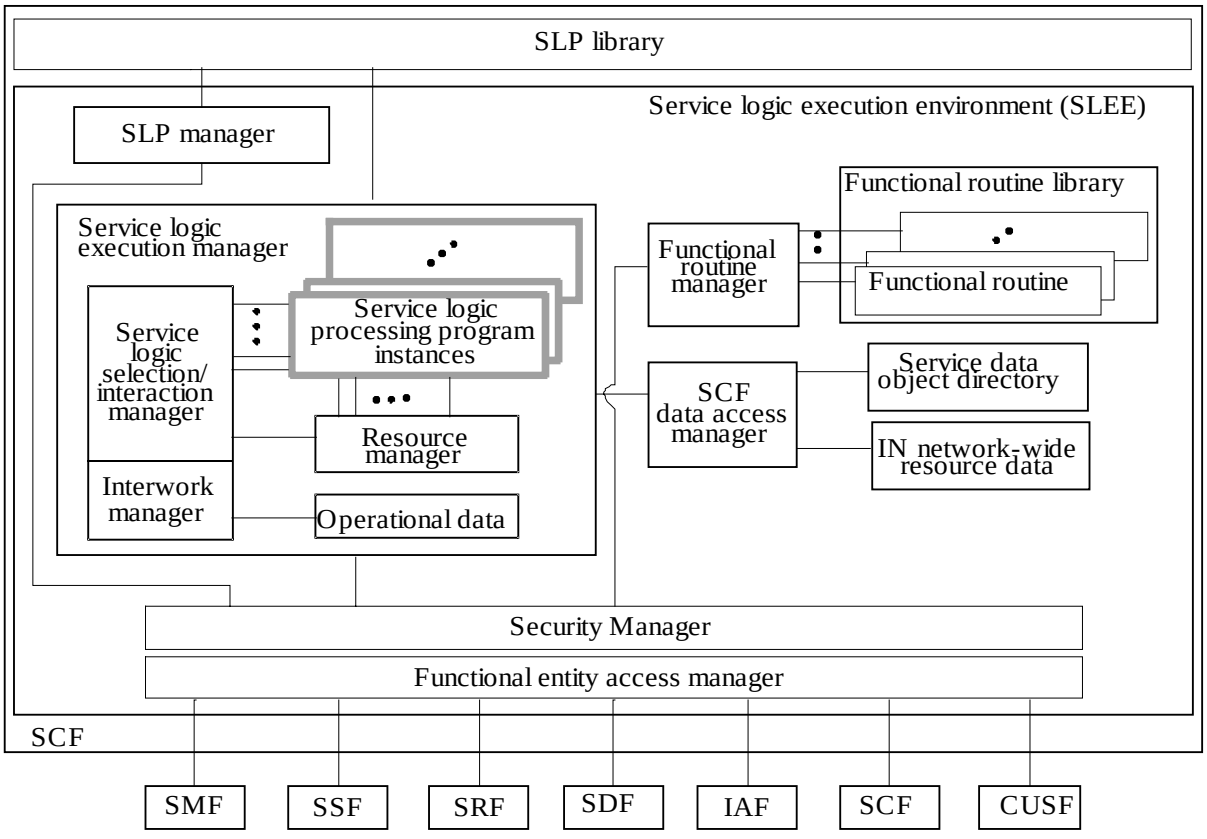


Figure 7: SCF Model

5.3.2.1 Service Logic Execution Manager (SLEM)

5.3.2.1.1 General

The SLEM contains the functions to handle and control the complete service logic execution action. The SLEM also contains service logic processing program instances (SLPIs), service logic selection/interaction manager, and resource manager. It also interacts with SCF data access manager and FEAM to support SLPI execution. In addition to these aspects, the SLEM needs functionality to:

a) execute SLPIs and maintain transient data associated with SLPIs (i.e. information that only persists during the lifetime of the SLPI, such as SLPI state information);

b) execute functional routines in support of SLPI execution;

c) manage SLPI access to SCF and SDF data via the SCF data access manager;

d) manage the exchange of information between SLPIs and entities in other functional entities via the FEAM.

5.3.2.1.2 Service Logic Selection/Interaction Manager (SLSIM)

The SLSIM is the entity that selects an SLP for execution and controls the simultaneous execution and/or execution order of multiple SLPs in the same SCF. This capability set does not define whether or not the SLSIM is explicitly divided into two different entities, i.e. the service logic selection manager and the service logic interaction manager.

As part of the functionality, the SLSIM provides a means to manage service interactions by managing interactions among multiple SLPIs in the same SCF that are simultaneously active on a single call. The relationship between SLSIM and the feature interaction manager/call manager in the CCF/SSF is not defined.

SLP selection is performed via the SLSIM in response to:

a) an external event from another functional entity;

b) the occurrence of internally recognized conditions (e.g. time of day or other internal events); and

c) the execution of a functional routine via an SLPI that requests the execution of another SLP.

In addition, the SLSIM should invoke the execution of the selected SLP and provide for mutual exclusion and precedence during this SLP selection and invocation:

a) mutual exclusion prevents the invocation of an SLP whose execution would be incompatible with a currently executing SLPI;

b) precedence provides a scheme to select a particular SLP from a set of SLPs which meet the same selection criteria.

5.3.2.1.3 Service Logic Processing program Instance (SLPI)

A Service Logic Processing program (SLP) is a service application program invoked by the SLEE and used to realize service processing. It contains logical constructs which, when executed, control the flow of service execution,, and invoke functional routines in the SCF to access network data resources needed for service execution. When an SLP is selected and invoked, it is referred to as a Service Logic Processing program Instance (SLPI). In contrast to an SLP, a corresponding SLPI is a dynamic entity that actively controls the flow of service execution and invokes SCF functional routines.

Functional routines are programs in the SCF that can be invoked by SLPIs to cause a sequence of actions to be performed in the network in support of service execution. This sequence of actions provide the functionality defined for a service independent building block (SIB) on the Global Functional Plane. Therefore, functional routines are considered to be service independent. Potential categories of functional routines are described in the following subclause entitled "Functional Routine Categories".

5.3.2.1.4 Resource manager

The resource manager provides functionality to control the allocation of local SCF resources and provides access to network resources in support of SLPI execution. The resource manager contains functionality to:

a) identify and locate local SCF resources;

b) identify and locate network resources via the SCF data access manager and IN network-wide resource data;

c) identify one or more local SCF resources requested by a particular SLPI;

d) release one or more local SCF resources no longer needed by a particular SLPI; and

e) interact with other functional entities via the FEAM to provide for the reservation and release of network resources to be used by SLPIs.

It should be noted that the selection of an SRF is not always performed by the SLEM resource manager. In some cases selection is performed by an SSF, for example, where assist/hand-off procedures are being used.

5.3.2.1.5 Internetworking manager

Internetworking manager provides the functionality required for internetworking:

a) access control based on the originating entity and the associated service/feature;

b) secure access to the service logic and monitoring of the illegal access;

c) accounting of the interworking activity (if needed).

5.3.2.2 SCF data access manager

5.3.2.2.1 General

The SCF data access manager provides the functionality required to provide for the storage, management, and access of shared and persistent information in the SCF (i.e. information that persists beyond the lifetime of a SLPI). The SCF data access manager also provides the functionality required to access remote information in SDFs. The SCF data access manager interacts with the SLEM to provide these capabilities to SLPIs.

Figure 7 identifies two data structures that contain SCF data. These include:

a) the service data object directory; and

b) the IN network-wide resource data.

These are described in the following subclauses.

5.3.2.2.2 Service data object directory

Figure 7 identifies a service data object directory. The Service data object directory provides a means to address the appropriate SCF for access to a specific data object.

The SLEM interacts with the SCF data access manager to access service data objects in SDFs. The SCF data access manager uses the service data object directory to locate service data objects in the network in a manner transparent to the SLEM (and its SLPI). As such, the SLEM (and its SLPIs) has a global and uniform view of service data objects in the network.

5.3.2.2.3 IN network-wide resource data

This is a data structure, accessible to SLPIs, in which information resides about the location and capabilities of resources in the network. It provides a means to address the appropriate functional entity (e.g. SRF) for access to specific resources with the appropriate capabilities.

The SLEM resource manager interacts with the SCF data access manager to access network resource data. The SLEM resource manager provides SLPIs with access to network resources in a manner transparent to SLPIs. As such, SLPIs have a global and uniform view of resources in the network.

5.3.2.3 Functional routine manager

The Functional Routine Manager is used for receiving and distributing of functional routines to the functional routine library via the FEAM. This entity also manages the addition, deletion and suspension of a particular functional routine. The management of these functional routines by the SMF is not defined.

Functional routine library is an entity where the actual functional routines reside.

5.3.2.4 Functional Entity Access Manager (FEAM)

The FEAM provides the functionality required by the SLEM to exchange information with other functional entities via messages, as described in clause 50.

5.3.2.5 SLP manager

The SLP manager manages the reception and distribution function of SLPs from other entities. The SLP manager, therefore, interworks with the FEAM. This entity also manages addition, deletion and suspension of a particular SLP. The management of the SLP is via the SMF.

5.3.2.6 Security manager

The security manager provides the functionality required for providing secure access to the SCF by other Functional Entities (SCF, SDF). The security manager should:

a) generate security parameters for outgoing messages to provide data confidentiality, data origin authentication, data integrity for a specific interface;

b) verify security parameters for incoming messages to provide data confidentiality, data origin authentication, data integrity for a specific interface. If an incoming message has an invalid security parameter then the message should be rejected and the invalid access attempt logged.

The level of security applied to an interface is determined by which of the security parameter features are active. The selection of active features is performed through interaction with the SMF.

5.3.3 The SCF FSM structure

Figure 8 shows the SCF FSM structure. The interfaces shown in this figure are internal, and are not for standardization In the SCF Call State Model (SCSM), there are four kinds of FSMs: SSF/SRF, SDF, SCF and CUSF. They consist of:

a) The SSF/SRF FSM (SCSM-SSF/SRF) handles the interaction with the SSF/SRF.

b) The SDF FSM (SCSM-SDF) handles the interaction with the SDF FSM.

c) The CUSF FSM (SCSM-CUSF) handles the interaction with the CUSF FSM.

d) The inter-SCF FSM(SCSM-SCF) handles interactions with other SCFs.

The SCF also includes the SCME which handles the interaction between the SCF and the SCF management functions.

The following text provides an overview of the procedural aspects of the interfaces between the SCF and other functional entities.

The relationship between the SLP and the SCF FSM may be described as follows:

a) 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;

b) When initiation of a call is requested from service logic, an instance of the SCSM is created.

In either case, the SCF FSM handles the interaction with the SSF FSM, SRF FSM, SCF FSM, CUSF FSM, and SDF FSM, and notifies the SLP of events as required.

The management functions related to the execution of operations received from the SSF, SRF, SDF, CUSF, or co-operating SCF are executed by the SCF Management Entity (SCME).

The SCME is comprised of the SCME-Control and multiple instances of SCME FSMs. The SCME-Control interfaces the different SCF Call State Models (SCSMs) and the Functional Entity Access Manager (FEAM). These instances of the SCF FSMs occur concurrently and asynchronously as SCF related events occur, which is why there is a 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:

a) Interprets the input messages from other FEs and translates them into corresponding SCSM events;

b) Translates the SCSM outputs into corresponding messages to other FEs;

c) Performs some asynchronous (with call processing) activities (one such activity is flow control) related to management and supervisory functions in the SCF and creates an instance of a SCME FSM. 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. Therefore, the SCME-Control separates the SCSM from these activities by creating instances of SCME FSMs for each context of related operations;

d) Supports persistent interactions between the SCF and other FEs.

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 will access the same SCME FSM handling this specific service filtering instance. ActivateServiceFiltering operations providing different Filtering Criteria cause the invocation of different SCME FSMs.

Finally, the FEAM provides the lower level maintenance functions on the interface as described in clause 5.

NOTE: 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 CCF/SSF manage call queues; however, the technical details of how the CCF/SSF performs queue management is beyond the scope of IN.

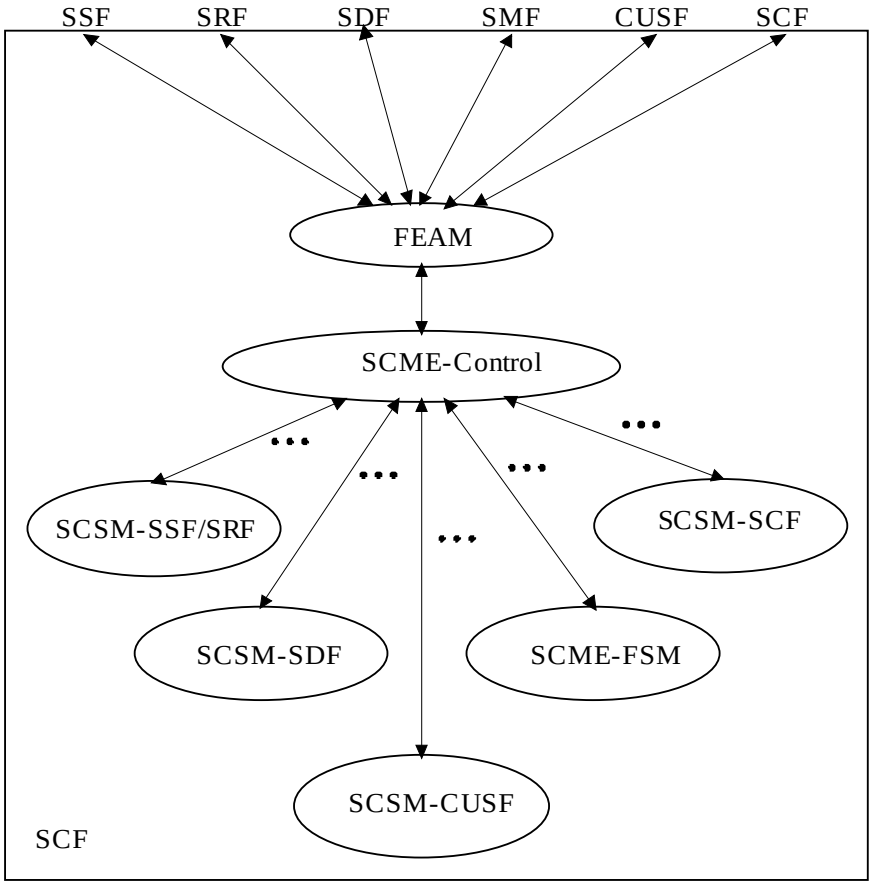


Figure 8: SCF FSM structure

5.4 Service Data Function (SDF) Models

5.4.1 General

A model of the SDF is shown in Figure 9. The purpose of this model is to provide a framework for the understanding of service data functions with respect to the SDF.

The following subclause describes the detailed SDF architecture, and classifies the data types which are handled by the SDF.

The SDF contains and manages data associated with the service logic processing programs (SLPs), and accessed during the execution of the SLP instances (SLPIs). Therefore, data such as SLP selection data and SCF directory, which are accessed before the execution of an SLPI, are not included in the SDF data handling.

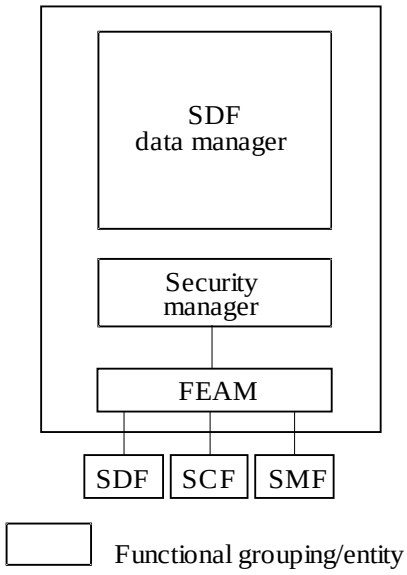


Figure 9: SDF Model

5.4.2 SDF Components

The SDF model to provide the functionality defined above is shown in Figure 9. Each of the functional entities shown in Figure 9 will be described in the following subclauses. It is noted that this shows a conceptual model of SDF and is not intended to imply an actual implementation of the SDF.

5.4.2.1 SDF Data Manager

The SDF data manager provides the functionality required for storing, managing and accessing information in the SDF. If, for example, the data is physically structured as a database, the SDF data manager may also handle the database accessing language such as SQL.

5.4.2.2 Functional Entity Access Manager (FEAM)

The FEAM provides the functionality required by the SDF data manager to exchange information with other functional entities (i.e. SCF, SDF and SMF), as described in clause 5.

5.4.2.3 Security Manager

The Security manager provides secure access to the different types of data held in the SDF, for example, denied access to the data for unauthenticated users. This functionality should:

a) check the access rights of the SCF;

b) authenticate users with provided information;

c) count the failed authentication attempts for a given user (whether the implementation of this function can be realized in the SDF has not been investigated yet);

d) block data access;

e) assign user's access rights;

f) memorize user's access rights during his request;

g) control user's right to access specific data.

The security manager also provides the functionality required for providing secure access to the functions of the SDF by other Functional Entities (SCF, SDF). This functionality should:

- generate security parameters for outgoing messages to provide data confidentiality, data origin authentication, data integrity for a specific interface;
- verify security parameters for incoming messages to provide data confidentiality, data origin authentication, data integrity for a specific interface. If an incoming message has an invalid security parameter then the message should be rejected and the invalid access attempt logged.

The level of security applied to an interface is determined by which of the security features are active. The selection of active features is performed through interaction with the SMF.

5.4.3 Data Types Handled by SDF

Data handled by the SDF can be classified into the following types:

a) Authentication data - This data is used to authenticate a user that accesses the database through an SCF, e.g. a PIN code, the value of a counter for failed authentication. The set of authentication data used is associated to a level of access rights.

b) Operational data - This data is not required by the SLPIs, but is used by the SDF itself for operational and administrative purposes, e.g. references to an object class, access control data.

c) Service data - This data is used for the provision of a service, e.g. a subscriber profile, service provider agreements. If necessary, this data can be used by several services.

5.4.4 SDF FSM Structure

Figure 10 shows the SDF FSM structure. The interfaces shown in this figure are internal, and are not for standardization.

In the SDF Call State Model (SDSM), there are two kinds of FSMs. They consists of:

a) The SCF FSM (SDSM-SCF) handles the interaction with the SCF;

b) The SDF FSM (SDSM-SDF) handles interactions with other SDF.

The SDF also includes the SDME which handles the interaction between the SDF and the SDF management functions.

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 Functional Entity Access Manager (FEAM).

The SDME Control maintains the associations with the SCF and co-operating SDFs on behalf of all instances of the SDF FSMs (e.g. SDSM-SCF, SDSM-SDF). 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:

a) interprets the input messages from other FEs and translates them into corresponding SDF FSM events;

b) translates the SDF FSM outputs into corresponding messages to other FEs;

c) 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.

Finally, the Functional Entity Access Manager (FEAM) provides the lower level maintenance functions on the interface as described in clause 5.

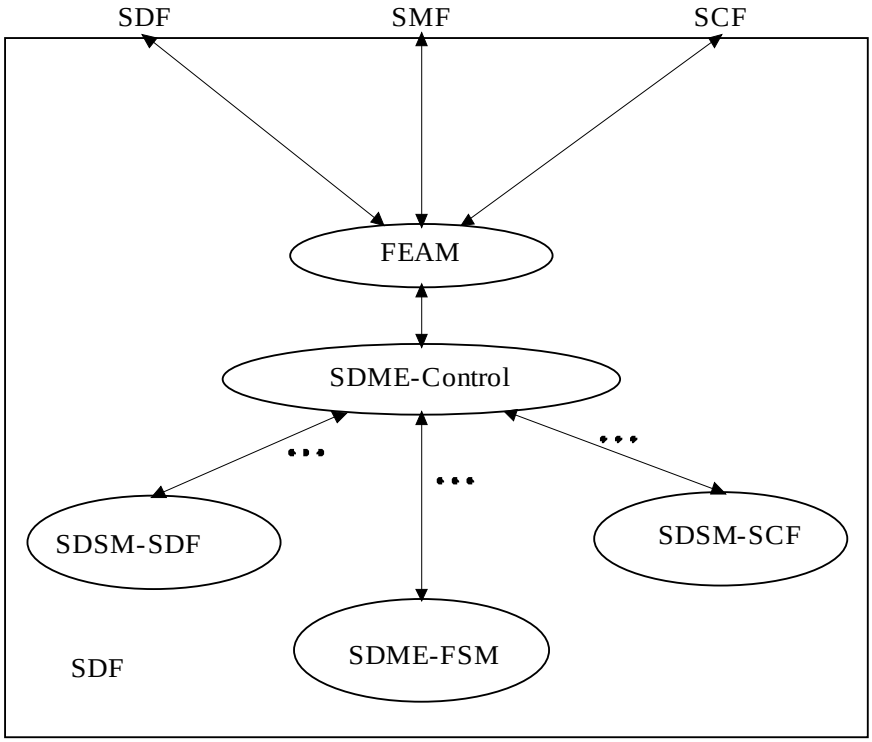


Figure 10: SDF FSM Structure

5.5 Call Control Function/Call Unrelated Service Function (CCF/CUSF) Models

5.5.1 General

The CUSF model shown in Figure 11 is for a single ended service logic instance related to a user. The purpose of this model is to provide a framework for the understanding of call unrelated modelling with respect to the CUSF.

The SCF-CCF/CUSF relationship is used for message between a SCF and a CUSF in the public network. this relationship provides service or service features which need call-unrelated user interaction. This provides, for example, user location registration, user authentication, supplementary service activation or de-activation.

The relationship is used for three cases, and respective cases are exclusive for the others:

Case 1: for sending/receiving ROSE APDUs to/from a user.

Case 2: for sending/receiving USI information to/from a user.

Case 3: for relaying some information for call unrelated association establishment from a supplementary service in CCF (BCUSM).

5.5.2 CCF/CUSF Components

The CCF/CUSF model consists of several entities. Each of the entities are briefly described below, with additional details in EN 301 931-4. It is noted that this shows a conceptual model of CCF/CUSF and is not intended to imply an actual implementation of the CCF/CUSF.

5.5.2.1 Basic Non-Call Manager (BNCM)

This entity provides an abstraction of an association and proceedings for call unrelated interactions between an user and a network. It detects basic call unrelated events that can lead to the invocation of IN service logic instances or those that should be reported to active IN service logic instances. It also manages resources required to support basic call unrelated control. The BNCM comprises the BCUSM and the DP processing logic. The DP processing logic is the entity of the BNCM that interacts with the FIM/NCM as described in the FIM/NCM description below.

5.5.2.2 IN - Non-Switching Manager (IN-NSM)

The entity which interacts with the SCF via Functional Entity Access Manager in the course of providing IN services/features to users. It provides the SCF an observable view of an CUSF call unrelated processing activities, and provides the SCF with access to CUSF capabilities, such as managing of an association and invoked operations. It also detects non-call processing events that should be reported to active IN service logic instances or can lead to the invocation of IN service logic instances. It manages CUSF resources required to support IN service logic instances, if any.

5.5.2.3 Feature Interaction Manager/Non-Call Manager (FIM/NCM)

The entity which provides mechanisms to prevent invocation of multiple instances of IN and non-IN service logic instances on a single association. The ability of the FIM/NCM to arbitrate IN/non-IN call associated/call unrelated associated services/features and IN call unrelated associated services/features is outside the scope of this capability set. The FIM/NCM integrates these interactions mechanisms with the BNCM and IN-NSM to provide the CUSF with a unified view of call unrelated processing internal to the CUSF for an association.

5.5.3 Relationship of CCF/CUSF Model Components

5.5.3.1 BNCM relationship to IN-NSM

This is the relationship that encompassed the interaction between the BNCM and the IN-NSM, through the FIM/NCM. The information flow related to this interaction is not externally visible and is not for standardization. However, an understanding of this subject is required to identify how basic call unrelated processing and IN call unrelated processing interact.

5.5.3.2 BNCM and IN-NSM relationships to FIM/NCM

This is the relationships that encompass the interaction between the BNCM and the FIM/NCM, and the IN-NSM and the FIM/NCM. The information flows related to these interactions are not externally visible and are not for standardization. However, an understanding of this subject is required in order to unify the BNCM, IN-NSM and FIM/NCM.

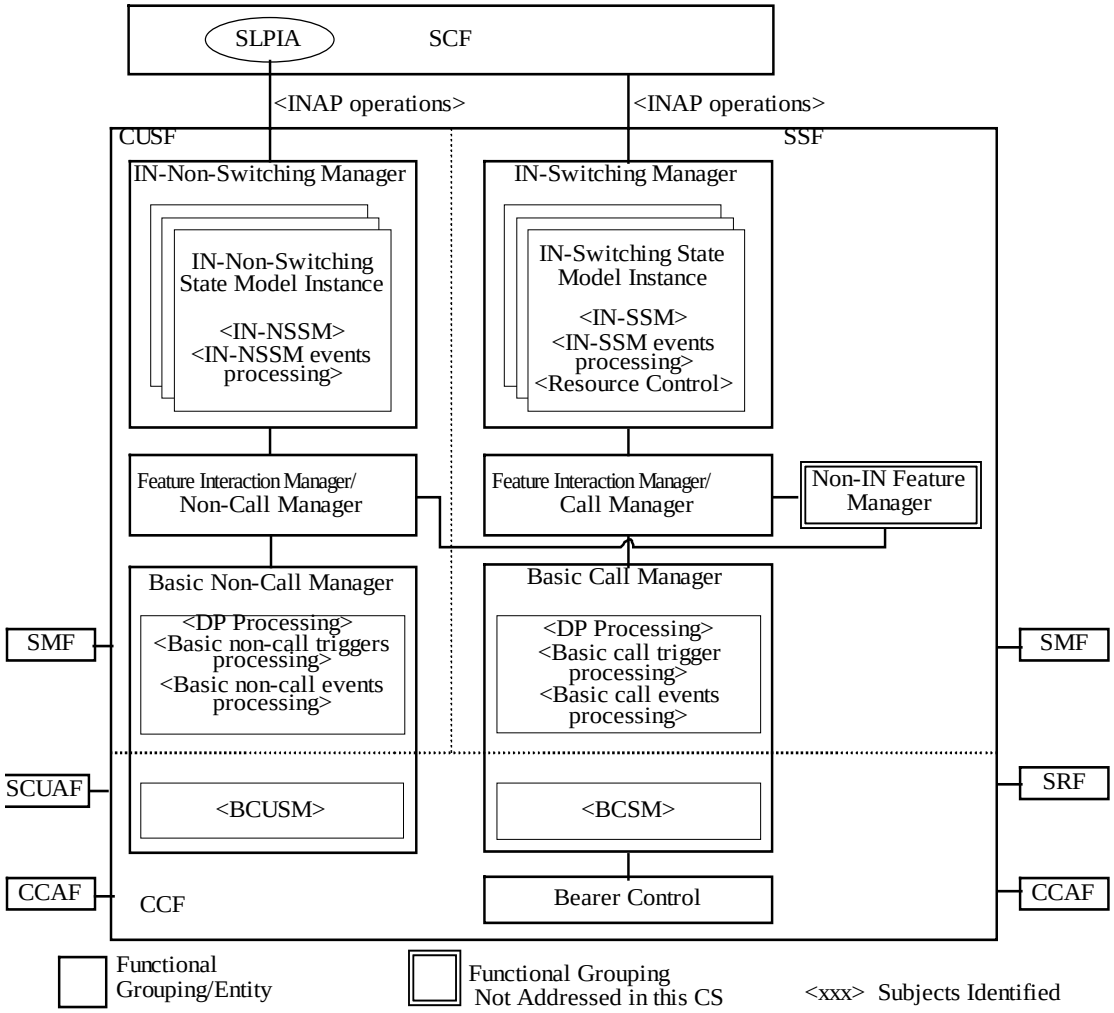


Figure 11: Internal CUSF Structure and the Relationship Between FEs

5.5.4 CUSF Trigger Information Objects

This subclause provides an overview of the trigger information required for the CUSF.

Detailed description of these objects can be found in EN 301 931-4.

The entities/components in the CUSF that are involved in the triggering mechanism are:

- Basic Non-Call Manager (BNCM);
- IN Non-Switching Manager (IN-NSM);
- Feature Interaction Manager / Non-Call Manager (FIM/NCM).

The entities/components shall collectively:

- perform the DP processing actions in EN 301 931-4;
- check for SCF accessibility;
- handle service feature interactions.

They shall hands back control to the CCF for at least the following cases:

- if a trigger (TDP) criteria match is not found (e.g., insufficient information to proceed): the CUSF logic returns supplementary service control to the CCF;
- if the association is abandoned: the CUSF logic returns supplementary service control to the CCF and continues processing;
- if the destination SCF is not accessible: the CUSF logic will apply final treatment to the end user for the TDP-R case.

The CUSF FSM passes component handling 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.

A Configuration Management Function shall be available for managing the trigger tables. The following minimal capabilities shall be provided:

- Introduction of new trigger data along with the corresponding service key and revision number;
- Remove a trigger data identified by its service key;
- Activate or Deactivate existing trigger data;
- Verification of the current configuration.

The CUSF trigger information model includes:

- Trigger mode (i.e.: an indication on whether triggering should cause call unrelated processing suspension);
- Detection Point in the BCUSM;
- Trigger Criteria;
- Information required for SCF addressing;
- Service Key;
- Application context name;
- Association to a CCF/CUSF user resource (e.g., trunk, line,...).

5.5.5 CUSF FSM Structure

Figure 12 shows the CUSF FSM structure. The interfaces shown in this figure are internal, and are not for standardization.

The relationship between the BCUSM and the CUSF FSM may be described as follows:

a) When a call unrelated associated association / operation attempt is initiated by an end user and is processed at an exchange, an instance of a BCUSM is created. As the BCUSM proceeds, it encounters detection points. If a DP is armed as a Trigger DP (TDP) an instance of a CUSF FSM is created.

b) 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 management functions related to the execution of operations received from the SCF are executed by the CUSF Management Entity (CUSME)-Control. The CUSME-control interfaces the different various CUSF FSMs and the Functional Entity Access Manager (FEAM). The Functional Entity Access Manager (FEAM) provides the lower level maintenance functions on the interface as described in clause 5.

The CUSME-control maintains the dialogues with the SCF on behalf of all instances of the CUSF Finite State Model (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.

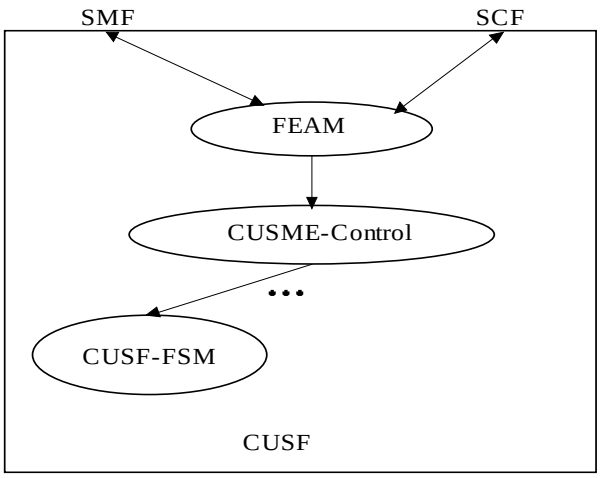


Figure 12: CUSF FSM Structure

5.6 Service Management Function (SMF) Models

5.6.1 General

A model of the SMF is shown in Figure 13.The purpose of this model is to provide a framework for the understanding of service deployment functions, service provisioning functions, service operation control functions, and service monitoring functions. Detailed description of the SMF can be found in ITU-T Recommendation Q.1236 [56].

5.6.2 SMF Components

The SMF model to provide the required functionality is shown in Figure 13. It is noted that this shows a conceptual model of the SMF and is not intended to imply an actual implementation of the SMF. The SMF platform provides a configuration manager, performance manager, fault manager, security control manager, and testing manager.

5.6.2.1 Configuration Manager

The Configuration Manager is responsible for changing configurations for service deployment and service operation control, or to correct faults.

5.6.2.2 Fault Manager

The Fault Manager analyses and correlates alarms and performs appropriate tests to determine root cause of fault conditions.

5.6.2.3 Performance Manager

The Performance Manager is responsible for gathering the appropriate performance information necessary to gauge the overall performance of service and network resources.

5.6.2.4 Testing Manager

The Testing Manager is responsible for the invocation and management of testing capabilities. The Testing Manager may be used by the Fault Manager in the fault resolution process.

5.6.2.5 Security Control Manager

The security control manager provides for logging of invalid access attempts and generation of alarms, activation and deactivation of appropriate security features in each FE to enable correct interface operation, and distribution of security related information.

5.6.2.6 Security Access Manager

The Security Manager provides the functionality needed for secure access to the functions of the IN FEs. This functionality applies security procedures to ensure data confidentiality, origin authentication and data integrity.

5.6.2.7 Functional Entity Access Manager (FEAM)

The FEAM provides the lower level maintenance functions on the interface as described in clause 50.

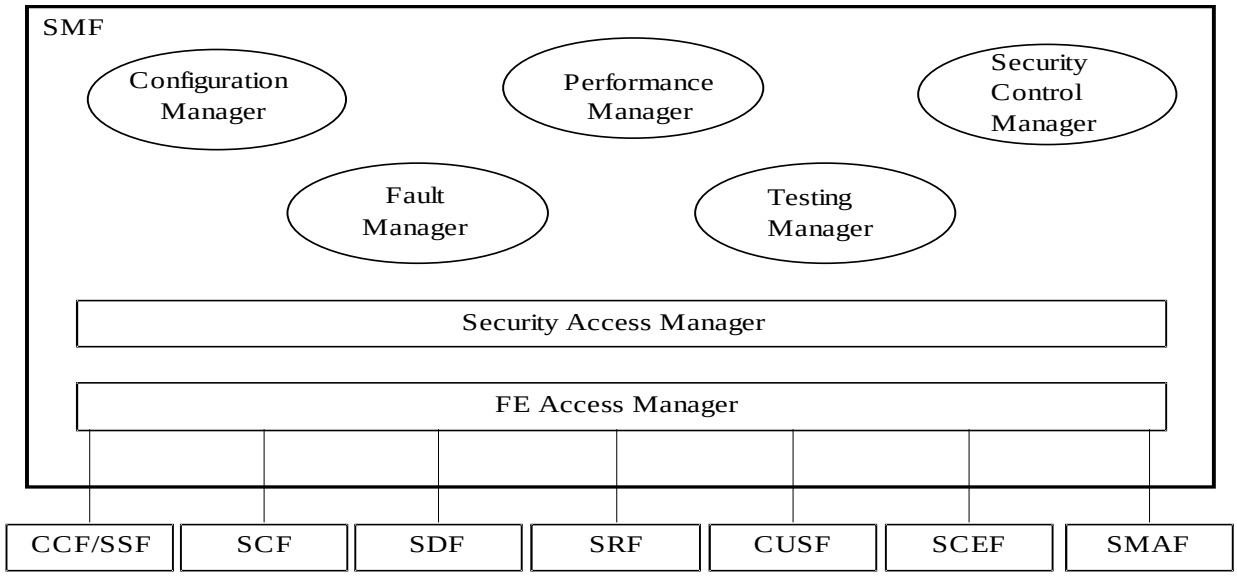


Figure 13: SMF Model

6 Use of FE Relationships

This subclause provides a summary of the FE relationships as defined in the present document subpart. A more detailed description can be found in the appropriate interface section of EN 301 931-2 to EN 301 931-4, and in ITU-T Recommendation Q.1236 [56].

Most of the relationships are designed only for use within a trusted network since they are not secure. However, a few internetworking interfaces have been designed to allow data transfer among networks. Where this capability is allowed, it is mentioned in the relationship description below.

6.1 SCF-SSF Relationship

The SCF-SSF relationship is used for the transfer of information between an SCF and an SSF in the public network. This relationship, with the aid of the SCF-SDF/SRF, and SDF-SDF relationship, provides a variety of service features like Freephone, UPT, VPN, etc.

6.2 SCF-SCF Relationship

While several types of information can be transferred across the SCF–SCF interface, the SCF-SCF relationship is used mainly for the distribution of service logic between two SCFs in the public network. The relationship is used to support call-related operations.

The SCF-SCF relationship is also one of a limited set of relationships to support internetworking. As such, it provides a point of interconnection to the network, effectively hiding the specific structure of the network and providing access security to the network from other public networks.

6.3 SCF-IAF Relationships

For INAP purposes, the IAF is treated as an SCF, and therefore has the same attributes as the SCF-SCF relationship.

6.4 SRF-CCF Relationship

Current modelling techniques place the CCF under control of the SSF, and therefore any relationship is also with the SSF. An SRF-CCF/SSF relationship exists to manage and convey information such that the bearer channel of a call can be connected to an SRF.

6.5 SRF-SCF Relationship

This section describes the relationship between the SRF and the SCF for connection to and control of specialized resources and User Interaction-scripts.

For all cases in which there is to be call-related interaction, the establishment of this relationship shall be preceded by the establishment of a relationship between the SCF and the SSF.

6.6 SCF-SDF Relationship

This SCF-SDF relationship is used for information flows between a SCF and a SDF in the public network. The relationship is used mainly for Secured Data Acquisition for service data.

The SCF-SDF relationship supports information flows for services when no call is actually taking place (call-unrelated) as well as during call processing. For the most part, the call-unrelated actions are to support the registration, authentication, encryption and handover procedures for terminal and personal mobility.

The SCF-SDF relationship is also one of a limited set of relationships to support internetworking. As such, it provides a point of interconnection to the network, effectively hiding the specific structure of the network and providing access security to the network from other public networks.

6.7 SDF-SDF Relationship

This SDF-SDF relationship is used for supporting data duplication or data replication between two or more SDFs.

The SDF-SDF relationship is also one of a limited set of relationships to support internetworking. As such, it provides a point of interconnection to the network, effectively hiding the specific structure of the network and providing access security to the network from other public networks.

6.8 CUSF-SCF Relationship

The SCF-CUSF relationship is used for messages between an SCF and an CUSF in the public network. This relationship provides service or service features which need call-unrelated user interaction. Major features of this relationship are: user location registration, user authentication, supplementary service activation or de-activation.

6.9 CUSF-SSF Relationship

The CUSF and the SSF may have a relationship, but the relationship is not specified, and is not mandatory. This relationship may be used to influence the basic call processing in the CCF/SSF (e.g. activation or de-activation of Call Forwarding) via call-unrelated interaction.

6.10 SMF-SCF Relationship

This relationship manages the entities/components and data related to the SCF.

This relationship performs:

a) Configuration Management Functions;

b) Fault Management Functions;

c) Performance Management Functions;

d) Testing functions;

e) Security Management Functions.

6.11 SMF-SDF Relationship

This relationship manages the entities/components and data related to the SDF.

This relationship performs:

a) Configuration Management Functions;

b) Fault Management Functions;

c) Security Management Functions.

6.12 SMF-CCF/SSF Relationship

This relationship manages the entities/components and data related to the CCF/SSF.

This relationship performs:

a) Configuration Management Functions;

b) Fault Management Functions;

c) Performance Management Functions;

d) Testing Functions;

e) Security Management Functions.

6.13 SMF-SRF Relationship

The relationship between the SRF and the SMF deals with the management of the specialized resources.

This relationship performs:

a) Configuration Management Functions;

b) Fault Management Functions;

c) Performance Management functions;

d) Testing Functions;

e) Security Management Functions.

6.14 SMF-SMAF Relationship

There are two kinds of SMAF users, service administrator and service subscriber. Service Administrator consists of both the Service Provider and Network Provider. The relationship manages the entities/components and data related to SMAF.

This relationship provides access to the following functions:

a) Fault Management Functions;

b) Testing Functions;

c) Performance Management Functions;

d) Security Management functions;

e) Configuration Management Functions.

6.15 SMF - SCEF Relationship

The SCEF-SMF relationship conveys the output of the SCEF to the SMF. Once a new service has been installed on the SMF from the SCEF, the task of deploying the service is the role of the SMF and initiated from within the SMF.

This relationship performs:

a) Configuration Management functions;

b) Testing Functions;

c) Generic Management Scripting interface;

d) Security Management Functions.

6.16 SMF-SMF Relationship

This relationship manages the interworking between entities/components and data residing in different networks. This relationship is the TMN X interface as defined in ITU-T Recommendation M.3320 [26].

The SMF-SMF relationship performs:

a) Fault Management Functions;

b) Testing Functions;

c) Performance Management Functions;

d) Security Management Functions.

6.17 SMF-CCF/CUSF Relationship

This relationship manages the entities/components and data related to the CCF/CUSF.

This relationship performs:

a) Configuration Management Functions;

b) Fault Management Functions;

c) Performance Management Functions;

d) Testing Functions;

e) Security Management Functions.

7 Protocol Realization

7.1 Overview

For the purpose of communicating with functional entities located in a different physical entities, each FE uses one or more Application Entity (AE).

On the following physical interfaces, the AE includes TC (Transaction Capabilities) and one Application Service Elements (ASE) for supporting each of the operation packages which may be used during an instance of communication:

- SSP-SCP Interface;
- SCP-IP Interface;
- SCP-SDP Interface;
- SDP-SDP Interface;
- SCP-SCP Interface;
- SCP-CUSP Interface.

Other protocol entity may also be included in an AE for supporting communication on other interfaces (e.g., ISUP).

During an instance of communication, an FE creates an application process instance (API) which has either single interactions or multiple co-ordinated interactions with application process invocations located in other physical entities. In the first case, a Single Association Control Function (SACF) co-ordinates the ASEs and their use of the TC services. The SACF together with the ASEs and TC resources involved in one instance of communication form a Single Association Object (SAO) In the second case, a Multiple Association Control Function (MACF) co-ordinates several SAOs, each of which interact with an SAO in a remote PE.

The MACF (when relevant) together with the SACF(s) and ASEs provide the same functionality than the FEAM, described in clause 5.

AE-types are not intended to be standardized. The organization of a PE in terms of AE is a local matter. Any combination of ASE/AE is acceptable provided that it supports all the application-contexts which realize the contracts defined for the contained ROS-objects (i.e., FEs).

7.2 Application Contexts

When communication between Physical Entities is supported by Signalling System No7, operation contracts are realized using the services of the Transaction Capabilities (TC) as defined in ITU-T Recommendation Q.771 [41].

The particular set of ASEs involved in the realization of a contract, together with TC and any rules for their co-ordinated working, constitutes an application-context.

The static aspects of an application context are specified using the APPLICATION-CONTEXT Information Object Class notation as defined in ITU-T Recommendation Q.775 [45]. This includes the identification of the abstract-syntaxes which are required for representing the information conveyed between the initiator and the responder of the contract.

The present document defines a particular realization of the operation contracts using SS7; however, other signalling systems may equally be used (e.g. DSS1 Layer 3) providing that all aspects of these operation contracts can be implemented.

7.3 Abstract Syntax and Transfer Syntax

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

a) the abstract syntax of TC dialogue control protocol data units, **dialogue-abstract-syntax,** which is needed to establish the dialogue between FEs and specified in ITU-T Recommendation Q.773 [43];

b) the abstract syntax for conveying the protocol data units for invoking the operations involved in the operation packages 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 **TCMessages{}** defined in ITU-T Recommendation Q.773 [43].

All these abstract syntaxes shall (as a minimum) be encoded according to the Basic ASN.1 encoding rules with the restrictions listed in ITU-T Recommendation Q.773 [43].

The present document includes a number of network specific parameters which are defined as being of type OCTET STRING. The internal structure of such parameters may in turn be defined using ASN.1 and use the related Basic Encoding Rules (BER). In such a case the value of this parameter (after the first tag and length information) is the BER encoding of the defined ASN.1 internal structure. The tag of this parameter as defined by ITU-T shall never be replaced.

7.4 SACF/MACF Rules

7.4.1 Reflection of TCAP AC

TCAP Application Context 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 Recommendation Q.77X-Series (*Transaction capabilities application part*) for a more detailed description of the TCAP AC negotiation mechanism.

7.4.2 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 with any other operation are:

- charging operations.

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 executed until some other operation has progressed to some extent or finished, the sending PE (usually SCP) can control this by sending the operations in two separate messages.

This method does not imply that all operations sent in the same message should 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 other parts of EN 301 931, the FE-specific rules take precedence over the generic rules.

8 Congestion Control

8.1 Lower layer flow control

SCCP flow control allows the SCCP User to toggle access to SCCP Subsystems (identified by SSN). This allows the user to turn on, or off, all of the traffic to a particular subsystem within the node. The same type of procedure shall apply as defined for ISDN User Part signalling congestion control. The INAP procedures for signalling congestion control shall as far as possible be aligned with the ISDN User Part signalling congestion control procedures as specified in ITU-T Recommendation Q.767 [40], 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, AnalysedInformation, or InitiateCallAttempt) into the affected direction in several steps. The above procedure may only apply to traffic which uses MTP Point Code addressing in the affected direction.

In addition to this subsystem management, SCCP provides the capability to restrict traffic to the SCCP based on a specific congestion level. This congestion level applies to the total SCCP traffic for the node and is determined locally within the SCCP. Changes to the congestion level are broadcast to SCCP User to enable traffic limitation. Traffic limitation is achieved by comparing the importance parameter passed with SCCP data messages to the current congestion level for the destination node and discarding excess traffic. Judicious choice of the importance values for TCAP messages would allow restriction of new service requests while allowing existing requests to proceed. This form of traffic limitation would apply to all IN services using the resources of the node under congestion.

In the absence of network operator defined values, the Importance parameter should be assigned from the default values defined in Table 1.

Table 1: Proposed Importance Assignment for INAP messages

	SCCP Importance Setting		
Traffic Type	**BEGIN**	**CONTINUE**	**END**
INAP (SSF->SCF)	0	4	4
INAP (CUSF->SCF)	0	4	4
INAP (SCF->SSF)	4	6	6
INAP (SCF->CUSF)	4	6	6
INAP (SCF-SCF)	4	6	6
INAP (SCF-SDF)	4	6	6
INAP (SCF-SRF)	4	6	6
INAP (SDF-SDF)	4	6	6
-> indicates restricted direction of initial BEGIN message. - indicates initial BEGIN can be in either direction			

8.2 Application level flow control

The INAP protocol includes procedures (e.g., call gapping on the SCP-SSP interface) for requesting other entities to reduce the number of new incoming dialogues according to a set of interface-specific criteria. Such procedures can be initiated by a local management function which detects congestion or by an SMF.

For all interfaces, other than the SSF-SCF or CUSF-SCF, the application of flow control mechanisms can potentially interrupt the operation of a Service Logic Program (SLP) executing in the initial SCF. The flow control logic shall ensure that the running SLP receives a meaningful error message (e.g. network congestion) as the result of a request being denied due to flow control limitations.

NOTE: This type of handling may not be acceptable if the service is one where the user has already performed a large degree of interaction with the service (e.g. collected authentication information).

The flow control logic can detect network congestion either by analysis of error return and/or operation request timeout patterns or by receipt of an explicit flow control message from the remote node.

The flow control logic should initiate SSF/CUSF level flow control in response to analysis of remote FE flow control to reduce network message flow closer to the source of the traffic.

For each interface, a detailed description is provided in EN 301 931-2 to EN 301 931-4.

9 Protocol mechanisms

9.1 Compatibility Mechanisms and extensibility rules

9.1.1 Introduction

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

There are three levels of changes:

a) Minor changes to INAP in future standardized versions:

A minor change can be defined as a change in functionality which is not essential for the requested IN service. In case 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.

b) Major changes to INAP in future standardized versions:

A major change can be defined as a change in functionality which is essential for the requested IN service. In case 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.

c) 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 EN. Such extensions may be defined by individual network operators, vendors or any other body (e.g., regional standardization organizations).

9.1.2 Definition of INAP compatibility mechanisms

9.1.2.1 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 application entity, 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 ITU-T Recommendation Q.773 [43].

9.1.2.2 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.680 [71]). These procedures are not applicable to SCF-SDF and SDF-SDF interfaces which have their own rules, inherited from ITU-T Recommendation X.519 [69] and defined in EN 301 931-4.

9.1.2.3 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 open type 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, similar to the comprehension required mechanism described in the previous subclause. Refer to ITU-T Recommendation Q.1400 [58] for a definition of this mechanism.

In addition to this mechanism, it is also possible to include network specific additions as components of an operation argument: the range of 50 to 59 (both inclusive) tag values is reserved for this usage.

9.2 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 [66] 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 [66] and SPKM for use in this capability set. This section also defines screening requirements for internetworking interfaces.

For this capability set the provision for security functions is required for the SCF-SDF, SDF-SDF, and SCF-SCF interfaces.

9.2.1 Interface Security Requirements

The security requirements on an IN system could be divided in the two following main families:

a) 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.

b) 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), INAP protocol screening (i.e. the filtering of INAP operations, parameter and parameter contents), etc.

For each of the above problems, the interfaces have to provide processes to enable data confidentiality, data origin authentication, data integrity and key management. These capabilities are generally described in ITU-T Recommendation X.500 [64] .

For the internetworking interfaces, the entities involved have to provide the functionality to enable screening of the messages sent and received across the interfaces.

9.2.2 INAP screening requirements

Screening refers to the ability to filter the messages and the contents of the messages sent/received across internetworking interfaces. With each internetwork traffic relation an INAP screening profile performing the screening actions shall be associated. This screening functionality is part of the FE's Management Entity.

It shall be possible to attach another INAP screening profile to a specific internetwork relation without severe service interruption (i.e. only outstanding transactions may be affected during the replacement of the screening profile and normal traffic handling shall be resumed immediately following the attachment of the new screening profile to the specific traffic relation).

9.2.2.1 INAP Application Context Screening Requirements

The present document defines a set of Application Contexts for INAP. INAP Application Context screening ensures the control per internetwork traffic relation of the Application Context of the INAP capability set. Therefore, INAP operations not within the Application Context negotiated shall be rejected. Also INAP parameters per specific operation, not within the Application Context negotiated shall be rejected or ignored.

The following signalling information shall pass the INAP Application Context screening functions unaffected:

1) Application Context negotiation information,

2) SCCP related address information elements.

9.2.2.2 INAP Protocol Screening Requirements

INAP protocol screening ensures the control per internetwork traffic relation of the signalling capabilities and handling of INAP operations, parameters and parameter contents. The INAP screening functions imply manipulations of received and/or sent signalling information per internetwork traffic relation. The screening actions may be different for INAP signalling information sent or received.

For INAP operations, the protocol screening functions shall be able to let the operation pass unchanged or discard the operation depending on the screening profile associated with the traffic relation.

For INAP parameters, the protocol screening functions shall be able to let the parameter pass unchanged or discard the parameter depending on the screening profile associated with the traffic relation.

For INAP parameter contents, the protocol screening functions shall be able to let the parameter content pass unchanged or change the parameter independent on the received/sent content or change the parameter content dependent on the received/sent content.

The following signalling information shall pass the INAP protocol screening functions unaffected:

1) information related to the basic operation or parameter structures;

2) information related to maintenance functions;

3) SCCP related address information elements.

9.2.3 Security Procedures and Algorithms

Several assumptions are provided to the current interface security procedures to focus the work, but which also provides 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 signalling 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 2: 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 such as Encryption and Digital Signatures (including Message Authentication Code (MAC) techniques) shall be used.

9.2.3.1 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 authentication uses possible procedures defined in ITU-T Recommendation X.509 [66], together with a new procedure, SPKM. The primary purpose of the ITU-T Recommendation X.509 [66] 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 implementers within the scope of this capability set.

The current authentication procedures are:

1) Simple Authentication (as defined in ITU-T Recommendation X.509 [66]), using Bind operations with.
2) Strong Authentication, (as defined in ITU-T Recommendation X.509 [66]), using Bind operations with strong credentials; or using SPKM.
3) External Procedures, using Bind operations.
4) SPKM (Simple Public Key GSS-API Mechanism)(as defined in ITU-T Recommendation X.519 [69]):
   - 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 Identifiers 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.

9.2.3.2 Three-Way Mutual Authentication

The use of three-way mutual authentication procedures (including SPKM) in the establishment of a security association between two Functional Entities (FEs) implies the use of the Generic Upper Layers Security (GULS) Security Exchange Service Element (SESE) protocol (defined in ITU-T Recommendations X.831 [82] and X.832 [83]) as described in ITU-T Recommendation X.519 [69] .

9.2.3.3 Assignment of Credentials

Procedures for assigning credentials to users or FEs are generally defined in ITU-T Recommendation X.500 [64] .

9.2.4 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.

9.2.5 Security FSM definitions

The Security FSMs for two-way and three-way Mutual Authentication are described in ITU-T Recommendation Q.1228 [54]. They illustrate how the security procedures are handled across the interface between two FEs.

10 Services assumed from Lower Layers

This clause provides a general description of services assumed from lower layers. Further descriptions in the context of functional entities are provided in EN 301 931-2 to EN 301 931-4.

10.1 Services assumed from TCAP

The SS7 application layer protocol defined in the present document, 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 Transaction Capabilities. 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 Application Contexts (AC) are to be used for FE differentiation within a physical node then the version of TC used shall support the dialogue portion of TC (i.e. White Book TC).

This requirement applies to all interfaces, not just those used for internetworking.

Table 3 defines which versions of TC are the minimum versions required to support the defined IN interfaces:

Table 3: Minimum TC Recommendations Requirements for INAP interfaces

Interface	IN CS-3
SSF - SCF	Blue Book (see Note)
SCF - SRF	Blue Book (see Note)
SCF - SDF	White Book (93)
SCF - SCF	White Book (93)
SDF - SDF	White Book (93)
CUSF - SCF	Blue Book (see Note)
NOTE: If AC negotiation is required then White Book (93) is the minimum version required.	

10.1.1 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.

10.1.1.1 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 physical entities:

- 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 ITU-T Recommendation Q.774 [44].

In case the dialogue is established and the FE has received an operation, leading to the termination of the relationship does not interest 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.

    - Prearranged 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 (prearranged end) after the last associated operation timer expires. The receiving entity can end the dialogue with a TC-END request primitive (prearranged 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.

10.1.1.2 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 the different interface parts of the present document.

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 T_{SRF} or T_{CUSF} or T_{SSF} (in case of the last CS in the CSA), dialogue shall be terminated by means of a 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 application time out).

In error situations prearranged end shall not be used to terminate the TCAP dialogue. In case any application entity 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 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 ITU-T Recommendation Q.774 [44].

10.1.1.3 Dialogue Handling

10.1.1.3.1 Dialogue Establishment

The establishment of an INAP dialogue involves two application processes, one that is the dialogue-initiator and one that is the dialogue-responder.

Application context negotiation may not be supported in all physical entities 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.

10.1.1.3.1.1 Sending of a TC-BEGIN request

Before issuing a TC-BEGIN request primitive, SACF shall store the AC-name and if present the user-information parameter.

SACF shall request the invocation of the associated operations using the TC-INVOKE service.

After processing of the last invocation request, SACF shall issue a TC-BEGIN request primitive.

The initiator SACF 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 "prearranged release".

If no TC indication primitive is expected because no dialogue is to be established according to the rules as stated in detail in the relevant clause of ITU-T Recommendations Q.1238.2 to Q.1238.7, SACF will wait for the last associated TCAP operation timer to expire and issue a TC-END request with the release method parameter set to "prearranged release".

10.1.1.3.1.2 Receipt of a TC-BEGIN indication

On receipt of a TC-BEGIN indication primitive, responder SACF shall:

- Analyse the application-context-name if included in the primitive. If it is supported, process any other indication primitives received from TC.
- If no dialogue is to be established according to the rules as stated in detail in the relevant clause of EN 301 931-2 to EN 301 931-4, SACF will wait for the last indication primitive from TC and issue a TC-END request with the release method parameter set to "prearranged 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.

10.1.1.3.1.3 Receipt of the first TC-CONTINUE indication

On receipt of the first TC-CONTINUE indication primitive for a dialogue, SACF shall check the value of the application-context-name parameter. If this value matches the one used in the TC-BEGIN request primitive, SACF shall process the following TC component handling indication primitives, 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.

10.1.1.3.1.4 Receipt of a TC-END indication

On receipt of a TC-END indication primitive in the dialogue initiated state, SACF 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 SACF shall process the following TC component handling indication primitives.

10.1.1.3.1.5 Receipt of a TC-U-ABORT indication

Receipt of a TC-U-ABORT indication primitive is described as part of user abort procedure (see the relevant subclause of this clause). If the abort reason is application context name not supported, the responding side may propose an alternative application context name in the TC-U-ABORT indication. If an alternative application context is proposed, the receiving entity shall check this name and if it can be supported a new dialogue may be established.

10.1.1.3.1.6 Receipt of a TC-P-ABORT indication

Receipt of a TC-P-ABORT indication primitive is described as part of provider abort procedure.

10.1.1.3.2 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.

10.1.1.3.2.1 Sending entity

SACF shall process any component handling request primitives.

After processing the last component handling request primitive, SACF shall issue a TC-CONTINUE request primitive.

10.1.1.3.2.2 Receiving entity

On receipt of a TC-CONTINUE indication primitive SACF shall accept zero, one or several TC component handling indication primitives and process them.

10.1.1.3.3 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 detail in the relevant clause of EN 301 931-2 to EN 301 931-4.

The dialogue termination procedure is driven by the following events:

- a TC-END request primitive;
- a TC-END indication primitive.

10.1.1.3.3.1 Sending of TC-END request

When the dialogue shall no longer be maintained, SACF shall process any component handling request primitives.

After processing the last component handling request primitive (if any), SACF 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 detail in the relevant clause of EN 301 931-2 to EN 301 931-4.

10.1.1.3.3.2 Receipt of a TC-END indication

On receipt of a TC-END indication primitive, the SACF shall accept any component handling indication primitives and process them.

After processing the last component handling primitive all dialogue related resources are released.

10.1.1.3.4 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.

10.1.1.3.4.1 Sending of TC-U-ABORT request

After issuing a TC-U-ABORT request primitive, all dialogue related resources are released.

10.1.1.3.4.2 Receipt of a TC-U-ABORT indication

On receipt of a TC-U-ABORT indication, all dialogue related resources are released.

10.1.1.3.5 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.

10.1.1.3.5.1 Receipt of a TC-P-ABORT indication

On receipt of a TC-P-ABORT indication, all dialogue related resources are released.

10.1.1.3.6 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 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 Application Context 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 with the following qualifications:

- The Application Context Name parameter of the TC-CONTINUE service shall be set to the value of the Application Context 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 with the following qualifications:

- The Application Context Name parameter of the TC-END service shall be set to the value of the Application Context 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 with the following qualifications:

- The Abort Reason parameter of the TC-U-ABORT service shall be used as specified in ITU-T Recommendation Q.771 [41].

- The Application Context 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 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 ITU-T Recommendation Q.771 [41].

10.1.1.3.7 Default Mapping to TC Dialogue Parameters

10.1.1.3.7.1 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.

10.1.1.3.7.2 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 Application Context Names can be found in clause 5 of the present document.

10.1.1.3.7.3 User information

This parameter may be used by both initiating and responding application process in a network operator specific manner.

10.1.1.3.7.4 Component present

This parameter is used by SACF as described in ITU-T Recommendation Q.771 [41].

10.1.1.3.7.5 Termination

The value of the release method parameter of the TC-END request primitive is set by SACF according to the rules as stated in detail in the relevant clause of EN 301 931-2 to EN 301 931-4.

10.1.1.3.7.6 Quality of service

The quality of service of TC request primitives is set by the SACF to the following value:

- sequencing requested;
- return option, this parameter is set by SACF in an implementation dependent manner.

10.1.1.4 Component Handling

10.1.1.4.1 Procedures for INAP Operations

This subclause describes the procedures for INAP operations.

10.1.1.4.1.1 Operation invocation

SACF 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.

10.1.1.4.1.2 Operation invocation receipt

On receipt of a TC-INVOKE indication primitive, SACF 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, request 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, SACF waits for the corresponding response.

10.1.1.4.1.3 Operation Response

For user confirmed operations, SACF 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.

10.1.1.4.1.4 Receipt of a response

On receipt of a TC-RESULT-NL indication, SACF 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, SACF 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, SACF shall:

- if the error code is not defined for the SACF or is not one associated with the operation referred to by the invoke identifier, 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, SACF shall:

- accept the TC-U-REJECT indication primitive.

On receipt of a TC-L-REJECT indicating "return result problem, return error unexpected", SACF shall inform the application process.

On receipt of a TC-L-REJECT indicating "return error problem, return error unexpected", SACF 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.

On receipt of a TC-L-CANCEL indication, the SACF 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.

10.1.1.4.1.5 Other events

This subclause describes the behaviour of SACF 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 detail in the relevant clause of EN 301 931-2 to EN 301 931-4. 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 detail in the relevant clause of EN 301 931-2 to EN 301 931-4.

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 SACF 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.

10.1.1.4.2 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 the relevant subclause of this clause 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 shall be one of the valid operations supported by the negotiated Application Context for the TC dialogue and shall 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 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 shall 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 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 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.

10.1.1.4.3 Default Mapping to TC Component Parameters

10.1.1.4.3.1 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 an AE within a specific TC dialogue. The TC dialogue is identified by the Dialogue Id parameter.

10.1.1.4.3.2 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.

10.1.1.4.3.3 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.

10.1.1.4.3.4 Class

The value of this parameter is set according to the type of the operation to be invoked according to the operation definitions in the various subparts of the present document Part 1 containing the interface specifications.

10.1.1.4.3.5 Time out

The value of this parameter is set according to the type of operation invoked.

10.1.1.4.3.6 Last component

This parameter is used as described in ITU-T Recommendation Q.771 [41].

10.1.1.4.3.7 Problem code

This parameter is used as described in the relevant subclause of this clause.

10.1.1.4.3.8 Abort reason

This parameter is used by SACF, and attributes and coding are specified by network operator.

10.2 Services assumed from SCCP

This subclause 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.

The services described are those given in the SCCP post White Book ITU-T Recommendations (1993) and Q.715 [35] (SCCP User Guide) should be consulted to identify possible interworking and compatibility issues between the different SCCP versions.

10.2.1 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.

10.2.2 Service Functions from SCCP

10.2.2.1 SCCP Connectionless Services

The following Connectionless 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 routing or segmentation/reassembly 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 routing/segmentation error occurs.

10.2.2.1.1 INAP Addressing

The INAP addressing elements consist of information contained within the Calling and the Called Addresses which are sent by the application to TC for use by SCCP.

The application expects the SCCP to route messages by either:

a) the use of the Destination Point Code (DPC), the Subsystem Number (SSN) and MTP SAP (Service Access Point) instance; or

b) the use of the Global Title (GT) plus optionally the SSN, DPC and MTP SAP.

The application also specifies to SCCP whether to use Route on SSN or Route on GT for both the Called and Calling Addresses.

If INAP requires additional addressing information it shall be carried in the GT portion of the address specification regardless of which form of routing is specified.

Method a) 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 ITU-T Recommendation Q.713 [33], may be allocated for the different network specific applications, e.g. a SSN may be allocated for a SCF functionality.

Method b) 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 SSNs allocations within the destination network. The network that determines the end-node to which the message is to be delivered has to perform a Global Title 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 ITU-T Recommendation Q.713 [33].

10.2.2.1.1.1 Global Title Indicator

This indicator specifies the method employed for the formatting of the address information. The format with the indicator value 4 is always used for internetwork connections.

10.2.2.1.1.2 Translation Type

The Translation Types are defined within ITU-T Recommendation Q.713 [33].

10.2.2.1.1.3 Numbering Plan

1) For Addresses used on internetwork interfaces the Numbering Plan shall be either Generic Numbering, or E.164 [90] as defined in Annex B of ITU-T Recommendation Q.713 [33].

2) For Addresses used on all other interfaces, any of the Numbering plans defined in ITU-T Recommendation Q.713 [33] may be used if deemed suitable.

10.2.2.1.1.4 Global Title Address Information

This is the actual INAP address information supplied by the application and is encoded as indicated by the encoding scheme.

10.2.2.1.1.5 Encoding Scheme

The application should set the value of the encoding scheme according to the format of the GTAI. The allowed values are defined in ITU-T Recommendation Q.713 [33].

The network provider shall ensure that any change of GT value during translation preserves any INAP specific information contained in the initial GT value. The GT translation data in the network shall not delete the GT information, if present, from the Address.

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 is at least White Book 1992.

10.2.2.1.2 Sequence Control

The application will specify whether SCCP protocol class 0 or 1 is required.

Class 0 is used when the in-sequence delivery of messages to a specific called address is not required.

Class 1 is used when the in-sequence delivery of messages to a specific called address is required.

10.2.2.1.3 Return on Error

The use of Return on Error mechanism may be required by the IN applications so 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 routing or segmentation/reassembly failure back to the issuing user.

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 ITU-T Recommendation Q.711 [32]. The N-NOTICE primitive is used to return the undelivered message to the originating user.

10.2.2.1.4 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 data.

Also the SCCP is expected to perform the reassembly function on received segmented messages and deliver the reassembled user data 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 user data into more than 16 segments.

10.2.2.1.5 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 (07/96) ITU-T Recommendation Q.711 [32].

Also there exist other congestion control mechanisms as indicated below in SCCP Management.

10.2.2.2 SCCP Connection Oriented Services

The use by IN applications for the Connection-oriented services is outside the scope of this capability set.

10.2.2.3 SCCP Management

The subsystems used within the IN scenario expect the SCCP to provide management procedures to maintain network performance by re-routing 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 ITU-T Recommendation Q.711 [32].

To achieve the above, the SCCP is expected to perform the following procedures:

- Signalling point status management (N-PCSTATE, which includes the signalling point prohibited, signalling point allowed, signalling point congested, and local MTP availability subprocedures).
- Subsystem status management (N-STATE, which includes the subsystem prohibited, subsystem allowed, and subsystem status test subprocedures).
- Co-ordinated state change (N-COORD, which is a procedure which allows a duplicated subsystem to be withdrawn from service without affecting the performance of the network).

11 Error Definitions

This clause provides a general description of errors related to INAP operations. Detailed error procedure descriptions are defined in EN 301 931-2 to EN 301 931-4.

11.1 AttributeError

11.1.1 Error description

The AttributeError error is sent by the SDF to the SCF or another SDF to report an attribute related problem. The conditions under which an AttributeError error is to be issued are defined in ITU-T Recommendation X.511 [67] .

11.1.2 Parameter description

The AttributeError parameters and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.1.3 Relevant interfaces

The AttributeError error may be reported for operations on the SCF-SDF interface and SDF-SDF interface.

11.2 Cancelled

11.2.1 Error description

The Cancelled error 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.

11.2.2 Parameter description

None.

11.2.3 Relevant interface

The Cancelled error may be reported for operations on the SCF-SRF interface.

11.3 CancelFailed

11.3.1 Error description

The CancelFailed error is returned by Cancel if the cancelling of an Operation, as requested by the SCF, was not successful.

11.3.2 Parameter description

The following parameters may be returned when this error is reported:

- ***problem***: this parameter identifies failure reasons:

    1) 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);

2) tooLate, when the invokeID is known but the execution of the operation is in a state that it cannot be Cancelled 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;

3) operationNotCancellable, when the invokeID points to an Operation that the SCF is not allowed to cancel.

- ***operation***: this parameter contains the information that the operation failed to be cancelled.

11.3.3 Relevant interfaces

The CancelFailed error may be reported for operations on the SCF-SCF interface and SSF-SRF interface.

11.4 ChainingRefused

11.4.1 Error description

The ChainingRefused error is sent by the chaining terminator (or initiator) supporting to the initiator (or terminator) supporting SCF to reject a chained operation.

11.4.2 Parameter description

None.

11.4.3 Relevant interface

The ChainingRefused error may be reported for operations on the SCF-SCF interface.

11.5 DirectoryBindError

11.5.1 Error description

The DirectoryBindError 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 [67] .

11.5.2 Parameter description

The DirectoryBindError parameters and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.5.3 Relevant interfaces

The DirectoryBindError may be reported for operations on the SCF-SDF interface and SDF-SDF interface.

11.6 DSAReferal

11.6.1 Error description

The DSAReferal 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 [68] .

11.6.2 Parameter description

The DSAReferral error parameters and problem codes are specified in ITU-T Recommendation X.518 [68] .

11.6.3 Relevant interface

The DSAReferral error may be reported for operations on the SDF-SDF interface.

11.7 ETCFailed

11.7.1 Error description

The 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).

11.7.2 Parameter description

None.

11.7.3 Relevant interface

The ETCFailed error may be reported for operations on the SSF-SCF interface.

11.8 ExecutionError

11.8.1 Error description

The ExecutionError is an error sent from the SDF to the 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.

11.8.2 Parameter description

The following parameters may be returned when this error is reported:

- ***problem:*** this parameter 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.
- ***common results:*** This parameter contains information which may be present to qualify the outcome of any operation used on the SCF-SDF or SDF-SDF interface.

11.8.3 Relevant interfaces

The ExecutionError error may be reported for operations used on the SCF-SDF interface and SDF-SDF interface.

11.9 ImproperCallerResponse

11.9.1 Error description

The ImproperCallerResponse error is reported when 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.

11.9.2 Parameter description

None.

11.9.3 Relevant interfaces

The ImproperCallerResponse error may be reported for operations on the SCF-SRF interface and SCF-SCF interface.

11.10 MissingCustomerRecord

11.10.1 Error description

The MissingCustumerRecord error is reported when the Service Logic Program could not be found in the SCF, because the required customer record does not exist, or the requested Service Logic Program Instance, 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.

11.10.2 Parameter description

None.

11.10.3 Relevant interfaces

The MissingCustomerRecord error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.11 MissingParameter

11.11.1 Error description

The MissingParameter error is reported when 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.

11.11.2 Parameter description

None.

11.11.3 Relevant interfaces

The MissingParameter error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.12 NameError

11.12.1 Error description

The NameError 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 [67] .

11.12.2 Parameter description

The name error parameter and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.12.3 Relevant interfaces

The NameError may be reported for operations on the SCF-SDF interface and SDF-SDF interface.

11.13 ParameterOutOfRange

11.13.1 Error description

The ParameterOutOfRange error is reported when 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.

11.13.2 Parameter description

None.

11.13.3 Relevant interfaces

The ParameterOutOfRange error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.14 Referral

11.14.1 Error description

The Referral 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 [67] .

11.14.2 Parameter description

The Referral error parameter and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.14.3 Relevant interface

The Referral error may be reported for operations on the SCF-SDF interface.

11.15 RequestedInfoError

11.15.1 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 CCF/SSF can not offer the information specified with RequestedInformationType but there exists other CCF/SSF that can offer the information.

11.15.2 Parameter description

A parameter with the following values may be returned when this error is reported:

- unknownRequestInfo when the requested information is not known to the SSF.
- requestedInfoNotAvailable when the requested information is not available in the SSF.

11.15.3 Relevant interface

The RequestedInfoError error may be reported for operation on the SSF-SCF interface.

11.16 ScfBindFailure

11.16.1 Error description

The ScfBindFailure 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.

11.16.2 Parameter description

The following parameter may be returned when the ScfBindFailure error is reported.

- ***FailureReason***: this parameter provides the reason why a bind operation has been rejected. It may convey one of the following kinds of indications:

  a) system failure;

  b) an scfTaskRefused; or

  c) a securityError.

In all cases, a more precise diagnostic is also provided.

11.16.3 Relevant interface

The ScfBindFailure error may be reported for operations on the SCF-SCF interface.

11.17 ScfReferral

11.17.1 Error description

The ScfReferral 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.

11.17.2 Parameter description

The following parameters may be returned when this error is reported:

- ***tryhere***, this parameter provides the Access Point Information as specified in ITU-T Recommendation X.511 [67] .
- ***securityParameters*** this parameter provides the security parameters as specified in ITU-T Recommendation X.511 [67] .

11.17.3 Relevant interface

The ScfReferral error may be reported for operations on the SCF-SCF interface.

11.18 ScfTaskRefused

11.18.1 Error description

The ScfTaskRefused error is returned by a physical entity if it was not able to fulfil a specific task as requested by an operation. It is identical to taskRefused when security protection is not activated.

11.18.2 Parameter description

The following parameter may be returned when the ScfTaskRefused error is reported.

- ***reason***: this parameter provides the same indications as the taskRefused Error parameter.
- ***securityParameter***: this parameter contains information for security protection.

11.18.3 Relevant interfaces

The ScfTaskRefused error may be reported for operations on the SCF-SCF interface.

11.19 SecurityError

11.19.1 Error description

The SecurityError 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 [67] .

11.19.2 Parameter description

The security error parameters and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.19.3 Relevant interfaces

The SecurityError error may be reported for operations on the SCF-SDF interface, SDF-SDF interface and SCF-SCF interface.

11.20 ServiceError

11.20.1 Error description

The ServiceError 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 [67] .

11.20.2 Parameter description

The ServiceError error parameters and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.20.3 Relevant interfaces

The ServiceError may be reported for operations on SCF-SDF interface and SDF-SDF interface.

11.21 ShadowError

11.21.1 Error description

The ShadowError 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 [70].

11.21.2 Parameter description

The ShadowError error parameters and problem codes are specified in ITU-T Recommendation X.525 [70].

11.21.3 Relevant interface

The ShadowError error may be reported for operations on the SDF-SDF interface.

11.22 SystemFailure

11.22.1 Error description

The SystemFailure error is returned by a physical entity if it was not able to fulfil a specific task as requested by an operation, and recovery is not expected to be completed within the current call instance.

11.22.2 Parameter description

A parameter with the following values may be returned when this error is reported:

- unavailable resource;
- a component failure;
- a basic call processing exception;
- a resource status failure;
- an end user failure;
- a screening failure.

11.22.3 Relevant interfaces

The systemFailure error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.23 TaskRefused

11.23.1 Error description

The TaskRefused Error is returned by a physical entity if it was not able to fulfil a specific task as requested by an operation, and recovery is expected to be completed within the current call instance.

11.23.2 Parameter description

A parameter with the following values may be returned when this error is reported:

- generic;
- unobtainable when e.g. the address used in the Connect operation is not obtainable;
- congestion.

11.23.3 Relevant interfaces

The TaskRefused error may be reported for operations on the SSF-SCF interface, SCF-SRF interface and SCF-CUSF interface.

11.24 TfcBindError

11.24.1 Error description

The TfcBindError error is sent by the SCF to the SCF/SDF, and by the SDF to the SDF to report a problem in establishing an authorized relationship. The conditions under which this error is to be issued are similar to these for the directoryBindError defined in ITU-T Recommendation X.511 [67] .

11.24.2 Parameter description

The following parameters may be returned when this error is reported:

- ***versions***: this parameter identifies the protocol version supported by the entity which sent this error.
- ***error***: this parameter is either a Service Problem or a Security Problem defined in ITU-T Recommendation X.511 [67] .

11.24.3 Relevant interfaces

The tfcBindError may be reported for operations on the SCF-SCF interface, SCF-SDF interface and SDF-SDF interface.

11.25 UnavailableResource

11.25.1 Error description

The UnavailvableResource error is reported when 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.

11.25.2 Parameter description

None.

11.25.3 Relevant interface

The UnavailableResource error may be reported for operations on the SCF-SRF interface.

11.26 UnexpectedComponentSequence

11.26.1 Error description

The UnexpectedComponentSequence error is reported when 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.

11.26.2 Parameter description

None.

11.26.3 Relevant interfaces

The UnexpectedComponentSequence error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.27 UnexpectedDataValue

11.27.1 Error description

The UnexpectedDataValue is reported when 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".

11.27.2 Parameter description

None.

11.27.3 Relevant interfaces

The UnexpectedDataValue error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.28 UnexpectedParameter

11.28.1 Error description

The UnexpectedParameter error is reported when 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.

11.28.2 Parameter description

None.

11.28.3 Relevant interfaces

The UnexpectedParameter error may be reported for operations on the SSF-SCF interface, SCF-SRF interface, SCF-SCF interface and SCF-CUSF interface.

11.29 UnknownLegID

11.29.1 Error description

The UnknownLegID 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.

11.29.2 Parameter description

None.

11.29.3 Relevant interfaces

The UnknownLegID error may be reported for operations on SSF-SCF interface and SCF-CUSF interface.

11.30 UnknownResource

11.30.1 Error description

The UnknownResource error is used to indicate to the SCF that a specific physical resource which is indicated by the ResourceID parameter, is not known to the SSF.

11.30.2 Parameter description

None.

11.30.3 Relevant interface

The UnknownResource error may be reported for operations on the SSF-SCF interface.

11.31 UnknownSubscriber

11.31.1 Error description

The UnknownSubscriber error is used to indicate to the SCF that a specific subscriber which is indicated by the SubscriberID parameter, is not known to the SRF.

11.31.2 Parameter description

None.

11.31.3 Relevant interface

The UnknownSubscriber error may be reported for operations on the SCF-SRF interface.

11.32 UpdateError

11.32.1 Error description

The UpdateError 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 [67] .

11.32.2 Parameter description

The UpdateError error parameters and problem codes are specified in ITU-T Recommendation X.511 [67] .

11.32.3 Relevant interfaces

The UpdateError may be reported for operations on the SCF-SDF interface and SDF-SDF interface.

12 Common Definitions

12.1 Object identifiers

The following ASN.1 module defines the object identifiers assigned to modules, packages and application contexts for IN CS-3.

```
IN-CS3-object-identifiers {itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
cs3(30) modules(1) in-cs3-object-identifiers(0) version1(0)}
DEFINITIONS ::=
BEGIN
-- For Modules from TCAP, ROS,
tc-Messages
    OBJECT IDENTIFIER ::= {ccitt recommendation q 773 modules(2) messages(1) version3(3)}
tc-NotationExtensions
    OBJECT IDENTIFIER ::= {ccitt recommendation q 775 modules(2)  notation-extension (4)
version1(1)}
ros-InformationObjects
    OBJECT IDENTIFIER ::= {joint-iso-itu-t remote-operations(4) informationObjects(5) version1(0)}
ros-genericPDUs
    OBJECT IDENTIFIER ::= {joint-iso-itu-t remote-operations(4) generic-ROS-PDUs(6) version1(0)}
ros-UsefulDefinitions
    OBJECT IDENTIFIER ::= {joint-iso-itu-t 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 [ED'S NOTE: IS IT USED?]
contexts    OBJECT IDENTIFIER ::= {ccitt recommendation q 1218 modules (0) contexts (8)
selectedContexts (1) version (1)}
-- For IN CS3 Modules
id-cs3          OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) cs3(30)}
modules     OBJECT IDENTIFIER ::= {id-cs3 modules(1)}
id-ac           OBJECT IDENTIFIER ::= {id-cs3 ac(3)}
id-at           OBJECT IDENTIFIER ::= {id-cs3 at(4)}
id-as           OBJECT IDENTIFIER ::= {id-cs3 as(5)}
id-oc           OBJECT IDENTIFIER ::= {id-cs3 oc(6)}
id-mt           OBJECT IDENTIFIER ::= {id-cs3 mt(7)}
id-sf           OBJECT IDENTIFIER ::= {id-cs3 sf(11)}
id-soa          OBJECT IDENTIFIER ::= {id-cs3 soa(21)}
id-aca          OBJECT IDENTIFIER ::= {id-cs3 aca(24)}
id-rosObject    OBJECT IDENTIFIER ::= {id-cs3 rosObject(25)}
id-contract     OBJECT IDENTIFIER ::= {id-cs3 contract(26)}
id-package      OBJECT IDENTIFIER ::= {id-cs3 package(27)}
```

```
id-avc          OBJECT IDENTIFIER ::= {id-cs3 avc(29)}
object-identifiers  OBJECT IDENTIFIER ::= {modules in-cs3-object-identifiers(0) version1(0)}
common-datatypes    OBJECT IDENTIFIER ::= {modules in-cs3-common-datatypes(1) version1(0)}
errortypes      OBJECT IDENTIFIER ::= {modules in-cs3-errortypes(2) version1(0)}
operationcodes  OBJECT IDENTIFIER ::= {modules in-cs3-operationcodes (3) version1(0)}
errorcodes      OBJECT IDENTIFIER ::= {modules in-cs3-errorcodes(4) version1(0)}
common-classes  OBJECT IDENTIFIER ::= {modules in-cs3-common-classes(5) version1(0)}
ssf-scf-datatypes   OBJECT IDENTIFIER ::= {modules in-cs3-ssf-scf-datatypes(6) version1(0)}
ssf-scf-classes OBJECT IDENTIFIER ::= {modules in-cs3-ssf-scf-classes(7) version1(0)}
ssf-scf-Operations  OBJECT IDENTIFIER ::= {modules in-cs3-ssf-scf-ops-args(8) version1(0)}
ssf-scf-Protocol    OBJECT IDENTIFIER ::= {modules in-cs3-ssf-scf-pkgs-contracts-acs(9) version1(0)}
scf-srf-datatypes   OBJECT IDENTIFIER ::= {modules in-cs3-scf-srf-datatypes(10) version1(0)}
scf-srf-classes OBJECT IDENTIFIER ::= {modules in-cs3-scf-srf-classes(11) version1(0)}
scf-srf-Operations  OBJECT IDENTIFIER ::= {modules in-cs3-scf-srf-ops-args (12) version1(0)}
scf-srf-Protocol    OBJECT IDENTIFIER ::= {modules in-cs3-scf-srf-pkgs-contracts-acs(13)
version1(0)}
scf-sdf-datatypes   OBJECT IDENTIFIER ::= {modules in-cs3-scf-sdf-datatypes (14) version1(0)}
scf-sdf-classes OBJECT IDENTIFIER ::= {modules in-cs3-scf-sdf-classes (15) version1(0)}
scf-sdf-Operations  OBJECT IDENTIFIER ::= {modules in-cs3-scf-sdf-ops-args(16) version1(0)}
scf-sdf-Protocol    OBJECT IDENTIFIER ::= {modules in-cs3-scf-sdf-pkgs-contracts-acs(17)
version1(0)}
sdf-sdf-Operations  OBJECT IDENTIFIER ::= {modules in-cs3-sdf-sdf-ops-args(18) version1(0)}
sdf-sdf-Protocol    OBJECT IDENTIFIER ::= {modules in-cs3-sdf-sdf-pkgs-contracts-acs(19)
version1(0)}
scf-scf-datatypes   OBJECT IDENTIFIER ::= {modules in-cs3-scf-scf-datatypes(20) version1(0)}
scf-scf-classes OBJECT IDENTIFIER ::= {modules in-cs3-scf-scf-classes(21) version1(0)}
scf-scf-Operations  OBJECT IDENTIFIER ::= {modules in-cs3-scf-scf-ops-args(22) version1(0)}
scf-scf-Protocol    OBJECT IDENTIFIER ::= {modules in-cs3-scf-scf-pkgs-contracts-acs(23)
version1(0)}
scf-cusf-datatypes  OBJECT IDENTIFIER ::= {modules in-cs3-scf-cusf-datatypes(24) version1(0)}
scf-cusf-classes    OBJECT IDENTIFIER ::= {modules in-cs3-scf-cusf-classes(25) version1(0)}
scf-cusf-Operations OBJECT IDENTIFIER ::= {modules in-cs3-scf-cusf-ops-args(26) version1(0)}
scf-cusf-Protocol   OBJECT IDENTIFIER ::= {modules in-cs3-scf-cusf-pkgs-contracts-acs (27)
version1(0)}
scf-sdf-Additional-Definitions  OBJECT IDENTIFIER ::= {modules in-cs3-scf-sdf-additional-definitions
(28) version1(0)}
-- Application Context
-- SSF/SCF Application Context
id-ac-cs3-ssf-scfGenericAC
    OBJECT IDENTIFIER ::= {id-ac ssf-scfGenericAC(4) version1(0)}
id-ac-cs3-ssf-scfAssistHandoffAC
    OBJECT IDENTIFIER ::= {id-ac ssf-scfAssistHandoffAC(6) version1(0)}
id-ac-cs3-ssf-scfServiceManagementAC
    OBJECT IDENTIFIER ::= {id-ac ssf-scfServiceManagementAC(7) version1(0)}
id-ac-cs3-scf-ssfGenericAC
    OBJECT IDENTIFIER ::= {id-ac scf-ssfGenericAC(8) version1(0)}
id-ac-cs3-scf-ssfTrafficManagementAC
    OBJECT IDENTIFIER ::= {id-ac scf-ssfTrafficManagementAC(10) version1(0)}
id-ac-cs3-scf-ssfServiceManagementAC
    OBJECT IDENTIFIER ::= {id-ac scf-ssfServiceManagementAC(11) version1(0)}
id-ac-cs3-scf-ssfStatusReportingAC
    OBJECT IDENTIFIER ::= {id-ac scf-ssfStatusReportingAC(12) version1(0)}
id-ac-cs3-scf-ssfTriggerManagementAC
    OBJECT IDENTIFIER ::= {id-ac scf-ssfTriggerManagementAC(13) version1(0)}
-- SRF/SCF Application Context
id-ac-srf-scfAC OBJECT IDENTIFIER ::= {id-ac srf-scfAC(14) version1(0)}
--SCF-SDF -  application contexts --
id-ac-indirectoryAccessAC
    OBJECT IDENTIFIER ::= {id-ac indirectoryAccessAC(1) version1(0)}
id-ac-indirectoryAccessWith3seAC
    OBJECT IDENTIFIER ::= {id-ac indirectoryAccessWith3seAC(2) version1(0)}
id-ac-inExtendedDirectoryAccessAC
    OBJECT IDENTIFIER ::= {id-ac inExtendedDirectoryAccessAC(3) version1(0)}
id-ac-inExtendedDirectoryAccessWith3seAC
    OBJECT IDENTIFIER ::= {id-ac inExtendedDirectoryAccessWith3seAC(27) version1(0)}
id-ac-trafficFlowControlAC
    OBJECT IDENTIFIER ::= {id-ac trafficFlowControlAC(28) version1(0)}
--SDF - SDF Application Contexts
id-ac-indirectorySystemAC
    OBJECT IDENTIFIER ::= { id-ac indirectorySystemAC(15) version1(0)}
id-ac-inShadowSupplierInitiatedAC
    OBJECT IDENTIFIER ::= { id-ac inShadowSupplierInitiatedAC(16) version1(0)}
id-ac-inShadowConsumerInitiatedAC
    OBJECT IDENTIFIER ::= { id-ac inShadowConsumerInitiatedAC(17) version1(0)}
id-ac-indirectorySystemWith3seAC
    OBJECT IDENTIFIER ::= { id-ac indirectorySystemWith3seAC(18) version1(0)}
id-ac-inShadowSupplierInitiatedWith3seAC
    OBJECT IDENTIFIER ::= { id-ac inShadowSupplierInitiatedWith3seAC(19) version1(0)}
id-ac-inShadowConsumerInitiatedWith3seAC
    OBJECT IDENTIFIER ::= { id-ac inShadowConsumerInitiatedWith3seAC(20) version1(0)}
-- SCF/SCF Application Context
id-ac-scf-scfOperationsAC
    OBJECT IDENTIFIER ::= {id-ac scf-scfOperationsAC(21) version1(0)}
```

```
id-ac-distributedSCFSystemAC
    OBJECT IDENTIFIER ::= {id-ac distributedSCFSystemAC(22) version1(0)}
id-ac-scf-scfOperationsWith3seAC
    OBJECT IDENTIFIER ::= {id-ac scf-scfOperationsWith3seAC(23) version1(0)}
id-ac-distributedSCFSystemWith3seAC
    OBJECT IDENTIFIER ::= {id-ac distributedSCFSystemWith3seAC(24) version1(0)}
-- CUSF/SCF Application Context
id-ac-cs3scfcusfGeneric     OBJECT IDENTIFIER ::= {id-ac scf-cusf-Generic(29) version1(0)}
id-ac-cs3cusfscfGeneric     OBJECT IDENTIFIER ::= {id-ac cusf-scf-Generic(30) version1(0)}
-- Attributes
-- SCF/SDF attributes
id-at-securityFacilityId    OBJECT IDENTIFIER ::= {id-at securityFacilityId(1)}
id-at-secretKey     OBJECT IDENTIFIER ::= {id-at secretKey(2)}
id-at-identifierList    OBJECT IDENTIFIER ::= {id-at identifierList(3)}
id-at-bindLevelIfOK OBJECT IDENTIFIER ::= {id-at bindLevelIfOK (4)}
id-at-lockSession       OBJECT IDENTIFIER ::= {id-at lockSession(5)}
id-at-failureCounter    OBJECT IDENTIFIER ::= {id-at failureCounter(6)}
id-at-maxAttempts       OBJECT IDENTIFIER ::= {id-at maxAttempts(7)}
id-at-currentList       OBJECT IDENTIFIER ::= {id-at currentList(8)}
id-at-stockId       OBJECT IDENTIFIER ::= {id-at stockId(9)}
id-at-source            OBJECT IDENTIFIER ::= {id-at source(10)}
id-at-sizeOfRestocking  OBJECT IDENTIFIER ::= {id-at sizeOfRestocking(11)}
id-at-challengeResponse OBJECT IDENTIFIER ::= {id-at challengeResponse(12)}
-- Abstract Syntaxes
-- SSF/SCF Abstract Syntaxes
id-as-ssf-scfGenericAS          OBJECT IDENTIFIER ::= {id-as ssf-scfGenericAS(4)}
id-as-assistHandoff-ssf-scfAS       OBJECT IDENTIFIER ::= {id-as assistHandoff-ssf-scfAS(6)}
id-as-scf-ssfGenericAS          OBJECT IDENTIFIER ::= {id-as scf-ssfGenericAS(7)}
id-as-scf-ssfTrafficManagementAS    OBJECT IDENTIFIER ::= {id-as scf-ssfTrafficManagementAS(9)}
id-as-scf-ssfServiceManagementAS    OBJECT IDENTIFIER ::= {id-as scf-ssfServiceManagementAS(10)}
id-as-ssf-scfServiceManagementAS    OBJECT IDENTIFIER ::= {id-as ssf-scfServiceManagementAS(11)}
id-as-scf-ssfStatusReportingAS      OBJECT IDENTIFIER ::= {id-as scf-ssfStatusReportingAS(12)}
id-as-scf-ssfTriggerManagementAS    OBJECT IDENTIFIER ::= {id-as scf-ssfTriggerManagementAS(13)}
-- SRF/SCF Abstract Syntaxes
id-as-basic-srf-scf OBJECT IDENTIFIER ::= { id-as basic-srf-scf(14)}
id-as-basic-scf-srf OBJECT IDENTIFIER ::= { id-as basic-scf-srf(15)}
-- SCF-SDF -  abstract syntaxes --
id-as-indirectoryOperationsAS   OBJECT IDENTIFIER ::= {id-as indirectoryOperationsAS(1)}
id-as-indirectoryBindingAS  OBJECT IDENTIFIER ::= {id-as indirectoryBindingAS(2)}
id-as-inExtendedDirectoryOperationsAS
                    OBJECT IDENTIFIER ::= {id-as inExtendedDirectoryOperationsAS(3) }
id-as-inSESEAS          OBJECT IDENTIFIER ::= {id-as inSESEAS(25) }
id-as-tfcOperationsAS       OBJECT IDENTIFIER ::= {id-as tfcOperationsAS(26)}
id-as-tfcBindingAS      OBJECT IDENTIFIER ::= {id-as tfcBindingAS(27)}
-- SDF-SDF -  abstract syntaxes
id-as-indirectorySystemAS       OBJECT IDENTIFIER ::= { id-as indirectorySystemAS(16) }
id-as-indirectoryDSABindingAS   OBJECT IDENTIFIER ::= { id-as indirectoryDSABindingAS(17) }
id-as-indirectoryShadowAS       OBJECT IDENTIFIER ::= { id-as indirectoryShadowAS(18) }
id-as-indsaShadowBindingAS      OBJECT IDENTIFIER ::= { id-as indsaShadowBindingAS(19) }
-- SCF/SCF Abstract Syntaxes
id-as-scf-scfOperationsAS       OBJECT IDENTIFIER ::= {id-as scf-scfOperationsAS(20)}
id-as-distributedSCFSystemAS        OBJECT IDENTIFIER ::= {id-as distributedSCFSystemAS(21)}
id-as-scf-scfBindingAS          OBJECT IDENTIFIER ::= {id-as scf-scfBindingAS(22)}
-- CUSF/SCF Abstract Syntaxes
id-as-cs3scfcusfGeneric     OBJECT IDENTIFIER ::= {id-as scf-cusf-Generic(28) }
id-as-cs3cusfscfGeneric     OBJECT IDENTIFIER ::= {id-as cusf-scf-Generic(29) }
-- Object Class
-- for SCF-SDF interface, Object Class
id-oc-securityUserInfo      OBJECT IDENTIFIER ::= { id-oc securityUserInfo(1)}
id-oc-tokensStock           OBJECT IDENTIFIER ::= { id-oc tokenStock(2)}
-- Methods
-- for SCF-SDF interface, Methods
id-mt-verifyCredentials     OBJECT IDENTIFIER ::= { id-mt verifyCredentials(1)}
id-mt-conformCredentials    OBJECT IDENTIFIER ::= { id-mt conformCredentials(2)}
id-mt-provideTokens     OBJECT IDENTIFIER ::= { id-mt provideTokens(3)}
id-mt-fillSecurityTokens        OBJECT IDENTIFIER ::= { id-mt fillSecurityTokens(4)}
-- Security Facility
-- for SCF-SDF interface, Security Facility
id-sf-pwd           OBJECT IDENTIFIER ::= {id-sf pwd(1)}
id-sf-challengeResponse     OBJECT IDENTIFIER ::= {id-sf challengeResponse(2)}
id-sf-onAirSubscription     OBJECT IDENTIFIER ::= {id-sf onAirSubscription(3)}
-- for SDF-SDF interface, SDF  Attributes
id-soa-methodRuleUse    OBJECT IDENTIFIER ::= {id-soa methodRuleUse(1)}
id-aca-prescriptiveACI  OBJECT IDENTIFIER ::= { id-aca prescriptiveACI(4) }
id-aca-entryACI         OBJECT IDENTIFIER ::= { id-aca entryACI(5) }
id-aca-subentryACI  OBJECT IDENTIFIER ::= { id-aca subentryACI(6) }
-- for ac, as, rosObject, contract and package, the values are identical to Q1228
-- ROS Objects
id-rosObject-scf    OBJECT IDENTIFIER ::= {id-rosObject scf(1)}
id-rosObject-ssf    OBJECT IDENTIFIER ::= {id-rosObject ssf(2)}
id-rosObject-srf    OBJECT IDENTIFIER ::= {id-rosObject srf(3)}
id-rosObject-sdf    OBJECT IDENTIFIER ::= {id-rosObject sdf(4)}
id-rosObject-cusf   OBJECT IDENTIFIER ::= {id-rosObject cusf(5)}
```

```
-- Contracts
-- SSF/SCF Contracts
id-inCs3SsfToScfGeneric     OBJECT IDENTIFIER ::= {id-contract inCs3SsfToScfGeneric(3)}
id-inCs3AssistHandoffSsfToScf   OBJECT IDENTIFIER ::= {id-contract inCs3AssistHandoffSsfToScf(5)}
id-inCs3ScfToSsfGeneric     OBJECT IDENTIFIER ::= {id-contract inCs3ScfToSsfGeneric(6)}
id-inCs3ScfToSsfTrafficManagement
                OBJECT IDENTIFIER ::= {id-contract inCs3ScfToSsfTrafficManagement(8)}
id-inCs3ScfToSsfServiceManagement
                OBJECT IDENTIFIER ::= {id-contract inCs3ScfToSsfServiceManagement(9)}
id-inCs3SsfToScfServiceManagement
                OBJECT IDENTIFIER ::= {id-contract inCs3SsfToScfServiceManagement(10)}
id-inCs3ScfToSsfStatusReporting
                OBJECT IDENTIFIER ::= {id-contract inCs3ScfToSsfStatusReporting(11)}
id-inCs3ScfToSsfTriggerManagement
                OBJECT IDENTIFIER ::= {id-contract inCs3ScfToSsfTriggerManagement(12)}
-- SRF/SCF Contracts
id-contract-srf-scf     OBJECT IDENTIFIER ::= {id-contract srf-scf(13)}
-- SCF-SDF contracts --
id-contract-dap         OBJECT IDENTIFIER ::= {id-contract dap(1)}
id-contract-dapExecute  OBJECT IDENTIFIER ::= {id-contract dapExecute(2) }
id-contract-tfc     OBJECT IDENTIFIER ::= {id-contract tfc(22)}
-- SDF - SDF Contracts.
id-contract-indsp           OBJECT IDENTIFIER ::= { id-contract indsp(14) }
id-contract-shadowConsumer  OBJECT IDENTIFIER ::= { id-contract shadowConsumer(15) }
id-contract-shadowSupplier  OBJECT IDENTIFIER ::= { id-contract shadowSupplier(17) }
-- SCF/SCF Contracts
id-contract-scf-scf         OBJECT IDENTIFIER ::= {id-contract scf-scf(18)}
id-contract-dssp            OBJECT IDENTIFIER ::= {id-contract dssp(19)}
-- CUSF/SCF Contracts
id-contract-cs3scfcusfGeneric       OBJECT IDENTIFIER ::= {id-contract scf-cusf-Generic(23) }
id-contract-cs3cusfscfGeneric       OBJECT IDENTIFIER ::= {id-contract cusf-scf-Generic(24) }
-- Operation Packages
id-package-emptyConnection      OBJECT IDENTIFIER ::= { id-package emptyConnection(60)}
-- SSF/SCF Operation Packages
id-package-scfActivation        OBJECT IDENTIFIER ::= {id-package scfActivation(11)}
id-package-srf-scfActivationOfAssist
    OBJECT IDENTIFIER ::= {id-package srf-scfActivationOfAssist(15)}
id-package-assistConnectionEstablishment
    OBJECT IDENTIFIER ::= {id-package assistConnectionEstablishment(16)}
id-package-genericDisconnectResource
    OBJECT IDENTIFIER ::= {id-package genericDisconnectResource(17)}
id-package-nonAssistedConnectionEstablishment
    OBJECT IDENTIFIER ::= {id-package nonAssistedConnectionEstablishment(18)}
id-package-connect          OBJECT IDENTIFIER ::= {id-package connect(19)}
id-package-callHandling         OBJECT IDENTIFIER ::= {id-package callHandling(20)}
id-package-bcsmEventHandling    OBJECT IDENTIFIER ::= {id-package bcsmEventHandling(21)}
id-package-dpSpecificEventHandling  OBJECT IDENTIFIER ::= {id-package dpSpecificEventHandling(22)}
id-package-chargingEventHandling    OBJECT IDENTIFIER ::= {id-package chargingEventHandling(23)}
id-package-ssfCallProcessing        OBJECT IDENTIFIER ::= {id-package ssfCallProcessing(24)}
id-package-scfCallInitiation        OBJECT IDENTIFIER ::= {id-package scfCallInitiation(25)}
id-package-timer                OBJECT IDENTIFIER ::= {id-package timer (26)}
id-package-billing              OBJECT IDENTIFIER ::= {id-package billing(27)}
id-package-charging         OBJECT IDENTIFIER ::= {id-package charging(28)}
id-package-trafficManagement        OBJECT IDENTIFIER ::= {id-package trafficManagement(29)}
id-package-serviceManagementActivate
    OBJECT IDENTIFIER ::= {id-package serviceManagementActivate(30)}
id-package-serviceManagementResponse
    OBJECT IDENTIFIER ::= {id-package serviceManagementResponse(31)}
id-package-callReport       OBJECT IDENTIFIER ::= {id-package callReport(32)}
id-package-signallingControl    OBJECT IDENTIFIER ::= {id-package signallingControl(33)}
id-package-activityTest     OBJECT IDENTIFIER ::= {id-package activityTest(34)}
id-package-statusReporting  OBJECT IDENTIFIER ::= {id-package statusReporting(35)}
id-package-cancel           OBJECT IDENTIFIER ::= {id-package cancel(36)}
id-package-cphResponse      OBJECT IDENTIFIER ::= {id-package cphResponse(37)}
id-package-entityReleased   OBJECT IDENTIFIER ::= {id-package entityReleased(38)}
id-package-triggerManagement OBJECT IDENTIFIER ::= {id-package triggerManagement(39)}
id-package-uSIHandling      OBJECT IDENTIFIER ::= {id-package uSIHandling(40)}
id-package-triggerCallManagement OBJECT IDENTIFIER ::= {id-package triggerCallManagement(63)}
-- SRF/SCF Operation Packages
id-package-specializedResourceControl
                    OBJECT IDENTIFIER ::= { id-package specializedResourceControl(42)}
id-package-srf-scfCancel        OBJECT IDENTIFIER ::= { id-package srf-scfCancel(43)}
id-package-messageControl   OBJECT IDENTIFIER ::= { id-package messageControl(44)}
id-package-scriptControl    OBJECT IDENTIFIER ::= { id-package scriptControl(45)}
id-package-srfManagement    OBJECT IDENTIFIER ::= { id-package srfManagement(66)}
-- SCF-SDF packages --
id-package-search           OBJECT IDENTIFIER ::= {id-package search(2)}
id-package-modify       OBJECT IDENTIFIER ::= {id-package modify(3)}
id-package-dapConnection    OBJECT IDENTIFIER ::={id-package dapConnection(10)}
id-package-execute      OBJECT IDENTIFIER ::={id-package execute(4) }
id-package-tfcOperations    OBJECT IDENTIFIER ::= {id-package tfcOperations(64)}
id-package-tfcConnection    OBJECT IDENTIFIER ::= {id-package tfcConnection(65)}
-- SDF - SDF Packages.
```

```
id-package-dspConnection    OBJECT IDENTIFIER ::= { id-package dspConnection(47) }
id-package-inchainedModify  OBJECT IDENTIFIER ::= { id-package inchainedModify(48) }
id-package-inchainedSearch  OBJECT IDENTIFIER ::= { id-package inchainedSearch(49) }
id-package-chainedExecute   OBJECT IDENTIFIER ::= { id-package chainedExecute(50) }
id-package-dispConnection   OBJECT IDENTIFIER ::= { id-package dispConnection(51) }
id-package-shadowConsumer   OBJECT IDENTIFIER ::= { id-package shadowConsumer(52) }
id-package-shadowSupplier   OBJECT IDENTIFIER ::= { id-package shadowSupplier (53) }
-- SCF/SCF Operation Packages
id-package-scf-scfConnection        OBJECT IDENTIFIER ::= {id-package scf-scfConnection(46)}
id-package-dsspConnection       OBJECT IDENTIFIER ::= {id-package dsspConnection(74)}
id-package-handlingInformation  OBJECT IDENTIFIER ::= {id-package handlingInformation(54)}
id-package-notification         OBJECT IDENTIFIER ::= {id-package notification(55)}
id-package-chargingInformation  OBJECT IDENTIFIER ::= {id-package chargingInformation(56)}
id-package-userInformation      OBJECT IDENTIFIER ::= {id-package userInformation(57)}
id-package-networkCapability        OBJECT IDENTIFIER ::= {id-package networkCapability(58)}
id-package-chainedSCFOperations OBJECT IDENTIFIER ::= {id-package chainedSCFOperations(59)}
-- CUSF/SCF Operation Packages
id-package-cusfTDPGenericInvocation
    OBJECT IDENTIFIER ::= {id-package cusfTDPGenericInvocation(62) }
id-package-cusfGenericEventHandling
    OBJECT IDENTIFIER ::= {id-package cusfGenericEventHandling (68) }
id-package-cusfSCFInitiation    OBJECT IDENTIFIER ::= {id-package cusfSCFInitiation(70) }
id-package-cusfContinue     OBJECT IDENTIFIER ::= {id-package cusfContinue(71) }
id-package-cusfConnect      OBJECT IDENTIFIER ::= {id-package cusfConnect(72) }
id-package-cusfRelease      OBJECT IDENTIFIER ::= {id-package cusfRelease(73) }
-- SDF  Attribute Value Contexts
id-avc-assignment           OBJECT IDENTIFIER ::= {id-avc assignment(1)}
id-avc-basicService         OBJECT IDENTIFIER ::= {id-avc assignment(2)}
id-avc-lineIdentity         OBJECT IDENTIFIER ::= {id-avc assignment(3)}
END
```

12.2 Common Data Types

```
IN-CS3-common-datatypes { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
cs3(30) modules(1) in-cs3-common-datatypes (1) version1(0)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS
    common-classes
FROM  IN-CS3-object-identifiers { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) cs3(30) modules(1) in-cs3-object-identifiers(0) version1(0) }

    EXTENSION,
    COMMON-BOUNDS,
    SupportedExtensions
FROM IN-CS3-common-classes common-classes;
CriticalityType ::= ENUMERATED {
        ignore(0),
        abort(1)
        }
Extensions {COMMON-BOUNDS : b1} ::= SEQUENCE SIZE (1..b1.&numOfExtensions) OF ExtensionField
ExtensionField::= SEQUENCE {
    type                        EXTENSION.&id ({SupportedExtensions}),
            -- shall identify the value of an EXTENSION type
    criticality                 CriticalityType DEFAULT ignore,
    value                       [1] EXTENSION.&ExtensionType
                                 ({SupportedExtensions}{@type})
    }
--This parameter indicates an extension of an argument data type. Its content is network operator
specific
Integer4 ::= INTEGER(0..2147483647)
InvokeID ::= INTEGER (-128..127)
UnavailableNetworkResource ::= ENUMERATED {
     unavailableResources(0),
     componentFailure(1),
     basicCallProcessingException(2),
     resourceStatusFailure(3),
     endUserFailure(4),
     screening(5)
     }
-- Indicates the network resource that failed
-- Note that in IN CS3 the screening value can only be used by the SCF.
END
```

12.3 Operation codes

The following ASN.1 module defines the operation codes which are allocated to each of the operations specified in EN 301 931-2 to EN 301 931-4, except those imported from the Directory Abstract Service as defined in ITU-T Recommendation X.519 [69] .

```
IN-CS3-operationcodes { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
cs3(30) modules(1) in-cs3-operationcodes(3) version1(0)}
DEFINITIONS ::=
BEGIN
IMPORTS
    ros-InformationObjects
FROM  IN-CS3-object-identifiers { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) cs3(30) modules(1) in-cs3-object-identifiers(0) 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
-- establishment ASE;
        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-selectFacility                   Code ::= local : 30
        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
-- Status reporting Package
        opcode-requestCurrentStatusReport           Code ::= local : 37
        opcode-requestEveryStatusChangeReport       Code ::= local : 38
        opcode-requestFirstStatusMatchReport        Code ::= local : 39
        opcode-statusReport                 Code ::= local : 40
-- 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
        opcode-cancelStatusReportRequest            Code ::= local : 54
-- 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
        opcode-createOrRemoveTriggerData            Code ::= local : 135
-- Trigger Call Management Package
        opcode-setServiceProfile                    Code ::= local : 136
-- 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
        opcode-srfCallGap                   Code ::= local : 139
-- 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
        opcode-runUserScript                    Code ::= local : 140
        opcode-transferSTSI                 Code ::= local : 141
-- SCF/SDF interface
        opcode-execute                      Code ::= local : 10
```

```
        opcode-trafficFlowControl               Code ::= local : 138
-- CUSF/SCF interface
        opcode-initiateAssociation              Code ::= local : 123
        opcode-releaseAssociation               Code ::= local : 126
        opcode-requestReportBCUSMEvent          Code ::= local : 127
        opcode-connectAssociation               Code ::= local : 132
        opcode-continueAssociation              Code ::= local : 133
        opcode-eventReportBCUSM             Code ::= local : 134
        opcode-initialAssociationDP             Code ::= local : 131
END
```

12.4 Errors

12.4.1 Error types

The following ASN.1 module defines the error types used by operations specified in EN 301 931-2 to EN 301 931-4, except those imported from the Directory Abstract Service as defined in ITU-T Recommendation X.519 [69] .

```
IN-CS3-errortypes { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) cs3(30)
modules(1) in-cs3-errortypes(2) version1(0)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS
    ros-InformationObjects,
    common-datatypes,
    errorcodes,
    sdf-sdf-Operations,
    scf-scf-Operations,
    ds-UsefulDefinitions,
    spkmGssTokens,
    tc-Messages
FROM  IN-CS3-object-identifiers { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) cs3(30) modules(1) in-cs3-object-identifiers(0) version1(0) }
    directoryAbstractService,
    distributedOperations,
    enhancedSecurity
FROM UsefulDefinitions ds-UsefulDefinitions
    OPTIONALLY-PROTECTED{}, DIRQOP
FROM EnhancedSecurity enhancedSecurity
    CommonResults,
    ServiceProblem,
    SecurityProblem,
    SecurityParameters,
    Versions
FROM DirectoryAbstractService directoryAbstractService
    SCFQOP
FROM IN-CS3-SCF-SCF-Operations scf-scf-Operations
    AccessPointInformation
FROM DistributedOperations distributedOperations
    SPKM-ERROR
FROM SpkmGssTokens spkmGssTokens
    ERROR
FROM Remote-Operations-Information-Objects ros-InformationObjects
    InvokeID,
    UnavailableNetworkResource
FROM IN-CS3-common-datatypes common-datatypes
    errcode-canceled,
    errcode-cancelFailed,
    errcode-chainingRefused,
    errcode-eTCFailed,
    errcode-executionError,
    errcode-improperCallerResponse,
    errcode-missingCustomerRecord,
    errcode-missingParameter,
    errcode-parameterOutOfRange,
    errcode-requestedInfoError,
    errcode-scfTaskRefused,
    errcode-scfReferral,
    errcode-systemFailure,
    errcode-taskRefused,
    errcode-unavailableResource,
    errcode-unexpectedComponentSequence,
    errcode-unexpectedDataValue,
    errcode-unexpectedParameter,
    errcode-unknownLegID,
    errcode-unknownResource,
    errcode-unknownSubscriber
FROM IN-CS3-errorcodes errorcodes;
-- TYPE DEFINITION FOR  IN CS3  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.
executionError ERROR ::= {
    PARAMETER OPTIONALLY-PROTECTED{
            SET {
            problem [0] EXPLICIT ExecutionProblem,
            COMPONENTS OF CommonResults },
            DIRQOP.&dirErrors-QOP{@dirqop}
            }
    CODE    errcode-executionError
    }
-- The executionError is returned by an Execute operation in the case of the operation is not completing.
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.
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 FFS
            }
    CODE    errcode-requestedInfoError
    }
-- The requested information cannot be found.
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 FFS
                            },
        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}
    }
systemFailure ERROR ::= {
    PARAMETER   UnavailableNetworkResource
    CODE    errcode-systemFailure
    }
-- The operation could not be completed due to e.g. a system failure at the serving physical entity, the
-- unavailability of the required resource or due to screening.
taskRefused ERROR ::= {
    PARAMETER   ENUMERATED {
                generic(0),
                unobtainable (1),
                congestion(2)
                --other values FFS
                            }
    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.
tfcBindError ERROR ::= {
    PARAMETER   SET {
            versions    [0] Versions DEFAULT {v1},
            error   CHOICE{
                serviceError    [1] ServiceProblem,
                securityError   [2] SecurityProblem}}
    }
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.
unknownSubscriber ERROR ::= {
    CODE    errcode-unknownSubscriber
    }
-- Subscriber whose status is being requested is not known to the serving entity.
END
```

12.4.2 Error codes

The following ASN.1 module defines the error codes which are allocated to each of the errors specified in EN 301 931-2 to EN 301 931-4, except those imported from the Directory Abstract Service as defined in ITU-T Recommendation X.519 [69] .

```
IN-CS3-errorcodes { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-network(1) cs3(30)
modules(1) in-cs3-errorcodes(4) version1(0)}
DEFINITIONS ::=
BEGIN
IMPORTS
    ros-InformationObjects
FROM  IN-CS3-object-identifiers { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) cs3(30) modules(1) in-cs3-object-identifiers(0) 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 IN CS2 error types follows
        errcode-scfReferral             Code ::= local : 21
        errcode-scfTaskRefused              Code ::= local : 22
        errcode-chainingRefused             Code ::= local : 23
        errcode-executionError              Code ::= local :10
        errcode-unknownSubscriber           Code::= local 24
END
```

12.5 Common classes

```
IN-CS3-common-classes { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-network(1)
cs3(30) modules(1) in-cs3-common-classes(5) version1(0)}
DEFINITIONS ::=
BEGIN
IMPORTS
    id-package-emptyConnection,
    id-rosObject-scf,
    id-rosObject-cusf,
    id-rosObject-sdf,
    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,
    sdf-sdf-Protocol,
    common-datatypes
FROM IN-CS3-object-identifiers { itu-t(0) identified-organization(4) etsi(0) inDomain(1) in-
network(1) cs3(30) modules(1) in-cs3-object-identifiers (0) version1(0)}
    ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, Code, OPERATION,
    CONNECTION-PACKAGE
FROM Remote-Operations-Information-Objects ros-InformationObjects
    emptyBind
FROM Remote-Operations-Useful-Definitions ros-UsefulDefinitions
    inCs3AssistHandoffSsfToScf,
    inCs3ScfToSsfGeneric,
    inCs3ScfToSsfStatusReporting,
    inCs3ScfToSsfServiceManagement,
    inCs3ScfToSsfTrafficManagement,
    inCs3ScfToSsfTriggerManagement,
    inCs3SsfToScfGeneric,
    inCs3SsfToScfServiceManagement
FROM IN-CS3-SSF-SCF-pkgs-contracts-acs ssf-scf-Protocol
    cs3cusfscfGenericContract,
    cs3scfcusfGenericContract
```

```
FROM IN-CS3-SCF-CUSF-pkgs-contracts-acs scf-cusf-Protocol
    dsspContract,
    scf-scfContract
FROM IN-CS3-SCF-SCF-pkgs-contracts-acs scf-scf-Protocol
    srf-scf-contract
FROM IN-CS3-SCF-SRF-pkgs-contracts-acs scf-srf-Protocol
    dapContract,
    dapExecuteContract,
    tfcContract
FROM IN-CS3-SCF-SDF-Protocol scf-sdf-Protocol
    indspContract,
    shadowConsumerContract,
    shadowSupplierContract
FROM IN-CS3-SDF-SDF-Protocol sdf-sdf-Protocol
    CriticalityType
FROM IN-CS3-common-datatypes common-datatypes
;
ssf ROS-OBJECT-CLASS ::= {
    INITIATES {inCs3SsfToScfGeneric|
               inCs3AssistHandoffSsfToScf|
               inCs3SsfToScfServiceManagement}
    RESPONDS     {inCs3ScfToSsfGeneric|
               inCs3ScfToSsfTrafficManagement|
               inCs3ScfToSsfServiceManagement|
               inCs3ScfToSsfTriggerManagement|
               inCs3ScfToSsfStatusReporting}
    ID        id-rosObject-ssf}
-- The ssf class of ROS-object describes the communication capabilities of an SSF
-- This object can act as the initiator of the following contracts
--
-- inCs3SsfToScfGeneric contract expresses the form of the service in which the SSF,
-- a ROS-object of class ssf, initiates the generic triggering approach contract.
-- This dialogue is initiated by the SSF with the InitialDP Operation.
-- inCs3AssistHandoffSsfToScf contract expresses the form of the service in which the SSF,
-- a ROS-object of class ssf, initiates the Assist or Hand-off contract.
-- This dialogue is initiated by the SSF with the AssistRequestInstructions Operation.
-- inCs3SsfToScfServiceManagement contract expresses the form of the service in which the SSF,
-- a ROS-object of class ssf, initiates ServiceManagement related contract for reporting
-- service Management results. This dialogue is initiated/ended by the SSF with
-- the ServicefilteringResponse Operation.
--
-- This object can act as the responder of the following contracts
--
-- inCs3ScfToSsfGeneric contract expresses the form of the service in which the SSF,
-- a ROS-object of class ssf, responds to the generic messaging approach for
-- the SCF Initiate Call Attempt contract. This dialogue is initiated by the SCF with
-- the InitiateCallAttempt or CreateCallSegmentAssociation, Generic case.
-- inCs3ScfToSsfTrafficManagement contract expresses the form of service in which the SSF,
-- a ROS object of class ssf, responds to the Traffic Management related contract.
-- This dialogue is initiated by the SCF with the CallGap Operation
-- inCs3ScfToSsfServiceManagement contract expresses the form of service in which the SSF,
-- a ROS object of class ssf, responds to the Service Management related contract.
--  This dialogue is initiated  by the SCF with the ActivateServiceFiltering Operation
-- inCs3ScfToSsfTriggerManagement contract expresses the form of service in which the SSF,
-- a ROS object of class ssf, responds to the Trigger Management related contract.
-- This dialogue is initiated by the SCF with the ManageTriggerData Operation
-- inCs3ScfToSsfStatusReporting contract expresses the form of service in which the SSF,
-- a ROS object of class ssf, responds to the Status Reporting related contract.
-- This dialogue is initiated by the SCF with the StatusReporting Operations.
srf ROS-OBJECT-CLASS ::= {
    INITIATES {srf-scf-contract}
    ID        id-rosObject-srf
    }
-- The srf class of ROS-object describes the communication capabilities of an SRF
-- This object can act as the initiator of the following contract
--
-- srf-scf-contract contract expresses the form of service in which the SRF, a ROS-object of class
srf,
-- initiates the srf related contract. This dialogue is initiated by the SRF with
-- the AssistRequestInstruction Operation
cusf ROS-OBJECT-CLASS ::= {
    INITIATES   {       Cs3cusfscfGenericContract}
    RESPONDS    {       cs3scfcusfGenericContract }
    ID  id-rosObject-cusf}
-- The cusf class of ROS-object describes the communication capabilities of an CUSF
-- This object can act as the initiator of the following contract
--
-- cs3cusfscfGenericContract  expresses the form of the service in which the CUSF,
-- a ROS-object of class cusf, initiates the  generic approach contract by using
-- an InitialAssociationDP operation.
```

```
--
-- This object can act as the responder of the following contract
--
-- cs3scfcusfGenericContract expresses the form of the contract in which the CUSF,
--  a ROS-object of class cusf, responds the generic  approach contract initiates by the SCF
-- using an InitiateAssociation operation.
scf ROS-OBJECT-CLASS ::= {
     INITIATES {inCs3ScfToSsfGeneric|
                inCs3ScfToSsfTrafficManagement|
                inCs3ScfToSsfServiceManagement|
                inCs3ScfToSsfTriggerManagement|
                inCs3ScfToSsfStatusReporting |
-- scf to sdf contracts
                dapContract|
                dapExecuteContract|
-- scf to scf contracts
                scf-scfContract |
                dsspContract |
-- tfc contract (scf to scf)
                tfcContract|
-- scf to cusf contracts
                cs3scfcusfGenericContract
                }
     RESPONDS       {inCs3SsfToScfGeneric|
                inCs3AssistHandoffSsfToScf|
                inCs3SsfToScfServiceManagement|
-- srf to scf contracts
                srf-scf-contract |
-- tfc contract (scf to scf, sdf to scf)
                tfcContract|
-- scf to scf contracts
                scf-scfContract |
                dsspContract|
-- cusf to scf contracts
                cs3cusfscfGenericContract
                }
     ID    id-rosObject-scf}
-- The scf class of ROS-object describes the communication capabilities of an SCF
-- This object can act as the initiator of the following contracts
--
-- scf to ssf contracts
-- inCs3ScfToSsfGeneric 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. This dialogue is initiated  by the SCF with the
InitiateCallAttempt
-- or CreateCallSegmentAssociation, Generic case.
-- inCs3ScfToSsfTrafficManagement contract expresses the form of service in which the SCF,
-- a ROS object of class scf, initiates the Traffic Management related contract. This dialogue is
initiated
-- by the SCF with the CallGap Operation
-- inCs3ScfToSsfServiceManagement contract expresses the form of service in which the SCF,
-- a ROS object of class  scf, initiates the Service Management related contract.
-- This dialogue is initiated  by the SCF with the ActivateServiceFiltering Operation
-- inCs3ScfToSsfTriggerManagement contract expresses the form of service in which the SCF,
--  a ROS object of class scf, initiates the Trigger Management related contract.
-- This dialogue is initiated by the SCF with the ManageTriggerData Operation
-- inCs3ScfToSsfStatusReporting contract expresses the form of service in which the SCF,
-- a ROS object of class scf, initiates the Status Reporting related contract. This dialogue is
initiated
-- by the SCF with the StatusReporting Operations.
--
-- scf to sdf
-- dapContract contract expresses the form of service in which the SCF, a ROS object of class scf,
-- initiates the SCF/SDF message exchange based on a DAP protocol (Search operation and Directory
-- Modify operations).
-- dapExecuteContract contract expresses the form of service in which the SCF, a ROS object of class
scf,
-- initiates the SCF/SDF message exchange based on a DAP protocol (Search operation and Directory
-- Modify operations) plus the Execute operation.
-- tfc contract (scf to scf)
-- tfcContract contract expresses the form of service in which the SCF, a ROS object of class scf,
-- initiates the traffic flow control mechanism.
--
-- scf to scf contracts
-- scf-scfContract contract expresses the form of service in which the SCF, a ROS object of class
scf,
-- initiates the SCF/SCF message exchange.
-- dsspContract contract expresses the form of service in which the SCF, a ROS object of class scf,
```

```
-- initiates the chained SCF/SCF message exchange.
--
-- scf to cusf contracts
-- cs3scfcusfGenericContract expresses the form of the contract in which the SCF, a ROS-object of
class scf,
-- initiates the generic approach contract by using an InitiateAssociation operation.
--
-- This object can act as the responder of the following contracts
--
-- ssf to scf contracts
-- inCs3SsfToScfGeneric contract expresses the form of the service in which the SCF,
-- a ROS-object of class scf, responds to the generic triggering approach contract.
-- This dialogue is initiated by the SSF with the InitialDP Operation.
-- inCs3AssistHandoffSsfToScf contract expresses the form of the service in which the SCF,
-- a ROS-object of class scf, responds to the Assist or Hand-off contract.
-- This dialogue is initiated by the SSF with the AssistRequestInstructions Operation.
-- inCs3SsfToScfServiceManagement contract expresses the form of the service in which the SCF,
--  a ROS-object of class  scf, responds to the ServiceManagement related contract for reporting
-- Service Management results.
-- This dialogue is initiated/ended by the SSF with the ServicefilteringResponse Operation.
--
-- srf to scf contracts
-- srf-scf-contract contract expresses the form of service in which the SCF, a ROS-object of class
scf,
--  responds to the srf related contract. This dialogue is initiated by the SRF with the
AssistRequestInstruction
-- tfc contract (scf to scf, sdf to scf)
-- tfcContract contract expresses the form of service in which the SCF, a ROS object of class scf,
-- responds to the traffic flow control contract initiated either by the SCF or the SDF.
--
-- scf to scf contracts
-- scf-scfContract contract expresses the form of service in which the SCF, a ROS object of class
scf,
-- responds to the previously initiated SCF/SCF message exchange.
-- dsspContract contract expresses the form of service in which the SCF, a ROS object of class scf,
-- responds to the previously initiated chained SCF/SCF message exchange.
--
-- cusf to scf contracts
-- cs3cusfscfGenericContract  expresses the form of the service in which the SCF,
-- a ROS-object of class scf, responds to the generic approach contract.
sdf ROS-OBJECT-CLASS ::= {
    INITIATES   {indspContract|
                shadowConsumerContract|
                shadowSupplierContract|
                tfcContract
            }
    RESPONDS    {dapContract|
            dapExecuteContract|
            indspContract|
                shadowConsumerContract|
                shadowSupplierContract|
                tfcContract
            }
    ID  id-rosObject-sdf}
-- The sdf class of ROS-Object describes the communication capabilities of an SDF
-- This object can act as the initiator of the following contracts
-- indspContract contract expresses the form of service in which the SDF, a ROS object of class sdf,
-- initiates the chained SCF/SDF message exchange, based on the DSP protocol.
-- shadowConsumerContract contract expresses the form of service in which the SDF,
-- a ROS object of class sdf, initiates the shadowing mechanism as a shadow consumer,
-- based on the DISP protocol.
-- shadowSupplierContract contract expresses the form of service in which the SDF,
-- a ROS object of class sdf, initiates the shadowing mechanism as a shadow supplier,
-- based on the DISP protocol.
-- tfcContract contract expresses the form of service in which the SDF, a ROS object of class sdf,
-- initiates the traffic flow control mechanism.
-- This object can act as the responder of the following contracts
-- dapContract contract expresses the form of service in which the SDF, a ROS object of class sdf,
-- responds to the previously initiated SCF/SDF message exchange.
-- dapExecuteContract contract expresses the form of service in which the SDF, a ROS object of class
sdf,
-- responds to the previously initiated SCF/SDF message exchange.
-- indspContract contract expresses the form of service in which the SDF, a ROS object of class sdf,
-- responds to the previously initiated chained SCF/SDF message exchange.
-- shadowConsumerContract contract expresses the form of service in which the SDF,
-- a ROS object of class sdf, responds to the previously initiated shadowing mechanism..
-- shadowSupplierContract contract expresses the form of service in which the SDF,
-- a ROS object of class sdf, responds to the previously initiated shadowing mechanism.
-- tfcContract contract expresses the form of service in which the SDF, a ROS object of class sdf,
-- responds to the traffic flow control initiated by the SDF.
```

```
-- Definition of the extension class
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 section 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 ffs
-- 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   EXTENSION ::= {firstExtension , ...
-- full set of network operator extensions --}
-- SupportedExtension is the full set of the network operator extensions.
inUnbind OPERATION ::= {
      RETURN RESULT      FALSE
      ALWAYS RESPONDSFALSE }
emptyConnectionPackage CONNECTION-PACKAGE ::= {
      BIND        emptyBind
      UNBIND    inUnbind
      RESPONDER UNBIND       TRUE
      ID            id-package-emptyConnection
      }
EmptyReturnable OPERATION ::= {...}
COMMON-BOUNDS ::= CLASS
{     &numOfExtensions      INTEGER OPTIONAL}
WITH SYNTAX
{     [NUM-OF-EXTENSIONS     &numOfExtensions]}
-- The following instance of the parameter bound is just an example
networkSpecificBoundSet  COMMON-BOUNDS ::=
{     NUM-OF-EXTENSIONS      1}
END
```

Annex A (informative): List of non backward compatible changes between IN CS2 and IN CS3

A.1 InitiateAssociation operation has changed to a class 1 operation

In order to enable the SCF to provide instructions to the CUSF before an event is reported, the InitiateAssociation operation is changed from a class 2 to a class 1 operation.

A.2 Tag modification in the argument of the MoveCallSegments operation

In the IN ITU-T CS-2 Recommendation, two parameters have been allocated the same tag "2", as follows:

[extract of IN CS-2 ITU-T Recommendation Q.1228]

```
MoveCallSegmentsArg {PARAMETERS-BOUND : bound} ::= SEQUENCE {
    targetCallSegmentAssociation    [0] CSAID { bound},
-- assignement of CSAID by SSF/SCF is ffs.
    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,
    ...
    }
```

Therefore, in the IN CS-3 recommendation, the tag allocated to the “extensions” parameter is changed from 2 to **3**.

[extract of IN CS-3 Q.1238.2 Recommendation]

```
MoveCallSegmentsArg {PARAMETERS-BOUND : bound} ::= SEQUENCE {
    targetCallSegmentAssociation    [0] CSAID { bound},
    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      [3] SEQUENCE SIZE (1..bound.&numOfExtensions) OF ExtensionField {bound}
    OPTIONAL,
    ...
    }
```

A.3 Removal of the LegID parameter in the RequestedUTSI datatype

In IN CS-2, there is a duplication of LegID in the argument for RequestReportUTSI operation. The legID parameter is included both as the parameter on the top level of RequestReportUTSIArg and as a subparameter within the RequestedUTSIList, as follow.

[extract of IN CS-2 ITU-T Recommendation Q.1228]

```
RequestReportUTSIArg {PARAMETERS-BOUND : bound} ::= SEQUENCE {
    requestedUTSIList   [0] RequestedUTSIList { bound},
    extensions  [1] SEQUENCE SIZE(1..bound.&numOfExtensions) OF ExtensionField {bound}  OPTIONAL,
    legID   [2] LegID                   OPTIONAL,
    ...
    }
RequestedUTSIList {PARAMETERS-BOUND : bound} ::= SEQUENCE SIZE
    bound.&minRequestedUTSINum.. bound.&maxRequestedUTSINum) OF  RequestedUTSI {bound}
RequestedUTSI {PARAMETERS-BOUND : bound} ::= SEQUENCE {
    uSIServiceIndicator [0] USIServiceIndicator {bound},
    uSImonitorMode  [1] USIMonitorMode,
    legID   [2] LegID   DEFAULT sendingSideID:leg1
      }
```

In order to remove any ambiguity, in IN CS-3 the LegID parameter within the RequestedUTSI parameter is deleted, and the LegID parameter on the argument level is changed from OPTIONAL to DEFAULT leg 1 (to secure that the default LegID for the operation is not changed).

[extract of IN CS-3 ITU-T Recommendation Q.1238.2]

```
RequestReportUTSIArg {PARAMETERS-BOUND : bound} ::= SEQUENCE {
    requestedUTSIList   [0] RequestedUTSIList { bound},
    extensions  [1] SEQUENCE SIZE(1..bound.&numOfExtensions) OF ExtensionField {bound}  OPTIONAL,
    legID   [2] LegID   DEFAULT sendingSideID:leg1,
    ...
    }
RequestedUTSIList {PARAMETERS-BOUND : bound} ::= SEQUENCE SIZE
    bound.&minRequestedUTSINum.. bound.&maxRequestedUTSINum) OF  RequestedUTSI {bound}
RequestedUTSI {PARAMETERS-BOUND : bound} ::= SEQUENCE {
    uSIServiceIndicator [0] USIServiceIndicator {bound},
    uSImonitorMode  [1] USIMonitorMode,
    …
    }
```

A.4 New transition to "Stable call" CSCVS (Stable-2-Party, Stable-1-Party)

In IN CS-3, the transition to "stable call" CSCVS (Stable-2-Party, Stable-1-Party) occurs when an acknowledgement has been received from the remote side (i.e. O-TermSeized DP or CallAccepted DP have been detected).

In IN CS-2, the transition from Originating_Setup CSCVS (resp. Originating_1_Party_Setup CSCVS) to Stable-2-Party CSCVS (resp. Stable-1-Party CSCVS) occurs at the SendCall PIC, on the sending of the "Setup request" in the signalling interface. In IN CS3, this transition is postponed until O-TermSeized DP is detected, on receipt of the "Setup acknowledgement".

In the same way, the transition from TerminationSetup CSCVS to Stable-2-Party CSCVS is performed as soon as the CallAccepted DP is detected, on receipt of the "Setup acknowledgement" for IN CS-3 (and no more at the receipt of the "Setup confirmation").

A direct impact of this change is the refusal of a SplitLeg operation in the TerminatingSetup CSCVS.

Bibliography

The following material, though not yet publicly available, gives supporting information.

ITU-T Recommendation Q.1238: "Interface recommendation for intelligent network capability set 3".

ITU-T Recommendation Q.1229: "Intelligent network User's Guide for Capability Set 2".

History

Document history			
V1.1.1	August 2000	Public Enquiry	PE 20001215: 2000-08-16 to 2000-12-15