INTERNATIONAL TELECOMMUNICATION UNION

ITU-T Q.1218

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(10/95)

INTELLIGENT NETWORK

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CS-1

ITU-T Recommendation Q.1218

(Previously “CCITT Recommendation”)

FOREWORD

The ITU-T (Telecommunication Standardization Sector) is a permanent organ of the International Telecommunication Union (ITU). The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, March 1-12, 1993).

ITU-T Recommendation Q.1218 was revised by ITU-T Study Group 11 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 17th of October 1995.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

© ITU 1996

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

CONTENTS

Page

0 Introduction 1
0.1 Normative references 1
0.2 Definition methodology 2
0.3 Example physical scenarios 3
0.4 INAP protocol architecture 9
0.5 INAP addressing 11
0.6 Relationship between Recommendation Q.1214 and this Recommendation 11
0.7 Compatibility mechanisms used for INAP 13
1 SACF/MACF rules 14
1.1 Reflection of TCAP-AC 14
1.2 Sequential/Parallel execution of operations 14
2 Abstract syntax of the IN CS-1 Application Protocol 15
2.1 SSF-SCF, SCF-SRF Interface 15
2.1.1 IN CS-1 Operation Types 15
2.1.2 IN CS-1 Error Types 31
2.1.3 IN CS-1 Data Types 33
2.1.4 IN CS-1 application protocol (operation and error codes) 57
2.1.5 Application Contexts 65
2.2 SCF-SDF interface 67
2.2.1 Introduction to the IN X.500 DAP Subset 67
2.2.1.1 Alignment between the X.500 concepts and the IN 67
2.2.1.2 Use of a limited subset of X.500 68
2.2.1.3 Working assumptions 68
2.2.2 The IN X.500 DAP Subset 69
2.2.2.1 Review of X.511 for use in the IN 69
2.2.2.2 Directory Access Protocol Subset 76
2.2.2.3 X.501 profile 85
2.2.2.4 ASN.1 Profile of the Directory Abstract Service for the IN CS-1 85
3 Semantics 92
3.1 Definition of procedures and entities 92
3.1.1 SSF application entity procedures 92
3.1.1.1 General 92
3.1.1.2 Model and interfaces 92
3.1.1.3 Relations between SSF-FSM and the CCF and maintenance functions 93
3.1.1.4 SSF Management Finite State Machine (SSME-FSM) 95
3.1.1.5 SSF state transition diagram 97
3.1.1.6 Assisting/Handoff SSF-FSM 104
3.1.2 SCF application entity procedures 107
3.1.2.1 General 107
3.1.2.2 Model and interfaces 107
3.1.2.3 Relationship between the SCF-FSM and the SLPs/maintenance functions 108
3.1.2.4 Partial SCF Management Entity (SCME) State Transition Diagram 110
3.1.2.5 The SCF Call State Model (SCSM) 113

Page

3.1.3 SRF application entity procedures 128
3.1.3.1 General 128
3.1.3.2 Model and interfaces 129
3.1.3.3 Relationship between the SRF-FSM and maintenance functions/bearer connection handling 129
3.1.3.4 The SRSM 131
3.1.3.5 Example SRF control procedures 134
3.1.4 SDF application entity procedures 150
3.1.4.1 General 150
3.1.4.2 Model and interfaces 150
3.1.4.3 Relationship between the SDF-FSM and the maintenance function 150
3.1.4.4 SDF state transition model 150
3.2 Error procedures 153
3.2.1 Operation related error procedures 153
3.2.1.1 Attribute error 153
3.2.1.2 Cancelled 154
3.2.1.3 CancelFailed 155
3.2.1.4 ETCFailed 156
3.2.1.5 ImproperCallerResponse 156
3.2.1.6 MissingCustomerRecord 157
3.2.1.7 MissingParameter 159
3.2.1.8 Name error 162
3.2.1.9 ParameterOutOfRange 163
3.2.1.10 RequestedInfoError 164
3.2.1.11 Service error 164
3.2.1.12 Security error 165
3.2.1.13 SystemFailure 166
3.2.1.14 Task refused 168
3.2.1.15 UnavailableResource 169
3.2.1.16 UnexpectedComponentSequence 170
3.2.1.17 UnexpectedDataValue 171
3.2.1.18 UnexpectedParameter 173
3.2.1.19 UnknownLegID 174
3.2.1.20 UnknownResource 174
3.2.1.21 Update error 174
3.2.2 Entity related error procedures 175
3.2.2.1 Expiration of T_{SSF} 175
3.2.2.2 Expiration of T_{SRF} 176
3.3 Detailed Operation procedures 176
3.3.1 ActivateServiceFiltering procedure 176
3.3.1.1 General description 176
3.3.1.2 Invoking Entity (SCF) 179
3.3.1.3 Responding Entity (SSF) 180
3.3.2 ActivityTest procedure 181
3.3.2.1 General description 181
3.3.2.2 Invoking Entity (SCF) 181
3.3.2.3 Responding Entity (SSF) 181
3.3.3 AddEntry procedure 181
3.3.3.1 General description 181
3.3.3.2 Invoking Entity (SCF) 182
3.3.3.3 Responding Entity (SDF) 182
3.3.4 AnalysedInformation Procedure 183
3.3.4.1 General description 183
3.3.4.2 Invoking Entity (SSF) 184
3.3.4.3 Responding Entity (SCF) 185
3.3.5 AnalyseInformation procedure 186
3.3.5.1 General description 186
3.3.5.2 Invoking Entity (SCF) 186
3.3.5.3 Responding Entity (SSF) 187

Page

3.3.6 ApplyCharging procedure 189
3.3.6.1 General description 189
3.3.6.2 Invoking Entity (SCF) 189
3.3.6.3 Responding Entity (SSF) 190
3.3.7 ApplyChargingReport procedure 190
3.3.7.1 General description 190
3.3.7.2 Invoking Entity (SSF) 190
3.3.7.3 Responding Entity (SCF) 191
3.3.8 AssistRequestInstructions procedure 191
3.3.8.1 General description 191
3.3.8.2 Invoking Entity (SSF-SRF) 192
3.3.8.3 Responding Entity (SCF) 192
3.3.9 Bind procedure 192
3.3.9.1 General description 192
3.3.9.2 Invoking Entity (SCF) 193
3.3.9.3 Responding Entity (SDF) 193
3.3.10 CallGap procedure 193
3.3.10.1 General description 193
3.3.10.2 Invoking Entity (SCF) 196
3.3.10.3 Responding Entity (SSF) 196
3.3.11 CallInformationReport procedure 197
3.3.11.1 General description 197
3.3.11.2 Invoking Entity (SSF) 197
3.3.11.3 Responding Entity (SCF) 198
3.3.12 CallInformationRequest procedure 199
3.3.12.1 General description 199
3.3.12.2 Invoking Entity (SCF) 199
3.3.12.3 Responding Entity (SSF) 200
3.3.13 Cancel procedure 200
3.3.13.1 General description 200
3.3.13.2 Invoking Entity (SCF) 200
3.3.13.3 Responding Entity (SRF) 201
3.3.13.4 Responding Entity (SSF) 201
3.3.14 CollectedInformation procedure 201
3.3.14.1 General description 201
3.3.14.2 Invoking Entity (SSF) 203
3.3.14.3 Responding Entity (SCF) 205
3.3.15 CollectInformation procedure 206
3.3.15.1 General description 206
3.3.15.2 Invoking Entity (SCF) 206
3.3.15.3 Responding Entity (SSF) 207
3.3.16 Connect procedure 207
3.3.16.1 General description 207
3.3.16.2 Invoking Entity (SCF) 208
3.3.16.3 Responding Entity (SSF) 209
3.3.17 ConnectToResource procedure 210
3.3.17.1 General description 210
3.3.17.2 Invoking Entity (SCF) 210
3.3.17.3 Responding Entity (SSF) 211
3.3.18 Continue procedure 211
3.3.18.1 General description 211
3.3.18.2 Invoking Entity (SCF) 211
3.3.18.3 Responding Entity (SSF) 211
3.3.19 DisconnectForwardConnection procedure 212
3.3.19.1 General description 212
3.3.19.2 Invoking Entity (SCF) 212
3.3.19.3 Responding Entity (SSF) 213

Page

3.3.20 EstablishTemporaryConnection procedure 213
3.3.20.1 General description 213
3.3.20.2 Invoking Entity (SCF) 214
3.3.20.3 Responding Entity (SSF) 214
3.3.21 EventNotificationCharging procedure 215
3.3.21.1 General description 215
3.3.21.2 Invoking Entity (SSF) 215
3.3.21.3 Responding Entity (SCF) 216
3.3.22 EventReportBCSM procedure 216
3.3.22.1 General description 216
3.3.22.2 Invoking Entity (SSF) 217
3.3.22.3 Responding Entity (SCF) 218
3.3.23 FurnishChargingInformation procedure 218
3.3.23.1 General description 218
3.3.23.2 Invoking Entity (SCF) 219
3.3.23.3 Responding Entity (SSF) 219
3.3.24 HoldCallInNetwork procedure 220
3.3.24.1 General description 220
3.3.24.2 Invoking Entity (SCF) 220
3.3.24.3 Responding Entity (SSF) 221
3.3.25 InitialDP procedure 221
3.3.25.1 General description 221
3.3.25.2 Invoking Entity (SSF) 223
3.3.25.3 Responding Entity (SCF) 223
3.3.26 InitiateCallAttempt procedure 224
3.3.26.1 General description 224
3.3.26.2 Invoking Entity (SCF) 224
3.3.26.3 Responding Entity (SSF) 225
3.3.27 ModifyEntry procedure 225
3.3.27.1 General description 225
3.3.27.2 Invoking Entity (SCF) 226
3.3.27.3 Responding Entity (SDF) 226
3.3.28 OAnswer procedure 227
3.3.28.1 General description 227
3.3.28.2 Invoking Entity (SSF) 227
3.3.28.3 Responding Entity (SCF) 228
3.3.29 ODisconnect procedure 229
3.3.29.1 General description 229
3.3.29.2 Invoking Entity (SSF) 230
3.3.29.3 Responding Entity (SCF) 231
3.3.30 ONoAnswer procedure 231
3.3.30.1 General description 231
3.3.30.2 Invoking Entity (SSF) 232
3.3.30.3 Responding Entity (SCF) 233
3.3.31 OriginationAttemptAuthorized procedure 233
3.3.31.1 General description 233
3.3.31.2 Invoking Entity (SSF) 235
3.3.31.3 Responding Entity (SCF) 235
3.3.32 PlayAnnouncement procedure 236
3.3.32.1 General description 236
3.3.32.2 Invoking Entity (SCF) 237
3.3.32.3 Responding Entity (SRF) 237
3.3.33 PromptAndCollectUserInformation procedure 238
3.3.33.1 General description 238
3.3.33.2 Invoking Entity (SCF) 241
3.3.33.3 Responding Entity (SRF) 241

Page

3.3.34 ReleaseCall procedure 242
3.3.34.1 General description 242
3.3.34.2 Invoking Entity (SCF) 242
3.3.34.3 Responding Entity (SSF) 243
3.3.35 RemoveEntry procedure 243
3.3.35.1 General description 243
3.3.35.2 Invoking Entity (SCF) 243
3.3.35.3 Responding Entity (SDF) 244
3.3.36 RequestCurrentStatusReport procedure 244
3.3.36.1 General description 244
3.3.36.2 Invoking Entity (SCF) 244
3.3.36.3 Responding Entity (SSF) 245
3.3.37 RequestFirstStatusMatchReport procedure 245
3.3.37.1 General description 245
3.3.37.2 Invoking Entity (SCF) 246
3.3.37.3 Responding Entity (SSF) 246
3.3.38 RequestNotificationChargingEvent procedure 247
3.3.38.1 General description 247
3.3.38.2 Invoking Entity (SCF) 247
3.3.38.3 Responding Entity (SSF) 247
3.3.39 RequestReportBCSMEvent procedure 248
3.3.39.1 General description 248
3.3.39.2 Invoking Entity (SCF) 249
3.3.39.3 Responding Entity (SSF) 250
3.3.40 ResetTimer procedure 250
3.3.40.1 General description 250
3.3.40.2 Invoking Entity (SCF) 250
3.3.40.3 Responding Entity (SSF) 250
3.3.41 RouteSelectFailure Procedure 251
3.3.41.1 General description 251
3.3.41.2 Invoking Entity (SSF) 253
3.3.41.3 Responding Entity (SCF) 253
3.3.42 Search procedure 254
3.3.42.1 General description 254
3.3.42.2 Invoking Entity (SCF) 254
3.3.42.3 Responding Entity (SDF) 254
3.3.43 SelectFacility procedure 255
3.3.43.1 General description 255
3.3.43.2 Invoking Entity (SCF) 255
3.3.43.3 Responding Entity (SSF) 256
3.3.44 SelectRoute procedure 256
3.3.44.1 General description 256
3.3.44.2 Invoking Entity (SCF) 257
3.3.44.3 Responding Entity (SSF) 257
3.3.45 SendChargingInformation procedure 259
3.3.45.1 General description 259
3.3.45.2 Invoking Entity (SCF) 259
3.3.45.3 Responding Entity (SSF) 259
3.3.46 ServiceFilteringResponse procedure 260
3.3.46.1 General description 260
3.3.46.2 Invoking Entity (SSF) 261
3.3.46.3 Responding Entity (SCF) 261
3.3.47 SpecializedResourceReport procedure 262
3.3.47.1 General description 262
3.3.47.2 Invoking Entity (SRF) 262
3.3.47.3 Responding Entity (SCF) 262

Page

3.3.48 StatusReport procedure 263
3.3.48.1 General description 263
3.3.48.2 Invoking Entity (SSF) 263
3.3.48.3 Responding Entity (SCF) 263
3.3.49 TAnswer procedure 264
3.3.49.1 General description 264
3.3.49.2 Invoking Entity (SSF) 264
3.3.49.3 Responding Entity (SCF) 265
3.3.50 TBusy procedure 265
3.3.50.1 General description 265
3.3.50.2 Invoking Entity (SSF) 267
3.3.50.3 Responding Entity (SCF) 267
3.3.51 TDisconnect procedure 268
3.3.51.1 General description 268
3.3.51.2 Invoking Entity (SSF) 269
3.3.51.3 Responding Entity (SCF) 270
3.3.52 TermAttemptAuthorized procedure 270
3.3.52.1 General description 270
3.3.52.2 Invoking Entity (SSF) 271
3.3.52.3 Responding Entity (SCF) 271
3.3.53 TNoAnswer procedure 272
3.3.53.1 General description 272
3.3.53.2 Invoking Entity (SSF) 272
3.3.53.3 Responding Entity (SCF) 273
3.3.54 Unbind procedure 273
3.3.54.1 General description 273
3.3.54.2 Invoking Entity (SCF) 274
3.3.54.3 Responding Entity (SDF) 274
3.3.55 RequestEveryStatusChangeReport procedure 274
3.3.55.1 General description 274
3.3.55.2 Invoking Entity (SCF) 274
3.3.55.3 Responding Entity (SSF) 275
3.4 Services assumed from TCAP 275
3.4.1 Normal procedures 276
3.4.1.1 SSF-to-SCF messages 276
3.4.1.2 SCF-to-SSF messages 278
3.4.1.3 SCF-to/from-SRF messages 280
3.4.2 Abnormal procedures 280
3.4.2.1 SCF-to-SSF/SRF messages 281
3.4.2.2 SSF/SRF-to-SCF messages 282
3.4.3 Dialogue establishment 282
3.4.3.1 Sending of a TC-BEGIN request primitive 282
3.4.3.2 Receipt of a TC-BEGIN indication 283
3.4.3.3 Receipt of the first TC-CONTINUE ind 283
3.4.3.4 Receipt of a TC-END ind 283
3.4.3.5 Receipt of a TC-U-ABORT ind 283
3.4.3.6 Receipt of a TC-P-ABORT ind 283
3.4.4 Dialogue continuation 283
3.4.4.1 Sending entity 283
3.4.4.2 Receiving entity 284
3.4.5 Dialogue termination 284
3.4.5.1 Sending of TC-END request 284
3.4.5.2 Receipt of a TC-END indication 284
3.4.6 User Abort 284
3.4.6.1 Sending of TC-U-ABORT request 284
3.4.6.2 Receipt of a TC-U-ABORT indication 284

Page

3.4.7 Provider Abort 284
3.4.7.1 Receipt of a TC-P-ABORT indication 284
3.4.8 Procedures for INAP operations 285
3.4.8.1 Operation invocation 285
3.4.8.2 Operation invocation receipt 285
3.4.8.3 Operation Response 285
3.4.8.4 Receipt of a response 285
3.4.8.5 Other events 286
3.4.9 Mapping on to TC services 287
3.4.9.1 Dialogue control 287
3.4.9.2 Operation procedures 288

Annex A – INAP SDL Diagrams 289
A.1 Introduction 289
A.2 SDL diagrams 289

Annex B – Description of the SCSM (SDF-related states) and of the SDSM 342
B.1 Description of the Process SCSM 342
B.1.1 State 1 – "Idle" 342
B.1.1.1 Normal procedures 342
B.1.1.2 Exceptional procedures 342
B.1.2 State 2 – "Wait for subsequent requests" 342
B.1.2.1 Normal procedures 342
B.1.2.2 Exceptional procedures 343
B.1.3 State 3 – "Wait_for_Bind_result" 344
B.1.3.1 Normal procedures 344
B.1.3.2 Exceptional procedures 344
B.1.4 State 4 – "SDF Bound" 344
B.1.4.1 Normal procedures 344
B.1.4.2 Exceptional procedures 345
B.2 Description of the Process SDSM 345
B.2.1 State 1 – "Idle" 345
B.2.1.1 Normal procedures 345
B.2.1.2 Exceptional procedures 345
B.2.2 State 2 – "Bind Pending" 345
B.2.2.1 Normal procedures 345
B.2.2.2 Exceptional procedures 348
B.2.3 State 3 – "SCF Bound" 349
B.2.3.1 Normal procedures 349
B.2.3.2 Exceptional procedures 349

Appendix I – Service data modelling 352
I.1 Review of need for Service data modelling 352
I.1.1 General modelling issues 353
I.1.2 Service modelling – Defining the information required 353
I.2 Guidelines for building Service Data Models 354
I.2.1 Selection of ATTRIBUTEs 354
I.2.1.1 ATTRIBUTE Type definitions 355
I.2.1.2 Defining Permitted Values 355
I.2.1.3 Defining Default Values 356
I.2.1.4 Defining MAX Attribute Values 356
I.2.1.5 Defining Attribute Lifetimes 356
I.2.1.6 Definition of MATCHING-RULES 356
I.2.1.7 Assigning the ATTRIBUTE Object Identifiers 356
I.2.2 Selection of OBJECT-CLASSs 357
I.2.2.1 Defining the OBJECT-CLASS 357
I.2.2.2 Selecting the RDN (NAME-FORM) 357
I.2.2.3 Assigning the OBJECT-CLASS Object Identifiers 357
I.2.3 Defining the DIT 357
I.2.3.1 Defining the Object Hierarchy (STRUCTURE-RULES) 357
I.2.3.2 Defining alternative naming paths (Aliases) 359
I.2.3.3 Defining MAX number of subordinate Entries 359

I.3 Example Service (UPT Like Service) ... 359
I.3.1 Example Service description ... 359
I.3.1.1 Service Feature – Incall ... 359
I.3.1.2 Service Feature – Incall Registration ... 360
I.3.1.3 Service Feature – Incall Screening ... 361
I.3.1.4 Service Feature – Outcall ... 361
I.3.1.5 Service Feature – Outcall screening by location ... 361
I.3.1.6 General Service Information ... 362
I.3.2 Example Service data modelling ... 362
I.3.2.1 Selection of Attributes ... 362
I.3.2.2 Selection of Object Classes ... 367
I.3.2.3 Designing the Service DIT ... 368
I.3.2.4 Assigning Access control ... 370
I.4 Defining Contexts ... 371
I.4.1 Numerical Index Attribute Value Context ... 372
Appendix II – Aspects of the intelligent network interface identified as "For Further Study" (FFS) relative to CS-1 ... 372
II.1 General ... 372
II.1.1 General consideration ... 372
II.1.2 Relationship to other appendices of the Q.1200-Series Recommendations ... 372
II.1.3 Format of Document ... 372
II.2 Operations ... 372
II.2.1 Consideration applicable to all operations in this appendix ... 372
II.2.2 Add party operation ... 373
II.2.2.1 Consideration ... 373
II.2.2.2 Description ... 373
II.2.3 Attach operation ... 373
II.2.3.1 Consideration ... 373
II.2.3.2 Description ... 373
II.2.4 Change parties operation ... 373
II.2.4.1 Description ... 373
II.2.5 Detach operation ... 373
II.2.5.1 Consideration ... 373
II.2.5.2 Description ... 373
II.2.6 Hold call party connection operation ... 373
II.2.6.1 Description ... 373
II.2.7 Initiate call attempt operation (for case of more than 1 party) ... 374
II.2.7.1 Consideration ... 374
II.2.7.2 Description ... 374
II.2.8 Reconnect operation ... 374
II.2.8.1 Description ... 374
II.2.9 Release call party connection operation ... 374
II.2.9.1 Consideration ... 374
II.2.9.2 Description ... 374
II.3 Parameters ... 374
II.3.1 Considerations applicable to all parameters in this appendix ... 374
II.3.2 Leg Id created parameter (from Analyse information operation) ... 374
II.3.2.1 Description ... 374
II.3.3 Leg Id created parameter (from connect operation) ... 374
II.3.3.1 Description ... 374

Page

II.3.4 Leg Id created parameter (from initiate call attempt operation) 375
II.3.4.1 Description 375
II.3.5 Leg Id created parameter (from select facility operation) 375
II.3.5.1 Description 375
II.3.6 Leg Id created parameter (from select route operation) 375
II.3.6.1 Description 375
II.3.7 Leg 1 parameter (from initial DP operation) 375
II.3.7.1 Description 375
II.3.8 Leg 2 Parameter (from initial DP operation) 375
II.3.8.1 Description 375
II.3.9 Call Id Parameter 375
II.3.9.1 Description 375
II.4 ASN.1 module of operations and parameters 375
II.4.1 Abstract syntax of the IN CS-1 application protocol – Appendix 375
II.5 Procedures 380

Appendix III – Expanded ASN.1 coding 380

SUMMARY

This Recommendation defines the intelligent network application protocol for the support of the capabilities required by the CS-1 target services over the CS-1 interfaces (SSF-SCF, SCF-SDF and SCF-SRF) as defined in Recommendation Q.1211. It defines some of the possible protocol stack scenarios, the operations which flow between the entities and the procedures to be followed at each functional entity.

Revisions to this Recommendation provide: details on operations procedures, details on services assumed from underlying protocol layers, extensions to ensure applicability in multiple administration networks, a new SCF-SDF interface for data administration and acquisition based upon the X.500-Series of Recommendations, and detailed error handling procedures. The intent of these revisions is to facilitate consistent multi-vendor implementations in multiple networks.

Associated standardization work is contained in all the Q.1200-Series Recommendations (Intelligent network).

Recommendation Q.1218

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CS-1

(Helsinki, 1993; modified 1995)

0 Introduction

This Recommendation defines the Intelligent Network Application Protocol, (INAP) required for support of Intelligent Network Capability Set 1. It supports interactions between the following four functional entities (FE's), as defined in the IN functional model:

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

The scope of this Recommendation is the further development of the INAP for both the Integrated Services Digital Network (ISDN) and Public Switched Telecommunications Network (PSTN).

It is intended as a guide to implementors and network operators to ensure interworking between different manufacturers equipment for all the IN CS-1 defined interfaces (SCF-SSF, SCF-SRF and SCF-SDF) and between network operators for the internetwork interface (SCF-SDF).

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

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

0.1 Normative references

The following Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of the editions indicated were valid. All Recommendations and other references are subject to revision: all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

- ITU-T Recommendation X.500 (1993) | ISO/IEC 9594-1:1995, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- ITU-T Recommendation X.501 (1993) | ISO/IEC 9594-2:1995, *Information technology – Open Systems Interconnection – The Directory: Models*.
- ITU-T Recommendation X.509 (1993) | ISO/IEC 9594-8:1995, *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.
- ITU-T Recommendation X.511 (1993) | ISO/IEC 9594-3:1995, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition*.
- ITU-T Recommendation X.518 (1993) | ISO/IEC 9594-4:1995, *Information technology – Open Systems Interconnection – The Directory*: *Procedures for distributed operation*.
- ITU-T Recommendation X.519 (1993) | ISO/IEC 9594-5:1995, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications*.
- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

– ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.

– ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.

– ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1):* Parameterization of ASN.1 specifications.

– ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.

– ITU-T Recommendation X.880 (1994) | ISO/IEC 13712-1:1995, *Information technology – Remote Operations: Concepts, model and notation*.

– CCITT Recommendation Q.29 (1988), *Causes of noise and ways of reducing noise in the telephone exchange*.

– ITU-T Recommendation Q.711 (1993), *Signalling System No. 7 – Functional description of the Signalling connection control part: Service definition*.

– ITU-T Recommendation Q.762 (1993) – *General function of messages and signals of the ISDN User Part of Signalling System No. 7*.

– ITU-T Recommendation Q.763 (1993), *Formats and codes of the ISDN User Part of Signalling System No. 7*.

– ITU-T Recommendation Q.771 (1993), *Signalling System No. 7 – Functional description of transaction capabilities*.

– ITU-T Recommendation Q.772 (1993), *Signalling System No. 7 – Transaction capabilities information elements definitions*.

– ITU-T Recommendation Q.773 (1993), *Signalling System No. 7 – SCCP formats and codes*.

– ITU-T Recommendation Q.774 (1993), *Signalling System No. 7 – Transaction capabilities procedures*.

– ITU-T Recommendation Q.775 (1993), *Signalling System No. 7 – Guidelines for using transaction capabilities*.

– ITU-T Recommendation Q.931 (1993), *Digital Subscriber Signalling System No. 1 (DSS 1) – ISDN user-network interface layer 3 specification for basic call control*.

– ITU-T Recommendation Q.932 (1993), *Digital Subscriber Signalling System No. 1 (DSS 1) – Generic procedures for the control of ISDN supplementary services*.

– ITU-T Recommendation Q.1290 (1995), *Glossary of terms used in the definition of intelligent networks*.

0.2 Definition methodology

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

– the definition of the SACF/MACF rules for the protocol (clause 1);

– the definition of the operations transferred between entities (clause 2);

– the definition of the actions taken at each entity (clause 3).

The SACF/MACF rules are defined in prose. The operation definitions are in Abstract Syntax Notation 1 (ASN.1, see Recommendations X.208 and X.680), and the actions are defined in terms of state transition diagrams. Further guidance on the actions to be performed on receipt of an operation can be gained from clause 2 and from the relevant information flow in clause 6/Q.1214 (see 0.6 for the relationship between the information flows and the operations).

The INAP is a ROSE user protocol (see Recommendations X.219 and 229). The ROSE protocol is contained within the component sublayer of TCAP (see Recommendations Q.771 to 775) and DSS 1 (see Recommendation Q.932). At

present the ROSE APDU's (Application protocol data units) are conveyed in transaction sublayer messages in SS No. 7 and in the Q.931 REGISTER, FACILITY and call control messages in DSS 1. Other supporting protocols may be added at a later date.

The INAP (as a ROSE user) and the ROSE protocol have been specified using ASN.1. At present, the only standardized way to encode the resulting PDU's is the Basic Encoding Rules (see Recommendation X.209).

0.3 Example physical scenarios

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

The figures depicted in this subclause show how INAP would be supported in an SS No. 7 network environment. This does not imply that only SS No. 7 may be used as the network protocol to support INAP.

For each physical scenario, the typical SRF control procedures to be applied are shown in 3.1.3.5.

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

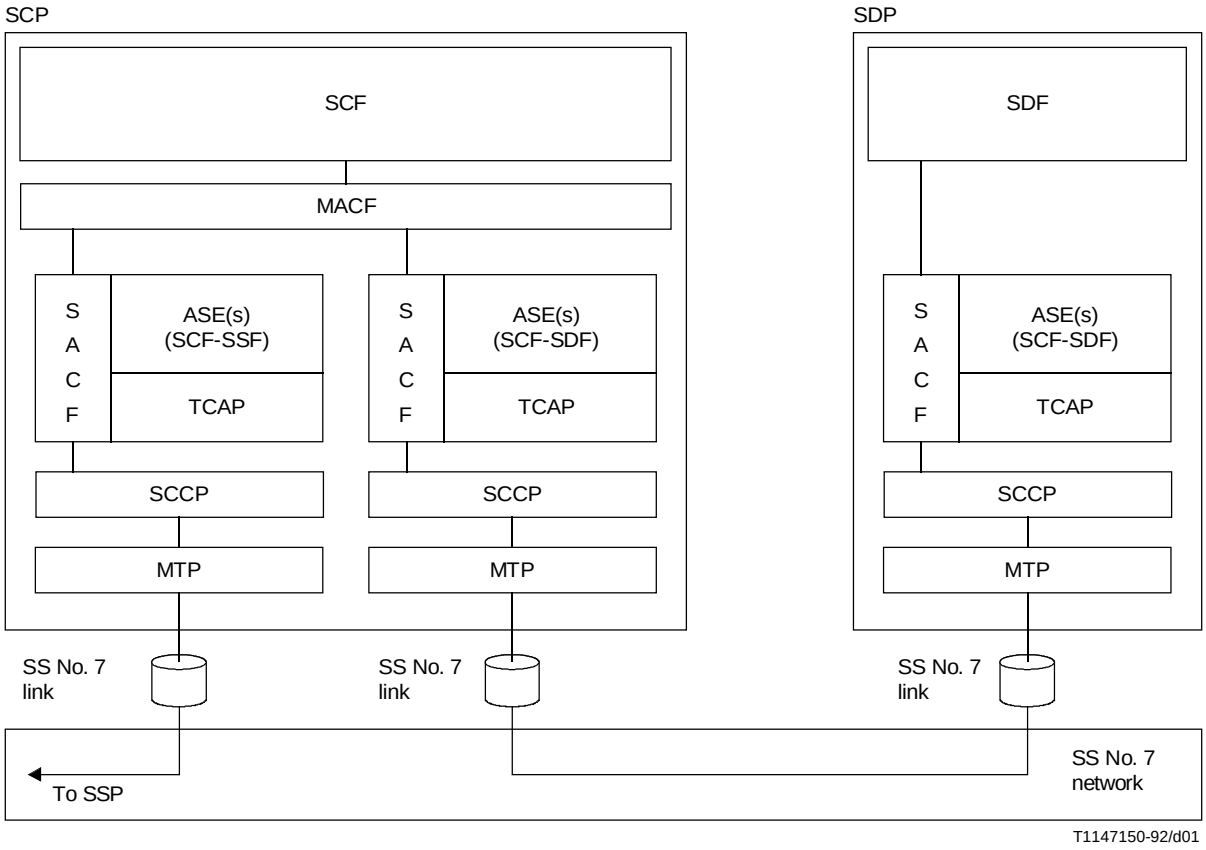


FIGURE 1/Q.1218

Physical interface between SCP and SDP

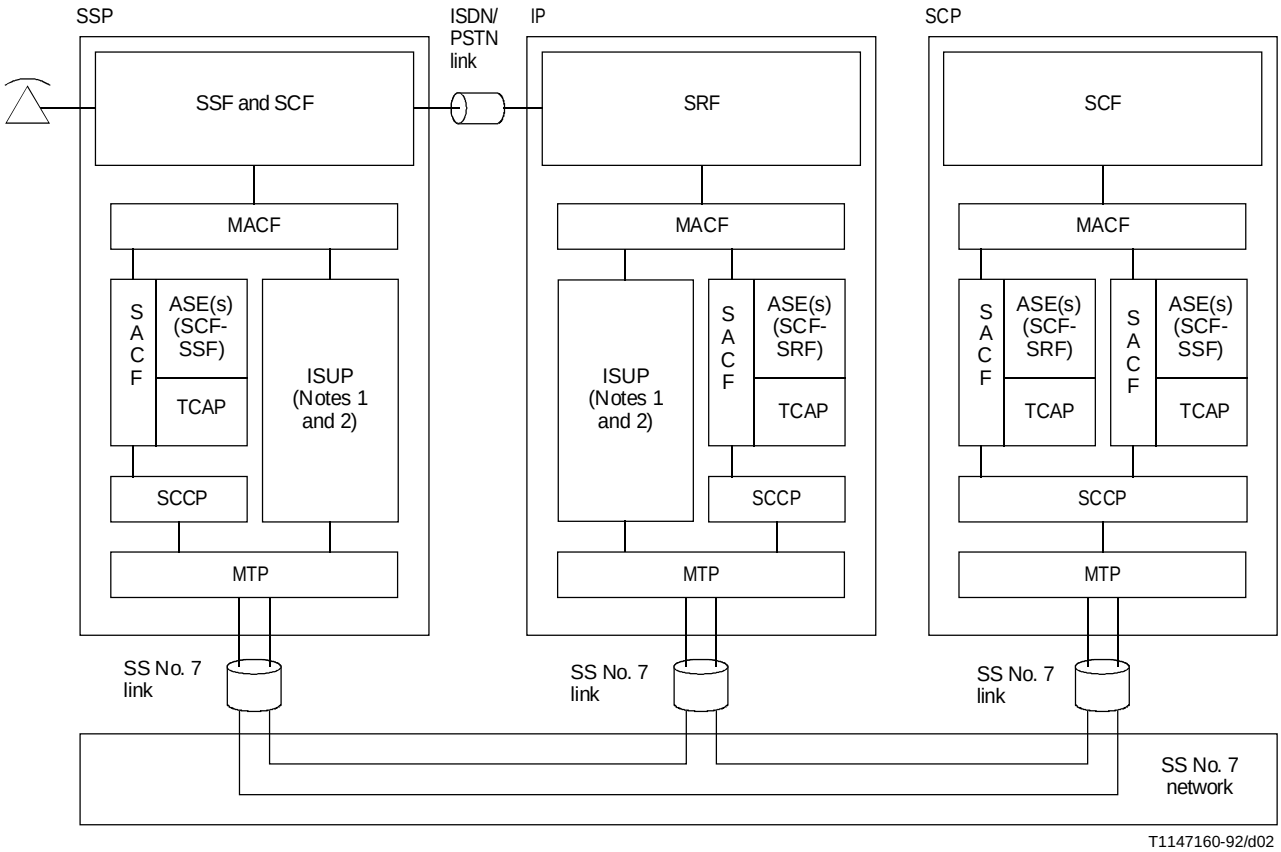
When TCAP appears in one of the following figures, it shall be understood as representing the TCAP functionalities associated with a single dialogue and transaction (as opposed to a TCAP entity).

If segmentation and re-assembly of INAP messages is required on the SCF-to-SDF interface (and on other interfaces, if needed) due to the length of messages, the segmentation and re-assembly procedure for SCCP connectionless messages, as specified in Q.714, should be used.

A number of example scenarios have been identified for support of the SCF, SSF and SRF functional entities as physical entities. These are illustrated as Figures 2 to 6. Each example is characterized by:

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

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



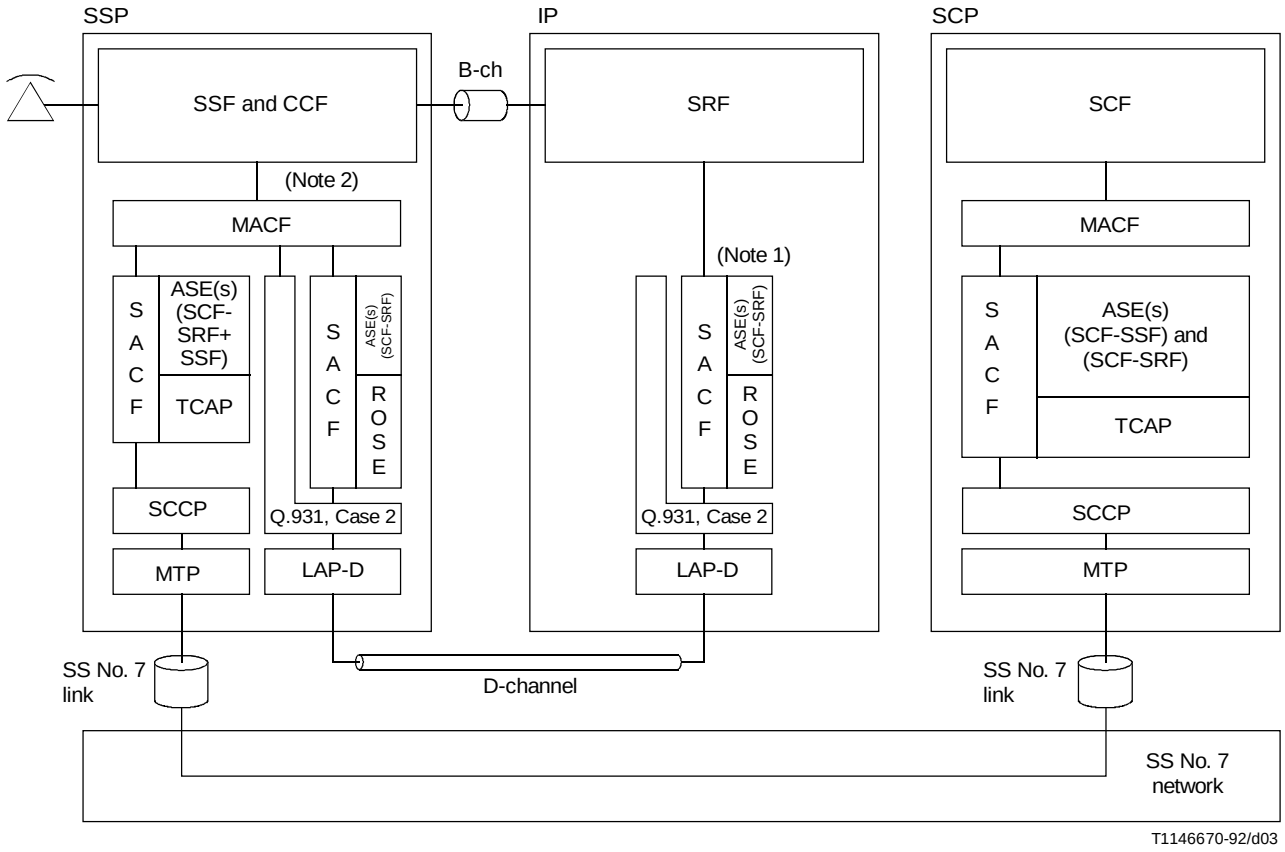
NOTES

1 Transfer of correlation information needs to be supported. This may be supported in ISUP without introducing a new ISUP parameter.

2 Other signalling systems may be used.

FIGURE 2/Q.1218

Example architecture for supporting SRF, Case 1 (SRF in IP connected to SSP and accessed by SCP through direct SS No. 7 connection)



NOTES

1 Info flows between SCF and SRF are supported by this (ROSE) entity.

2 Relay function is provided either by MACF or by application process at SSP.

FIGURE 3/Q.1218

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

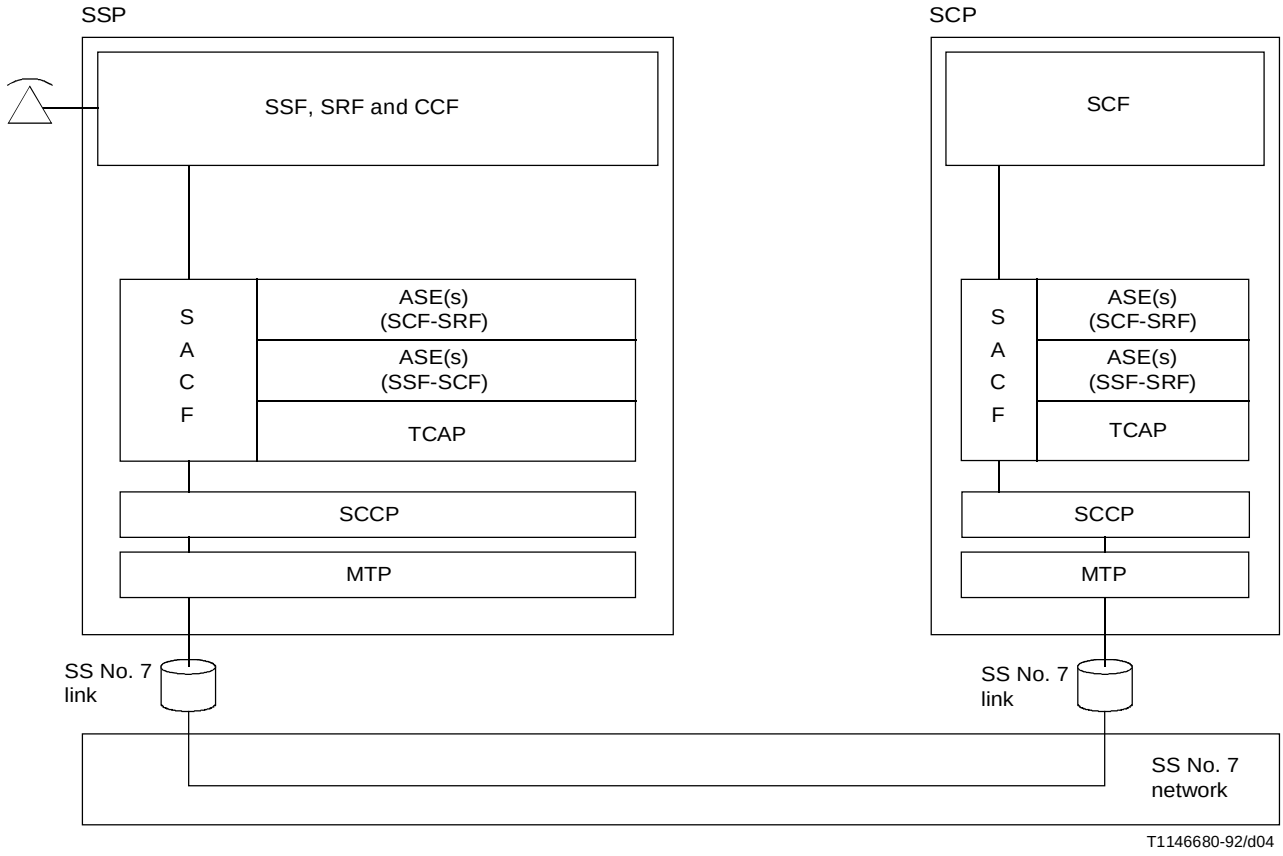
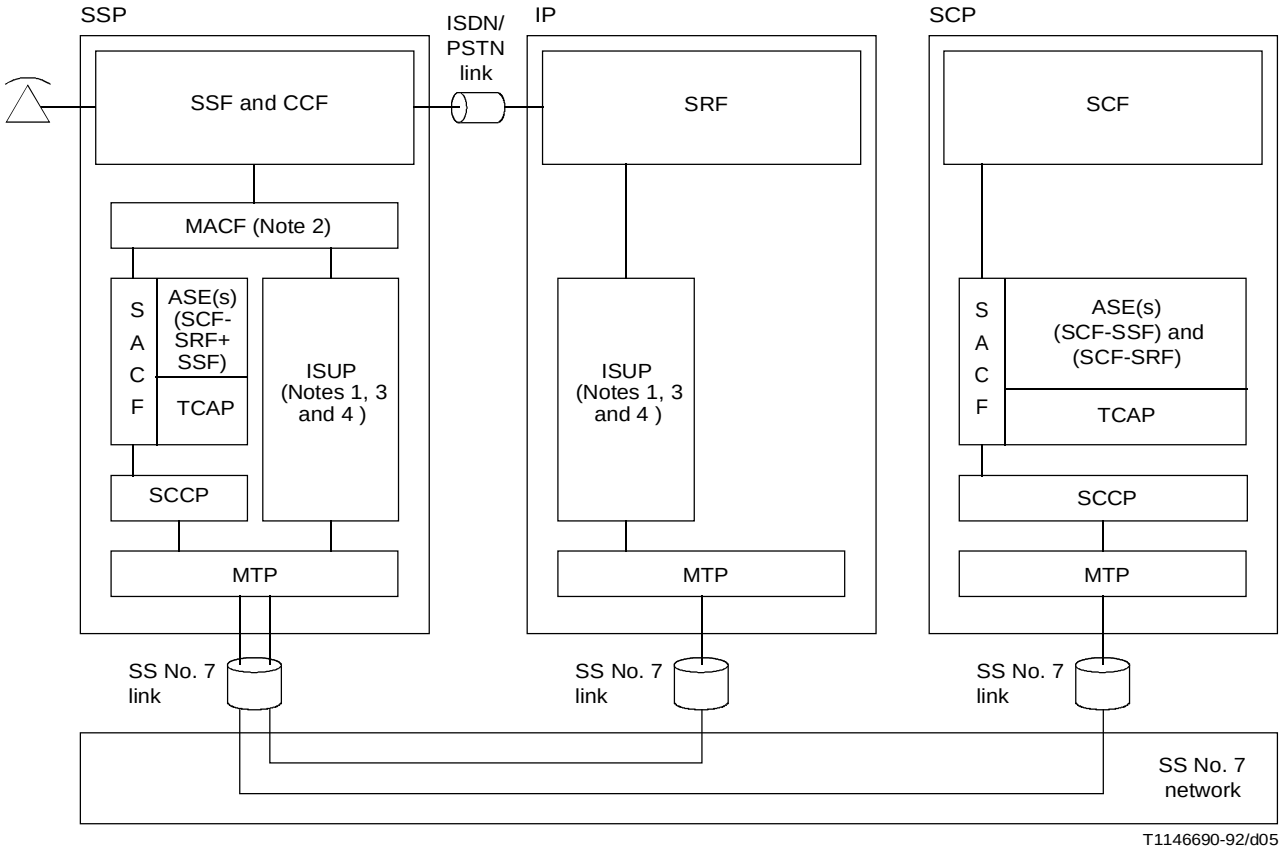


FIGURE 4/Q.1218

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



NOTES

1 Info flows between SCF and SRF as well as connection control are directly supported by ISUP.

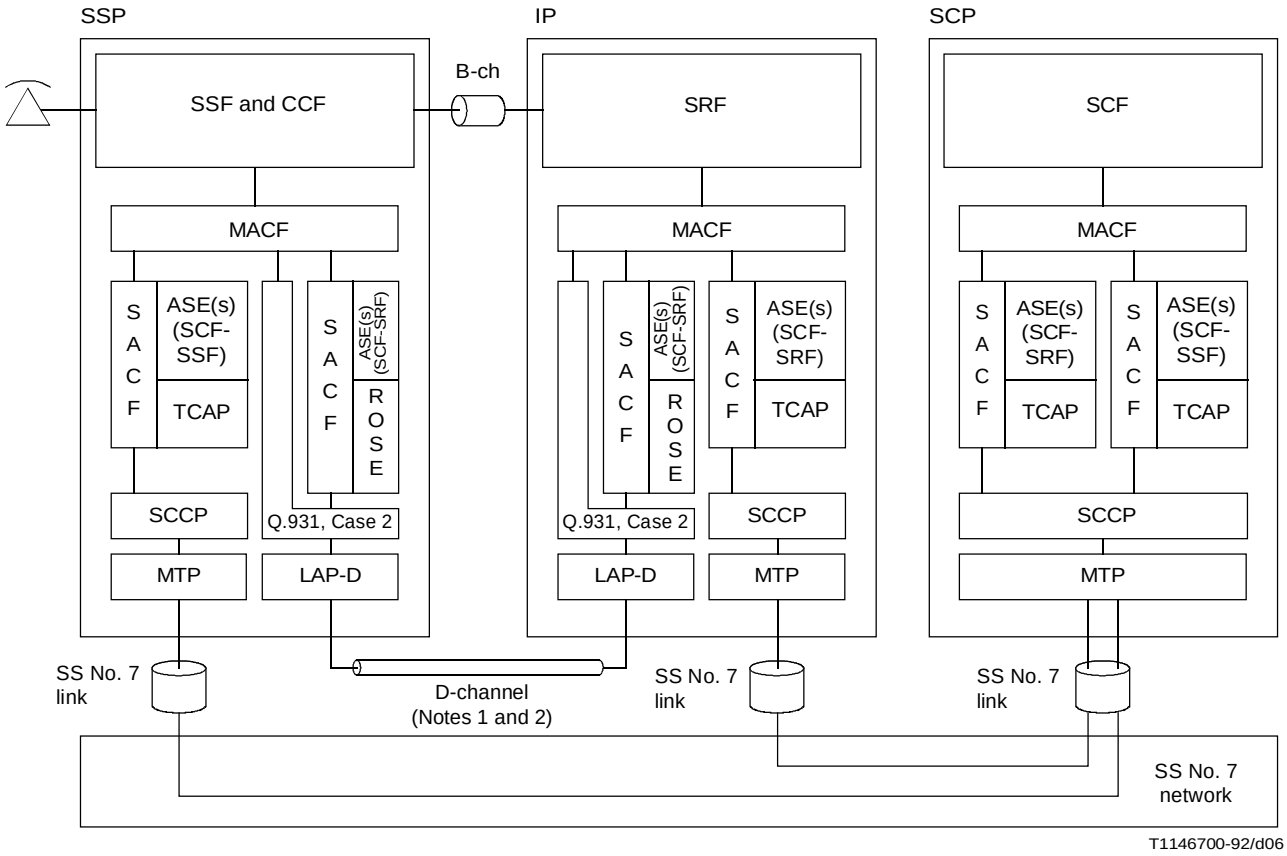
2 Relay function is provided either by MACF or by application process at SSP.

3 Assumes that ISUP provides a means to transport ROSE information.

4 Other signalling systems may be used.

FIGURE 5/Q.1218

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



NOTES

1 Transfer of correlation information needs to be supported.

2 Other signalling systems may be used.

FIGURE 6/Q.1218

Example architecture for supporting SRF, Case 5 (SRF in IP connected to SCP and SSP and accessed via both SS No. 7 and D-channel respectively)

Table 1 summarizes the selection of features for each figure.

TABLE 1/Q.1218

Type of signalling system between SSF and SRF	Method to support SCF-SRF relationship	
	Direct TCAP link	Relay via SSP
ISUP	Figure 2[a)]	Figure 5[d)]
DSS 1	Figure 6[e)]	Figure 3[b)]
Implementation dependent	As Figures 2 or 6 but with implementation dependent SCP-IP interface	Figure 4[c)]

Additional information related to each figure:

a) Figure 2: All associations are supported by SS No. 7, either TCAP or ISUP. In this case the IP is one of the network nodes.

b) Figure 3: IP can be accessed by DSS 1 only. The IP can be a physical entity residing outside the network.

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

d) Figure 5: IP can be accessed by ISUP only. The handling of SRF by SCF could be the same as that of Figure 3.

e) Figure 6: The handling of SRF by SCF could be the same as that of Figure 2. Other types of signalling systems could be used.

0.4 INAP protocol architecture

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

The INAP protocol architecture can be illustrated as shown in Figure 7.

A physical entity has either single interactions (Case a) or multiple coordinated interactions (Case b) with other physical entities.

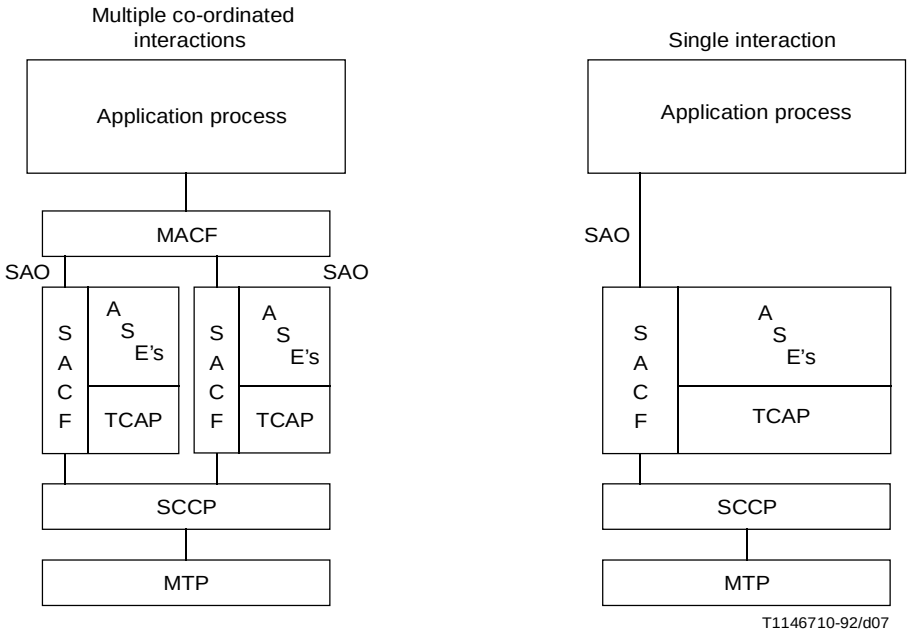
In case a, SACF provides a coordination function in using ASE's, which includes the ordering of operations supported by ASE(s), (based on the order of received primitives). The SAO represents the SACF plus a set of ASE's to be used over a single interaction between a pair of PE's.

In Case b, MACF provides a coordinating function among several SAO's, each of which interacts with an SAO in a remote PE.

Each ASE supports one or more operations. Description of each operation is tied with the action of corresponding FE modelling (see Recommendation Q.1214 and clause 3 of this Recommendation). Each operation is specified using the OPERATION macro described in Figure 8 except for the SCF-to-SDF interface, which is described using the CLASS notation.

The use of the application context negotiation mechanism [as defined in the Q.770-Series (Transaction capabilities application part)] allows the two communicating entities to identify exactly what their capabilities are and also what the capabilities required on the interface should be. This should be used to allow evolution through Intelligent Network capability sets.

If the indication of a specific application context is not supported by a pair of communicating FE's, some mechanism to pre-arrange the context must be supported.



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

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

FIGURE 7/Q.1218

INAP protocol architecture

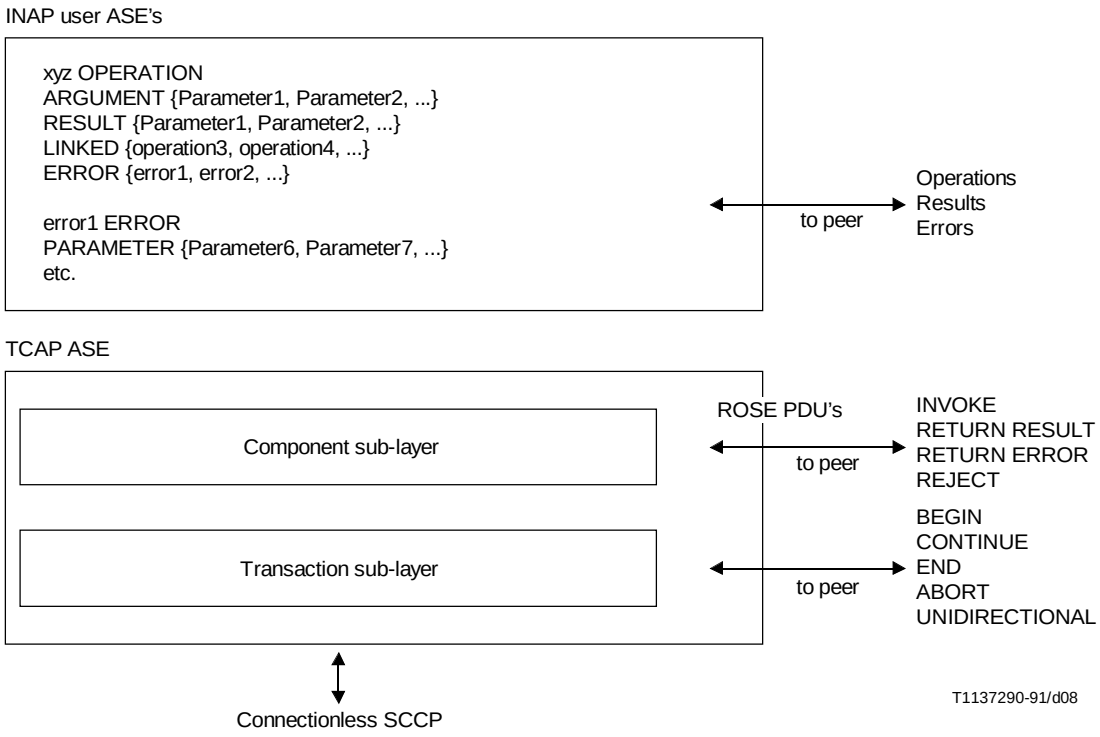


FIGURE 8/Q.1218

Operation description

0. 4.1 INAP signalling congestion control for Signalling System No. 7

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 (see D.2.11/Q.767), 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, AnalyzedInformation, or InitiateCallAttempts) 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.

0.5 INAP addressing

SCCP global title and MTP point code addressing [see Q.710-Series (Signalling connection control part) and Q.700-Series (Message transfer part)] ensure that PDU's reach their physical destination (i.e. the correct point code) regardless of which network it is in.

Within a node, it is the choice of the network operator/implementor as to which SSN or SSNs are assigned to INAP.

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

0.6 Relationship between Recommendation Q.1214 and this Recommendation

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

Rec. Q.1214 Reference	*Information Flow*	*Operation*
6.4.2.1	Activate Service Filtering	Same
6.4.2.2	Activity Test	Same
6.4.2.3	Activity Test Response	Return Result from ActivityTest
6.4.2.4	Analysed Information	Same, InitialDP, EventReportBCSM
6.4.2.5	Analyse Information	Same, Connect
6.4.2.6	Apply Charging	Same
6.4.2.7	Apply Charging Report	Same
6.4.2.8	Assist Request Instructions	Same
6.4.2.9	Call Gap	Same
6.4.2.10	Call Information Report	Same
6.4.2.11	Call Information Request	Same
6.4.2.13	Cancel Status Report Request	Same
6.4.2.14	Collected Information	Same, InitialDP, EventReportBCSM
6.4.2.15	Collect Information	Same
6.4.2.16	Connect	Same
6.4.2.17	Connect to Resource	Same

Rec. Q.1214 Reference	*Information Flow*	*Operation*
6.4.2.18	Continue	Same
6.4.2.19	Disconnect Forward Connection	Same
6.4.2.20	Establish Temporary Connection	Same
6.4.2.21	Event Notification Charging	Same
6.4.2.22	Event Report BCSM	Same
6.4.2.23	Furnish Charging Information	Same
6.4.2.24	Hold call in network	Same, ResetTimer and FurnishChargingInformation
6.4.2.25	Initial DP	Same
6.4.2.26	Initiate Call Attempt	Same
6.4.2.27	OAnswer	Same, InitialDP, EventReportBCSM
6.4.2.28	OCalledPartyBusy	Same, InitialDP, EventReportBCSM
6.4.2.29	ODisconnect	Same, InitialDP, EventReportBCSM
6.4.2.30	OMidCall	Same, InitialDP, EventReportBCSM
6.4.2.31	ONo_Answer	Same, InitialDP, EventReportBCSM
6.4.2.32	Origination Attempt Authorized	Same, InitialDP
6.4.2.33	Release Call	Same
6.4.2.34	Request Notification Charging Event	Same
6.4.2.35	Request Report BCSM Event	Same
6.4.2.36	Request Status Report	RequestCurrentStatusReport RequestFirstStatusMatchReport RequestEveryStatusChangeReport
6.4.2.37	Reset Timer	Same
6.4.2.38	Route Select Failure	Same, InitialDP, EventReportBCSM
6.4.2.39	Select Facility	Same
6.4.2.40	Select Route	Same, Connect
6.4.2.41	Send Charging Information	Same
6.4.2.42	Service Filtering Response	Same
6.4.2.43	Status Report	Same
6.4.2.44	TAnswer	Same, InitialDP, EventReportBCSM
6.4.2.45	TCalled Party Busy	Same, InitialDP, EventReportBCSM
6.4.2.46	TDisconnect	Same, InitialDP, EventReportBCSM
6.4.2.47	Term Attempt Authorized	Same, InitialDP

Rec. Q.1214 Reference	*Information Flow*	*Operation*
6.4.2.48	TMidCall	Same, InitialDP, EventReportBCSM
6.4.2.49	TNoAnswer	Same, InitialDP, EventReportBCSM
6.5.2.1	AssistRequestInstructions from SRF	AssistRequestInstructions
6.5.2.2	Cancel Announcement	Cancel
6.5.2.3	Collected User Information	Return Result from Prompt and collect user information
6.5.2.4	Play Announcement	Same
6.5.2.5	Prompt and collect user information	Same
6.5.2.6	Specialized Resource Report	Same
6.6.2.1	Search	Same
6.6.2.2	Search Result	Return Result from Search
6.6.2.3	ModifyEntry	Same
6.6.2.4	Modify Entry Result	Return Result from Modifyentry
6.6.2.5	Authenticate	Bind
6.6.2.6	Authenticate Result	Return Result from Bind
6.6.2.7	AddEntry	Same
6.6.2.8	Add Entry Result	Return Result from AddEntry
6.6.2.9	RemoveEntry	Same
6.6.2.10	Remove Entry Result	Return Result from RemoveEntry

0.7 Compatibility mechanisms used for INAP

0.7.1 Introduction

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

There are three categories of compatibility:

– *Minor changes to INAP in future standardized versions:*

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

– *Major changes to INAP in future standardized versions:*

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

– *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 Recommendations.

0.7.2 Definition of INAP compatibility mechanisms

0.7.2.1 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 Recommendation Q.773.

0.7.2.2 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 Recommendation X.680-Series).

0.7.2.3 Procedures for inclusion of network specific additions to INAP

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

For the SCF-to-SDF interface, the extension mechanisms are defined in 2.2.

1 SACF/MACF rules

1.1 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 Q.770-Series (Transaction capabilities application part) for a more detailed description of the TCAP AC negotiation mechanism.

1.2 Sequential/Parallel execution of operations

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

– charging operations may be synchronized with any other operation.

The method of indicating that operations are to be synchronized is to include them in the same message. Where one of the operations identified above must not be 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 clause 3, the FE-specific rules take precedence over the generic rules.

2 Abstract syntax of the IN CS-1 Application Protocol

This clause specifies the abstract syntax for the IN CS-1 Application Protocol using Abstract Syntax Notation One (ASN.1), defined in Recommendations X.208 and X.680.

The encoding rules which are applicable to the defined abstract syntax are the basic encoding rules for ASN.1, defined in Recommendation X.209 with the restrictions as described in 4.1.1/Q.773. Additional encodings are cited for parameters used in existing ISUP (Q.763) and DSS 1 (Q.931) Recommendations.

For the ISUP and DSS 1 parameters used in the INAP, only the coding of the parameter value will be coded as defined in ISUP or DSS 1. The DSS 1/ISUP defined parameter identifiers are removed and replaced by the INAP defined parameter identifiers.

The mapping of OPERATION and ERROR to TCAP components is defined in Recommendation Q.773. The class of an operation is not stated explicitly but is specified in the ASN.1 OPERATION MACRO, as follows:

Class 1 – Both RESULT and ERRORS appear in the ASN.1 OPERATION MACRO definition.

Class 2 – Only ERRORS appears in the ASN.1 OPERATION MACRO definition.

Class 3 – Only RESULT appears in the ASN.1 OPERATION MACRO definition.

Class 4 – Neither RESULT nor ERRORS appears in the ASN.1 OPERATION MACRO definition.

These map to the classes 2 through 5, respectively, specified in Recommendations X.219 and Q.932.

The abstract syntax for INAP is composed of several ASN.1 modules describing operations, errors, and associated data types. The values (operation codes and error codes) are defined in a separate module.

The module containing all the type definitions for INAP operations is **IN-CS-1-Operations** and is described in 2.1.1 and 2.2.2.4.1.

The module containing all the type definitions for INAP errors is **IN-CS-1-Errors** and is described in 2.1.2 and 2.2.2.4.1.

The module containing all the type definitions for INAP data types is **IN-CS-1-Data Types** and is described in 2.1.3 and 2.2.2.4.1.

The module containing the operation codes and error codes for INAP is **IN-CS-1-Codes** and is described in 2.1.4 and 2.2.2.4.1.

2.1 SSF-SCF, SCF-SRF Interface

2.1.1 IN CS-1 Operation Types

```
IN-CS-1-Operations { ccitt recommendation q 1218 modules(0) cs-1-operations(0) version1(0) }
-- This module contains the type definitions for the IN CS-1 operations.
-- There may be functional redundancies in the operation set related to call processing.
-- This may make product interworking more difficult. Administrations wishing to deploy
-- IN and equipment manufacturers implementing IN should take this into account.

DEFINITIONS ::=

BEGIN

IMPORTS

        OPERATION
```

FROM TCAPMessages { ccitt recommendation q 773 modules(0) messages(1) version2(2) }

-- *error types*

Cancelled,
CancelFailed,
ETCFailed,
ImproperCallerResponse,
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
RequestedInfoError,
SystemFailure,
TaskRefused,
UnavailableResource,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter,
UnknownLegID,
UnknownResource

FROM IN-CS-1-Errors { ccitt recommendation q 1218 modules(0) cs-1-errors(1) version1(0) }

-- *argument types*

ActivateServiceFilteringArg,
AnalysedInformationArg,
AnalyseInformationArg,
ApplyChargingArg,
ApplyChargingReportArg,
AssistRequestInstructionsArg,
CallGapArg,
CallInformationReportArg,
CallInformationRequestArg,
CancelArg,
CancelStatusReportRequestArg,
CollectedInformationArg,
CollectInformationArg,
ConnectArg,
ConnectToResourceArg,
EstablishTemporaryConnectionArg,
EventNotificationChargingArg,
EventReportBCSMArg,
FurnishChargingInformationArg,
HoldCallInNetworkArg,
InitialDPArg,
InitiateCallAttemptArg,
MidCallArg,
OAnswerArg,
OCalledPartyBusyArg,
ODisconnectArg,
ONoAnswerArg,
OriginationAttemptAuthorizedArg,
PlayAnnouncementArg,
PromptAndCollectUserInformationArg,
ReceivedInformationArg,
ReleaseCallArg,
RequestCurrentStatusReportArg,
RequestCurrentStatusReportResultArg,
RequestEveryStatusChangeReportArg,
RequestFirstStatusMatchReportArg,
RequestNotificationChargingEventArg,
RequestReportBCSMEventArg,
ResetTimerArg,
RouteSelectFailureArg,
SelectFacilityArg,
SelectRouteArg,
SendChargingInformationArg,
ServiceFilteringResponseArg,
SpecializedResourceReportArg,

```
        StatusReportArg,
        TAnswerArg,
        TBusyArg,
        TDisconnectArg,
        TermAttemptAuthorizedArg,
        TNoAnswerArg,

FROM IN-CS-1-DataTypes { ccitt recommendation q 1218 modules(0) cs-1-datatypes(2) version1(0) };

-- TYPE DEFINITIONS FOR  IN CS-1  OPERATIONS FOLLOWS

-- SCF-SSF operations

ActivateServiceFiltering ::= OPERATION
        ARGUMENT
            ActivateServiceFilteringArg
        RESULT
        ERRORS {
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedParameter
            }
-- Direction: SCF –> SSF, Timer: T_asf
-- When receiving this operation, the SSF handles calls to destination in a specified manner
-- without sending queries for every detected call. It is used for example for providing
-- televoting or mass calling services. Simple registration functionality (counters) and
-- announcement control may be  located at the SSF. The operation initializes the specified
-- counters in the SSF.

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

AnalysedInformation ::= OPERATION
        ARGUMENT
            AnalysedInformationArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- Direction:  SSF –> SCF, Timer: T_adi
-- This operation is used to indicate availability of routing address and call type. (DP 3 –
-- Analysed_Info).
-- For additional information on this operation and its use with open numbering plans, refer to
-- 4.2.2.2a)3)/Q.1214.

AnalyseInformation ::= OPERATION
        ARGUMENT
            AnalyseInformationArg
        ERRORS {
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
```

UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SCF –> SSF, Timer: T_{ai}*
-- *This operation is used to request the SSF to perform the originating basic call processing actions*
-- *to analyse destination information that is either collected from a calling party or provided by the SCF*
-- *(e.g. for number translation). This includes actions to validate the information according to an office*
-- *or customized dialling plan, and if valid, to determine call termination information, to include the called*
-- *party address, the type of call (e.g. intra-network or inter-network), and carrier (if inter-network).*
-- *If the called party is not served by the SSF, the SSF also determines a route index based on the called*
-- *party address and class of service, where the route index points to a list of outgoing trunk groups.*

ApplyCharging ::= OPERATION
ARGUMENT
ApplyChargingArg
ERRORS {
MissingParameter,
UnexpectedComponentSequence,
UnexpectedParameter,
UnexpectedDataValue,
ParameterOutOfRange,
SystemFailure,
TaskRefused
}

-- *Direction: SCF –> SSF, Timer: T_{ac}*
-- *This operation is used for interacting from the SCF with the SSF charging mechanisms. The*
-- *ApplyChargingReport operation provides the feedback from the SSF to the SCF.*

ApplyChargingReport ::= OPERATION
ARGUMENT
ApplyChargingReportArg
ERRORS {
MissingParameter,
UnexpectedComponentSequence,
UnexpectedParameter,
UnexpectedDataValue,
ParameterOutOfRange,
SystemFailure,
TaskRefused
}

-- *Direction: SSF –> SCF, Timer: T_{acr}*
-- *This operation is used by the SSF to report to the SCF the occurrence of a specific charging event as*
-- *requested by the SCF using the ApplyCharging operation.*

AssistRequestInstructions ::= OPERATION
ARGUMENT
AssistRequestInstructionsArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SSF –> SCF or SRF –> SCF, Timer: T_{ari}*
-- *This operation is used when there is an assist or a hand-off procedure and may be sent by the SSF*
-- *or SRF to the SCF. This operation is sent by the assisting SSF or assisting SRF to the SCF, when the*
-- *initiating SSF has set up a connection to the SRF or to the assisting SSF as a result of receiving*
-- *an EstablishTemporaryConnection or Connect operation (in the case of hand-off) from the SCF.*
-- *Refer to clause 3 for a description of the procedures associated with this operation.*

CallGap ::= OPERATION
ARGUMENT
CallGapArg
-- Direction: SCF –> SSF, Timer: T_{cg}
-- This operation is used to request the SSF to reduce the rate at which specific service requests are sent to
-- the SCF. Use of this operation by the SCF to gap queries and updates at the SDF is for further study.

CallInformationReport ::= OPERATION
ARGUMENT
CallInformationReportArg
-- Direction: SSF –> SCF, Timer: T_{cirp}
-- This operation is used to send specific call information for a single call to the SCF as requested by the
-- SCF in a previous CallInformationRequest.

CallInformationRequest ::= OPERATION
ARGUMENT
CallInformationRequestArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
RequestedInfoError,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}
-- Direction: SCF –> SSF, Timer: T_{cirq}
-- This operation is used to request the SSF to record specific information about a single call and report it to
-- the SCF (with a CallInformationReport operation).

Cancel ::= OPERATION
ARGUMENT
CancelArg
ERRORS {
CancelFailed,
MissingParameter,
TaskRefused
}
-- Direction: SCF –> SSF, or SCF –> SRF, Timer: T_{can}
-- This generic operation cancels the correlated previous operation or all previous requests. The following
-- operations can be canceled: PlayAnnouncement, PromptAndCollectUserInformation.

CancelStatusReportRequest ::= OPERATION
ARGUMENT
CancelStatusReportRequestArg
ERRORS {
CancelFailed,
MissingParameter,
TaskRefused
}
-- Direction: SCF –> SSF, Timer: T_{csr}
-- This operation cancels the following processes: RequestFirstStatusMatchReport and
-- RequestEveryStatusChangeReport.

CollectedInformation ::= OPERATION
ARGUMENT
CollectedInformationArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SSF –> SCF, Timer: T_{cdi}*
-- *This operation is used to indicate availability of complete initial information package/dialling string from*
-- *originating party. (This event may have already occurred in the case of en bloc signalling, in which case*
-- *the waiting duration in this PIC is zero.) (DP 2 – Collected_Info). For additional information on this*
-- *operation and its use with open numbering plans, refer to 4.2.2.2.1-2/Q.1214.*

CollectInformation ::= OPERATION
ARGUMENT
CollectInformationArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}
-- *Direction: SCF –> SSF, Timer: T_{ci}*
-- *This operation is used to request the SSF to perform the originating basic call processing actions to*
-- *prompt a calling party for destination information, then collect destination information according to a*
-- *specified numbering plan (e.g. for virtual private networks).*

Connect ::= OPERATION
ARGUMENT
ConnectArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}
-- *Direction: SCF –> SSF, Timer: T_{con}*
-- *This operation is used to request the SSF to perform the call processing actions to route or forward a call*
-- *to a specified destination. To do so, the SSF may or may not use destination information from the calling*
-- *party (e.g. dialled digits) and existing call set-up information (e.g. route index to a list of trunk groups),*
-- *depending on the information provided by the SCF.*
-- *When address information is only included in the Connect operation, call processing resumes at PIC3 in*
-- *the O-BCSM.*
-- *When address information and routing information is included, call processing resumes at PIC4.*

ConnectToResource ::= OPERATION
ARGUMENT
ConnectToResourceArg
ERRORS {
MissingParameter,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}
-- *Direction: SCF –> SSF, Timer: T_{ctr}*
-- *This operation is used to connect a call from the SSP to the physical entity containing the SRF.*
-- *Refer to clause 3 for a description of the procedures associated with this operation.*

Continue ::= OPERATION
-- *Direction: SCF –> SSF, Timer: T_{cue}*
-- *This operation is used to request the SSF to proceed with call processing at the DP at which it*
-- *previously suspended call processing to await SCF instructions (i.e. proceed to the next point*
-- *in call in the BCSM). The SSF continues call processing without substituting new data from SCF.*

```
DisconnectForwardConnection ::= OPERATION
        ERRORS {
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence
            }
-- Direction: SCF –> SSF, Timer: T_dfc
-- This operation is used to disconnect a forward temporary connection or a connection to a resource.
-- Refer to clause 3 for a description of the procedures associated with this operation.

EstablishTemporaryConnection ::= OPERATION
        ARGUMENT
            EstablishTemporaryConnectionArg
        ERRORS {
            ETCFailed,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- Direction: SCF –> SSF, Timer: T_etc
-- This operation is used to create a connection to a resource for a limited period of time
-- (e.g. to play an announcement, to collect user information); it implies the use of the assist
-- procedure. Refer to clause 3 for a description of the procedures associated with this operation.

EventNotificationCharging ::= OPERATION
        ARGUMENT
            EventNotificationChargingArg
-- Direction: SSF –> SCF, Timer: T_enc
-- This operation is used by the SSF to report to the SCF the occurrence of a specific charging event
-- type as previously requested by the SCF in a RequestNotificationChargingEvent operation.

EventReportBCSM ::= OPERATION
        ARGUMENT
            EventReportBCSMArg
-- Direction: SSF –> SCF, Timer: T_erb
-- This operation is used to notify the SCF of a call-related event (e.g. BCSM events such as busy or
-- no answer) previously requested by the SCF in a RequestReportBCSMEvent operation.

FurnishChargingInformation ::= OPERATION
        ARGUMENT
            FurnishChargingInformationArg
        ERRORS {
            MissingParameter,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- Direction: SCF –> SSF, Timer: T_fci
-- This operation is used to request the SSF to generate, register a call record or to include some
-- information in the default call record. The registered call record is intended for off line charging of the
-- call.

HoldCallInNetwork ::= OPERATION
        ARGUMENT
            HoldCallInNetworkArg
        ERRORS {
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- *Direction: SCF –> SSF, Timer: T_{hcn}*
-- *This operation is used to provide the capability of queueing a call during the set-up phase (e.g. to provide*
-- *a call completion to busy, the call would be queued until the destination becomes free).*

```
InitialDP ::= OPERATION
        ARGUMENT
            InitialDPArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- *Direction: SSF –> SCF, Timer: T_{idp}*
-- *This operation is used after a TDP to indicate request for service.*

```
InitiateCallAttempt ::= OPERATION
        ARGUMENT
            InitiateCallAttemptArg
        ERRORS {
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- *Direction: SCF –> SSF, Timer: T_{ica}*
-- *This operation is used to request the SSF to create a new call to one call party using address*
-- *information provided by the SCF.*

```
OAnswer ::= OPERATION
        ARGUMENT
            OAnswerArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- *Direction: SSF –> SCF, Timer: T_{oa}*
-- *This operation is used for indication from the terminating half BCSM that the call is accepted and*
-- *answered by terminating party (e.g. terminating party goes offhook, Q.931 Connect message received,*
-- *ISDN-UP Answer message received) (DP 7 – O_Answer). For additional information on this operation,*
-- *refer to 4.2.2.2.1-5/Q.1214.*

```
OCalledPartyBusy ::= OPERATION
        ARGUMENT
            OCalledPartyBusyArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- Direction: SSF –> SCF, Timer: T_{ob}
-- This operation is used for Indication from the terminating half BCSM that the terminating party is busy
-- (DP 5 – O_Called_Party_Busy). For additional information on this operation, refer to 4.2.2.2.1-4/Q.1214.

ODisconnect ::= OPERATION
ARGUMENT
ODisconnectArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- Direction: SSF –> SCF, Timer: T_{od}
-- This operation is used for a disconnect indication (e.g. onhook, Q.931 Disconnect message, SS7 Release
-- message) is received from the originating party, or received from the terminating party via the terminating
-- half BCSM. (DP 9 – O_Disconnect). For additional information on this operation, refer to
-- 4.2.2.2.1-5/Q.1214.

OMidCall ::= OPERATION
ARGUMENT
MidCallArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- Direction: SSF –> SCF, Timer: T_{omc}
-- This operation is used to indicate a feature request is received from the originating party
-- (e.g. hook flash, ISDN feature activation, Q.931 HOLD or RETrieve message). (DP 8 – O_Mid_Call).
-- For additional information on this operation, refer to 4.2.2.2.1-5/Q.1214.

ONoAnswer ::= OPERATION
ARGUMENT
ONoAnswerArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- Direction: SSF –> SCF, Timer: T_{ona}
-- This operation is used for indication from the terminating half BCSM that the terminating party does not
-- answer within a specified time period (DP 6 – O_No_Answer). For additional information on this
-- operation, refer to 4.2.2.2.1-4/Q.1214.

OriginationAttemptAuthorized ::= OPERATION
ARGUMENT
OriginationAttemptAuthorizedArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,

TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SSF –> SCF, Timer: T_{oaa}*
-- *This operation is used to indicate the desire to place outgoing call (e.g. offhook, Q.931 Set-up message,*
-- *ISDN-UP IAM message) and authority/ability to place outgoing call verified*
-- *(DP 1 – Origination_Attempt_Authorized). For additional information on this operation, refer to*
-- *4.2.2.2.1-1/Q.1214.*

ReleaseCall ::= OPERATION
ARGUMENT
ReleaseCallArg

-- *Direction: SCF –> SSF, Timer: T_{rc}*
-- *This operation is used to tear down an existing call at any phase of the call for all parties*
-- *involved in the call.*

RequestCurrentStatusReport ::= OPERATION
ARGUMENT
RequestCurrentStatusReportArg
RESULT
RequestCurrentStatusReportResultArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedParameter,
UnknownResource
}

-- *Direction: SCF –> SSF, Timer: T_{rcs}*
-- *This operation is used to request the SSF to report immediately the busy/idle status of a physical*
-- *termination resource.*

RequestEveryStatusChangeReport ::= OPERATION
ARGUMENT
RequestEveryStatusChangeReportArg
RESULT
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedParameter,
UnknownResource
}

-- *Direction: SCF –> SSF, Timer: T_{res}*
-- *This operation is used to request the SSF to report every change of busy/idle status of a physical*
-- *termination resource.*

RequestFirstStatusMatchReport ::= OPERATION
ARGUMENT
RequestFirstStatusMatchReportArg
RESULT
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedParameter,
UnknownResource
}

-- *Direction: SCF –> SSF, Timer: T_{rfs}*
-- *This operation is used to request the SSF to report the first change busy/idle to the specified status of*
-- *a physical termination resource.*

RequestNotificationChargingEvent ::= OPERATION
ARGUMENT
RequestNotificationChargingEventArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SCF –> SSF, Timer: T_{rnc}*
-- *This operation is used by the SCF to instruct the SSF on how to manage the charging events*
-- *which are received from other FE's and not under control of the service logic instance.*

RequestReportBCSMEvent ::= OPERATION
ARGUMENT
RequestReportBCSMEventArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SCF –> SSF, Timer: T_{rrb}*
-- *This operation is used to request the SSF to monitor for a call-related event (e.g. BCSM events such as*
-- *busy or no answer), then send a notification back to the SCF when the event is detected.*
-- *It is proposed that Event Detection Point (EDP) processing be always initiated by RequestReportBCSMEvent*
-- *and the EDP may be acknowledged with either an EventReportBCSM or by a DP-specific operations:*
-- *NOTE – The application context should identify whether Request Report BCSM Event ASE and DP*
-- *Generic BCSM EventReport ASE are being used, or whether Request Report BCSM EventASE,*
-- *Basic BCP DP ASE, and Advanced BCP DP ASE are being used.*
-- *– For a particular IN, only one of the two alternatives identified by the respective ASEs should be*
-- *selected (i.e. only one approach should be selected for a given application context).*
-- *– Further study is required to identify the small set of parameters required to be conveyed for EDPs*
-- *when the Basic BCP DP ASE and Advanced BCP DP ASE are used.*
-- *– For CS2 further study should be given for the feasibility of progressing of one of both*
-- *alternatives for both TDPs and EDPs.*
-- *– Every EDP must be explicitly armed by the SCF via a RequestReportBCSMEvent operation. No*
-- *implicit arming of EDPs at the SSF after reception of any operation (different from*
-- *RequestReportBCSMEvent) from the SCF is allowed.*

ResetTimer ::= OPERATION
ARGUMENT
ResetTimerArg
ERRORS {
MissingParameter,
ParameterOutOfRange,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SCF –> SSF, Timer: T_{rt}*
-- *This operation is used to request the SSF to refresh an application timer in the SSF.*

```
RouteSelectFailure ::= OPERATION
        ARGUMENT
            RouteSelectFailureArg
        ERRORS {
            MissingCustomerRecord,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- Direction: SSF –> SCF, Timer: Trsf
-- This operation is used to indicate that the SSP is unable to select a route (e.g. unable to determine a
-- correct route, no more routes on route list) or indication from the terminating half BCSM that a call
-- cannot be presented to the terminating party (e.g. network congestion) (DP 4 – Route_Select_Failure).
-- For additional information on this operation, refer to 4.2.2.2.1-4/Q.1214.

SelectFacility ::= OPERATION
        ARGUMENT
            SelectFacilityArg
        ERRORS {
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- Direction: SCF –> SSF, Timer: Tsf
-- This operation is used to request the SSF to perform the terminating basic call processing
-- actions to select the terminating line if it is idle, or selects an idle line from a multi-line hunt
-- group, or selects an idle trunk from a trunk group, as appropriate. If no idle line or trunk is
-- available, the SSF determines that the terminating facility is busy.

SelectRoute ::= OPERATION
        ARGUMENT
            SelectRouteArg
        ERRORS {
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }

-- Direction: SCF –> SSF, Timer: Tsr
-- This operation is used to request the SSF to perform the originating basic call processing actions to
-- determine routing information and select a route for a call, based either on call information available
-- to the SSF, or on call information provided by the SCF (e.g. for alternate routing), to include the
-- called party address, type of call, carrier, route index, and one or more alternate route indices.
-- Based on the routing information, the SSF attempts to select a primary route for the call, and if the
-- route is busy, attempts to select an alternate route. The SSF may fail to select a route for the call
-- if all routes are busy.

SendChargingInformation ::= OPERATION
        ARGUMENT
            SendChargingInformationArg
        ERRORS {
            MissingParameter,
            UnexpectedComponentSequence,
            UnexpectedParameter,
```

ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnknownLegID
}

-- *Direction: SCF –> SSF, Timer: T_{sci}*
-- *This operation is used to instruct the SSF on the charging information to send by the SSF. The charging*
-- *information can either be sent back by means of signalling or internal if the SSF is located in the local*
-- *exchange. In the local exchange this information may be used to update the charge meter or to create a*
-- *standard call record.*

ServiceFilteringResponse ::= OPERATION
ARGUMENT
ServiceFilteringResponseArg

-- *Direction: SSF –> SCF, Timer: T_{sfr}*
-- *This operation is used to send back to the SCF the values of counters specified in a previous*
-- *ActivateServiceFiltering operation.*

StatusReport ::= OPERATION
ARGUMENT
StatusReportArg

-- *Direction: SSF –> SCF, Timer: T_{srp}*
-- *This operation is used as a response to RequestFirstStatusMatchReport or*
-- *RequestEveryStatusChangeReport operations.*

TAnswer ::= OPERATION
ARGUMENT
TAnswerArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SSF –> SFC, Timer: T_{ta}*
-- *This operation is used to indicate that the call is accepted and answered by terminating party*
-- *(e.g. terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer message*
-- *received) (DP 15 – T_Answer). For additional information on this operation, refer to 4.2.2.2.2-10/Q.1214.*

TBusy ::= OPERATION
ARGUMENT
TBusyArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *Direction: SSF –> SCF, Timer: T_{tb}*
-- *This operation is used to indicate all resources in group busy (DP 13 – TBusy).*
-- *For additional information on this operation, refer to 4.2.2.2.2-8/Q.1214.*

TDisconnect ::= OPERATION
ARGUMENT
TDisconnectArg
ERRORS {
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,

```
                SystemFailure,
                TaskRefused,
                UnexpectedComponentSequence,
                UnexpectedDataValue,
                UnexpectedParameter
                }
-- Direction: SSF –> SCF, Timer: Ttd
-- This operation is used for a disconnect indication (e.g. onhook, Q.931 Disconnect message,
-- SS7 Release message) is received from the terminating party, or received from the originating party
-- via the originating half BCSM. (DP 17 – T_Disconnect.) For additional information on this operation,
-- refer to 4.2.2.2.2-10/Q.1214.

TermAttemptAuthorized ::= OPERATION
        ARGUMENT
                TermAttemptAuthorizedArg
        ERRORS {
                MissingCustomerRecord,
                MissingParameter,
                ParameterOutOfRange,
                SystemFailure,
                TaskRefused,
                UnexpectedComponentSequence,
                UnexpectedDataValue,
                UnexpectedParameter
                }
-- Direction: SSF –> SCF, Timer: Ttaa
-- This operation is used for indication of incoming call received from originating half BCSM and authority
-- to route call to a specified terminating resource (or group) verified. (DP 12 – Termination_Authorized.)
-- For additional information on this operation, refer to 4.2.2.2.2-7/Q.1214.

TMidCall ::= OPERATION
        ARGUMENT
                MidCallArg
        ERRORS {
                MissingCustomerRecord,
                MissingParameter,
                ParameterOutOfRange,
                SystemFailure,
                TaskRefused,
                UnexpectedComponentSequence,
                UnexpectedDataValue,
                UnexpectedParameter
                }
-- Direction: SSF –> SCF, Timer: Ttmc
-- This operation is used to indicate that a feature request is received from the terminating party (e.g. hook
-- flash, ISDN feature activation Q.931 HOLD or RETrieve message). (DP 16 – T_Mid_Call.)
-- For additional information on this operation, refer to 4.2.2.2.2-10/Q.1214.

TNoAnswer ::= OPERATION
        ARGUMENT
                TNoAnswerArg
        ERRORS {
                MissingCustomerRecord,
                MissingParameter,
                ParameterOutOfRange,
                SystemFailure,
                TaskRefused,
                UnexpectedComponentSequence,
                UnexpectedDataValue,
                UnexpectedParameter
                }
-- Direction: SSF –> SCF, Timer: Tna
-- This operation is used to indicate that the terminating party does not answer within a specified duration.
-- (DP 14 – T_No_Answer.) For additional information on this operation, refer to 4.2.2.2.2-9/Q.1214.
```

-- ***SCF-SRF operations***
-- *AssistRequestInstructions*
-- *SRF –> SCF*
-- *Refer to previous description of this operation in the SCF-SSF operations clause.*
-- *Cancel*
-- *SCF –> SRF*
-- *Refer to previous description of this operation in the SCF-SSF operations clause.*

```
PlayAnnouncement ::= OPERATION
        ARGUMENT
            PlayAnnouncementArg
        ERRORS {
            Cancelled,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter,
            UnavailableResource
            }
        LINKED {
            SpecializedResourceReport
            }
```

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

```
PromptAndCollectUserInformation ::= OPERATION
        ARGUMENT
            PromptAndCollectUserInformationArg
        RESULT
            ReceivedInformationArg
        ERRORS {
            Cancelled,
            ImproperCallerResponse,
            MissingParameter,
            ParameterOutOfRange,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnavailableResource,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- *Direction: SCF –> SRF, Timer: T_{pc}*
-- *This operation is used to interact with a user to collect information.*

```
SpecializedResourceReport ::= OPERATION
        ARGUMENT
            SpecializedResourceReportArg
```

-- *Direction: SRF –> SCF, Timer: T_{srr}*
-- *This operation is used as the response to a PlayAnnouncement operation when the announcement*
-- *completed report indication is set.*

```
END
```

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

Short: 1-10 seconds

Medium: 1-60 seconds

Long: 1 second-30 minutes

ffs For further study

Table 2 below lists all operation timers and the value range for each timer. The definitive value for each operation timer may be network specific and has to be defined by the network operator.

TABLE 2/Q.1218

Operation Name	Timer	Value range
ActivateServiceFiltering	T_{asf}	Medium
ActivityTest	T_{at}	Short
AnalyzeInformation	T_{ai}	ffs
AnalyzedInformation	T_{adi}	ffs
ApplyCharging	T_{ac}	Short
ApplyChargingReport	T_{acr}	Short
AssistRequestInstructions	T_{ari}	Short
CallGap	T_{cg}	Short
CallInformationReport	T_{cirp}	Short
CallInformationRequest	T_{cirq}	Short
Cancel	T_{can}	Short
CancelStatusReportRequest	T_{csr}	ffs
CollectedInformation	T_{cdi}	ffs
CollectInformation	T_{ci}	Medium
Connect	T_{con}	Short
ConnectToResource	T_{ctr}	Short
Continue	T_{cue}	Short
DisconnectForwardConnection	T_{dfc}	Short
EstablishTemporaryConnection	T_{etc}	Medium
EventNotificationCharging	T_{enc}	Short
EventReportBCSM	T_{erb}	Short
FurnishChargingInformation	T_{fci}	Short
HoldCallInNetwork	T_{hcn}	ffs
InitialDP	T_{idp}	Short
InitiateCallAttempt	T_{ica}	Short
OAnswer	T_{oa}	ffs
OCalledPartyBusy	T_{ob}	ffs
ODisconnect	T_{od}	ffs
OMidCall	T_{omc}	ffs
ONoAnswer	T_{ona}	ffs
OriginationAttemptAuthorized	T_{oaa}	ffs
ReleaseCall	T_{rc}	Short

TABLE 2/Q.1218 *(end)*

Operation Name	Timer	Value range
RequestCurrentStatusReport	T_{rcs}	ffs
RequestEveryStatusChangeReport	T_{res}	Short
RequestFirstStatusMatchReport	T_{rfs}	Short
RequestNotificationChargingEvent	T_{rnc}	Short
RequestReportBCSMEvent	T_{rrb}	Short
ResetTimer	T_{rt}	Short
RouteSelectFailure	T_{rsf}	ffs
SelectFacility	T_{sf}	ffs
SelectRoute	T_{sr}	ffs
SendChargingInformation	T_{sci}	Short
ServiceFilteringResponse	T_{sfr}	Short
StatusReport	T_{srp}	ffs
TAnswer	T_{ta}	ffs
TBusy	T_{tb}	ffs
TDisconnect	T_{td}	ffs
TermAttemptAuthorized	T_{taa}	ffs
TMidCall	T_{tmc}	ffs
TNoAnswer	T_{tna}	ffs
PlayAnnouncement	T_{pa}	Long
PromptAndCollectUserInformation	T_{pc}	Long
SpecializedResourceReport	T_{srr}	Short

2.1.2 IN CS-1 Error Types

```
IN-CS-1-Errors { ccitt recommendation q 1218 modules(0) cs-1-errors(1) version1(0) }
-- This module contains the type definitions for the IN CS-1 errors.
-- Where a parameter of type CHOICE is tagged with a specific tag value, the tag is automatically
-- replaced with an EXPLICIT tag of the same value.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS
        ERROR

FROM TCAPMessages { ccitt recommendation q 773 modules(0) messages(1) version2(2) }
        InvokeID,
        UnavailableNetworkResource

FROM IN-CS-1-DataTypes { ccitt recommendation q 1218 modules(0) cs-1-datatypes(2) version1(0) };
-- TYPE DEFINITION FOR IN CS-1 ERRORS FOLLOWS

Cancelled ::= ERROR
-- The operation has been cancelled.
```

```
CancelFailed ::= ERROR
        PARAMETER SEQUENCE {
            problem         [0]   ENUMERATED {
                                  unknownOperation(0),
                                  tooLate(1),
                                  operationNotCancellable(2)
                                  },
            operation       [1]   InvokeID
        }
-- The operation failed to be cancelled.

ETCFailed ::= ERROR
-- The establish temporary connection failed.

ImproperCallerResponse ::= ERROR
-- The caller response was not as expected.

MissingCustomerRecord ::= ERROR
-- The Service Logic Program could not be found in the SCF.

MissingParameter ::= ERROR
-- An expected optional parameter was not received.

ParameterOutOfRange ::= ERROR
-- The parameter was not as expected (e.g. missing or out of range).

RequestedInfoError ::= ERROR
        PARAMETER ENUMERATED {
            unknownRequestedInfo(1),
            requestedInfoNotAvailable(2)
            -- other values FFS
            }
-- The requested information cannot be found.

SystemFailure ::= ERROR
        PARAMETER
            unavailableNetworkResource  UnavailableNetworkResource
-- The operation could not be completed due to a system failure at the serving physical entity.

TaskRefused ::= ERROR
        PARAMETER ENUMERATED {
            generic(0),
            unobtainable (1),
            congestion(2)
            -- other values FFS
            }
-- An entity normally capable of the task requested cannot or chooses not to perform the task at this
-- time. This includes error situations like congestion and unobtainable address as used in (e.g. the
-- connect operation).

UnavailableResource ::= ERROR
-- A requested resource is not available at the serving entity.

UnexpectedComponentSequence ::= ERROR
-- An incorrect sequence of Components was received (e.g."DisconnectForwardConnection"
-- followed by "PlayAnnouncement").

UnexpectedDataValue ::= ERROR
-- The data value was not as expected (e.g. routing number expected but billing number received).

UnexpectedParameter ::= ERROR
-- A parameter received was not expected.

UnknownLegID ::= ERROR
-- Leg not known to the SSF.

UnknownResource ::= ERROR
-- Resource whose status is being requested is not known to the serving entity.


END
```

2.1.3 IN CS-1 Data Types

```
EXTENSION MACRO::=

BEGIN

TYPE NOTATION      ::= ExtensionType Criticality
VALUE NOTATION     ::= value(INTEGER)
ExtensionType      ::= "EXTENSION-SYNTAX" type
Criticality        ::= "CRITICALITY" value(CriticalityType)
CriticalityType    ::= ENUMERATED }
                           ignore(0),
                           abort(1)
                           }
END
-- 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 macro:
--
-- SomeNetworkSpecificIndicator ::= EXTENSION
-- EXTENSION-SYNTAX BOOLEAN
-- CRITICALITY abort
--
-- someNetworkSpecificIndicator SomeNetworkSpecificIndicator ::= 1
--
-- Example of transfer syntax, using the ExtensionField datatype as specified in the module
-- below. 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.

IN-CS-1-datatypes { ccitt recommendation q 1218 modules(0) cs-1-datatypes(2) version1(0) }
-- This module contains the type definitions for the IN CS-1 data types.
-- Where a parameter of type CHOICE is tagged with a specific tag value, the tag is automatically
-- replaced with an EXPLICIT tag of the same value.
-- The following parameters map onto bearer protocol (i.e. Q.931, case 2 and ISUP) parameters:
-- CallingPartySubaddress, CalledPartyNumber,
-- Prefix (derived from dialled digits), DestinationRoutingAddress,
-- DialledDigits, ISDNAccessRelatedInformation, CallingPartysCategory, LocationNumber,
-- TravellingClassMark, AssistingSSPIPRoutingAddress, AlertingPattern (Q.931 only),
-- ReleaseCause (and other Cause parameters),  ServiceProfileIdentifier (Q.932 only),
-- BearerCapability, CallingPartyNumber, HighLayerCompatibility, OriginalCalledPartyID,
-- RedirectingPartyID, and RedirectionInformation.
-- The procedures for mapping of parameters onto bearer protocol are ffs.

-- The following SSF parameters do not map onto bearer protocol (i.e. Q.931, case 2 and ISUP)
-- parameters and therefore are assumed to be local to the switching system: CallingPartyBusinessGroupID
-- FacilityGroup, FacilityGroupMember, RouteList, LegID, IPSSPCapabilities, IPAvailable, CGEncountered,
-- ForwardingCondition, CorrelationID, ApplicationTimer, TerminalType, MiscCallInfo, TriggerType and
-- ServiceKey.

-- Where possible, Administrations should specify the maximum size within their network of
-- parameters specified in this Recommendation that are of an indeterminate length.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS
        InvokeIdType
FROM TCAPMessages {ccitt recommendation q.773 modules(2) messages(1) version2(2)};
```

```
-- TYPE DEFINITIONS FOR IN CS-1 DATA TYPES FOLLOWS
-- Argument data types

-- The ordering of parameters in the argument sequences has been arbitrary. Further study may be
-- required to order arguments in a manner which will facilitate efficient encoding and decoding.

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

AnalysedInformationArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        dialledDigits                   [1]  CalledPartyNumber                  OPTIONAL,
        callingPartyBusinessGroupID     [2]  CallingPartyBusinessGroupID        OPTIONAL,
        callingPartySubaddress          [3]  CallingPartySubaddress             OPTIONAL,
        callingFacilityGroup            [4]  FacilityGroup                      OPTIONAL,
        callingFacilityGroupMember      [5]  FacilityGroupMember                OPTIONAL,
        originalCalledPartyID           [6]  OriginalCalledPartyID              OPTIONAL,
        prefix                          [7]  Digits                             OPTIONAL,
        redirectingPartyID              [8]  RedirectingPartyID                 OPTIONAL,
        redirectionInformation          [9]  RedirectionInformation             OPTIONAL,
        routeList                       [10] RouteList                          OPTIONAL,
        travellingClassMark             [11] TravellingClassMark                OPTIONAL,
        extensions                      [12] SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        featureCode                     [13] FeatureCode                        OPTIONAL,
        accessCode                      [14] AccessCode                         OPTIONAL,
        carrier                         [15] Carrier                            OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

AnalyseInformationArg ::= SEQUENCE {
        destinationRoutingAddress       [0]  DestinationRoutingAddress,
        alertingPattern                 [1]  AlertingPattern                    OPTIONAL,
        iSDNAccessRelatedInformation    [2]  ISDNAccessRelatedInformation       OPTIONAL,
        originalCalledPartyID           [3]  OriginalCalledPartyID              OPTIONAL,
        extensions                      [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        callingPartyNumber              [5]  CallingPartyNumber                 OPTIONAL,
        callingPartysCategory           [6]  CallingPartysCategory              OPTIONAL,
        calledPartyNumber               [7]  CalledPartyNumber                  OPTIONAL,
        chargeNumber                    [8]  ChargeNumber                       OPTIONAL,
        travellingClassMark             [9]  TravellingClassMark                OPTIONAL,
        carrier                         [10] Carrier                            OPTIONAL
--      ...
        }
ApplyChargingArg ::= SEQUENCE {
        aChBillingChargingCharacteristics [0]  AChBillingChargingCharacteristics,
        partyToCharge                   [2]  LegID                              OPTIONAL,
        extensions                      [3]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL
--      ...
        }
-- The partyToCharge parameter indicates the party in the call to which the ApplyCharging operation
-- should be applied. If it is not present, then it is applied to the A-party.
```

```
ApplyChargingReportArg ::= CallResult

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

CallGapArg ::= SEQUENCE {
        gapCriteria                     [0]  GapCriteria,
        gapIndicators                   [1]  GapIndicators,
        controlType                     [2]  ControlType                    OPTIONAL,
        gapTreatment                    [3]  GapTreatment                   OPTIONAL,
        extensions                      [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                 OPTIONAL
--      ...
        }
-- OPTIONAL denotes network operator optional. If gapTreatment is not present, the SSF will use
-- a default treatment depending on network operator implementation.

CallInformationReportArg ::= SEQUENCE {
        requestedInformationList        [0]  RequestedInformationList,
        correlationID                   [1]  CorrelationID                  OPTIONAL,
        extensions                      [2]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                 OPTIONAL
--      ...
        }
-- OPTIONAL denotes network operator optional.

CallInformationRequestArg ::= SEQUENCE {
        requestedInformationTypeList    [0]  RequestedInformationTypeList,
        correlationID                   [1]  CorrelationID                  OPTIONAL,
        extensions                      [2]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                 OPTIONAL
--      ...
        }
-- OPTIONAL denotes network operator optional.

CancelArg ::= CHOICE {
        invokeID                        [0]  InvokeID,
        allRequests                     [1]  NULL
        }
-- The InvokeID has the same value as that which was used for the operation to be cancelled.

CancelStatusReportRequestArg ::= SEQUENCE {
        resourceID                      [0]  ResourceID                     OPTIONAL,
        extensions                      [1]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                 OPTIONAL
--      ...
        }

CollectedInformationArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        dialledDigits                   [1]  CalledPartyNumber              OPTIONAL,
        callingPartyBusinessGroupID     [2]  CallingPartyBusinessGroupID    OPTIONAL,
        callingPartySubaddress          [3]  CallingPartySubaddress         OPTIONAL,
        callingFacilityGroup            [4]  FacilityGroup                  OPTIONAL,
        callingFacilityGroupMember      [5]  FacilityGroupMember            OPTIONAL,
        originalCalledPartyID           [6]  OriginalCalledPartyID          OPTIONAL,
        prefix                          [7]  Digits                         OPTIONAL,
        redirectingPartyID              [8]  RedirectingPartyID             OPTIONAL,
        redirectionInformation          [9]  RedirectionInformation         OPTIONAL,
```

```
        travellingClassMark             [10] TravellingClassMark                    OPTIONAL,
        extensions                      [11] SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                         OPTIONAL,
        featureCode                     [12] FeatureCode                            OPTIONAL,
        accessCode                      [13] AccessCode                             OPTIONAL,
        carrier                         [14] Carrier                                OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

CollectInformationArg ::= SEQUENCE {
        alertingPattern                 [0]  AlertingPattern                        OPTIONAL,
        numberingPlan                   [1]  NumberingPlan                          OPTIONAL,
        originalCalledPartyID           [2]  OriginalCalledPartyID                  OPTIONAL,
        travellingClassMark             [3]  TravellingClassMark                    OPTIONAL,
        extensions                      [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                         OPTIONAL,
        callingPartyNumber              [5]  CallingPartyNumber                     OPTIONAL,
        dialledDigits                   [6]  CalledPartyNumber                      OPTIONAL
--      ...
        }

ConnectArg ::= SEQUENCE {
        destinationRoutingAddress       [0]  DestinationRoutingAddress,
        alertingPattern                 [1]  AlertingPattern                        OPTIONAL,
        correlationID                   [2]  CorrelationID                          OPTIONAL,
        cutAndPaste                     [3]  CutAndPaste                            OPTIONAL,
        forwardingCondition             [4]  ForwardingCondition                    OPTIONAL,
        iSDNAccessRelatedInformation    [5]  ISDNAccessRelatedInformation           OPTIONAL,
        originalCalledPartyID           [6]  OriginalCalledPartyID                  OPTIONAL,
        routeList                       [7]  RouteList                              OPTIONAL,
        scfID                           [8]  ScfID                                  OPTIONAL,
        travellingClassMark             [9]  TravellingClassMark                    OPTIONAL,
        extensions                      [10] SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                         OPTIONAL,
        carrier                         [11] Carrier                                OPTIONAL,
        serviceInteractionIndicators    [26] ServiceInteractionIndicators           OPTIONAL,
        callingPartyNumber              [27] CallingPartyNumber                     OPTIONAL,
        callingPartysCategory           [28] CallingPartysCategory                  OPTIONAL,
        redirectingPartyID              [29] RedirectingPartyID                     OPTIONAL,
        redirectionInformation          [30] RedirectionInformation                 OPTIONAL
--      ...
        }
-- For alerting pattern, OPTIONAL denotes that this parameter only applies if SSF is the terminating
-- local exchange for the subscriber.

ConnectToResourceArg ::= SEQUENCE {
        resourceAddress                 CHOICE {
            ipRoutingAddress            [0]  IPRoutingAddress,
            legID                       [1]  LegID,
            both                        [2]  SEQUENCE {
                                             ipRoutingAddress  [0]  IPRoutingAddress,
                                             legID             [1]  LegID
                                             },
            none                        [3]  NULL
            },
        extensions                      [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                         OPTIONAL,
        serviceInteractionIndicators    [30] ServiceInteractionIndicators           OPTIONAL
--      ...
        }

DpSpecificCommonParameters ::= SEQUENCE {
        serviceAddressInformation       [0]  ServiceAddressInformation,
        bearerCapability                [1]  BearerCapability                       OPTIONAL,
        calledPartyNumber               [2]  CalledPartyNumber                      OPTIONAL,
        callingPartyNumber              [3]  CallingPartyNumber                     OPTIONAL,
        callingPartysCategory           [4]  CallingPartysCategory                  OPTIONAL,
```

```
        iPSSPCapabilities               [5]  IPSSPCapabilities                  OPTIONAL,
        iPAvailable                     [6]  IPAvailable                        OPTIONAL,
        iSDNAccessRelatedInformation    [7]  ISDNAccessRelatedInformation       OPTIONAL,
        cGEncountered                   [8]  CGEncountered                      OPTIONAL,
        locationNumber                  [9]  LocationNumber                     OPTIONAL,
        serviceProfileIdentifier        [10] ServiceProfileIdentifier           OPTIONAL,
        terminalType                    [11] TerminalType                       OPTIONAL,
        extensions                      [12] SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        chargeNumber                    [13] ChargeNumber                       OPTIONAL,
        servingAreaID                   [14] ServingAreaID                      OPTIONAL
--      ...
        }
-- OPTIONAL for iPSSPCapabilities, iPAvailable, and cGEncountered denotes network operator
-- specific use. OPTIONAL for callingPartyNumber, and callingPartysCategory refer to clause 3 for
-- the trigger detection point processing rules to specify when these parameters are included in the
-- message. bearerCapability should be appropriately coded as speech.

EstablishTemporaryConnectionArg ::= SEQUENCE {
        assistingSSPIPRoutingAddress    [0]  AssistingSSPIPRoutingAddress,
        correlationID                   [1]  CorrelationID                      OPTIONAL,
        legID                           [2]  LegID                              OPTIONAL,
        scfID                           [3]  ScfID                              OPTIONAL,
        extensions                      [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        carrier                         [5]  Carrier                            OPTIONAL,
        serviceInteractionIndicators    [30] ServiceInteractionIndicators       OPTIONAL
--      ...
        }

EventNotificationChargingArg ::= SEQUENCE {
        eventTypeCharging               [0]  EventTypeCharging,
        eventSpecificInformationCharging [1] EventSpecificInformationCharging  OPTIONAL,
        legID                           [2]  LegID                              OPTIONAL,
        extensions                      [3]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        monitorMode                     [30] MonitorMode            DEFAULT notifyAndContinue
--      ...
        }
-- OPTIONAL denotes network operator specific use.

EventReportBCSMArg ::= SEQUENCE {
        eventTypeBCSM                   [0]  EventTypeBCSM,
        bcsmEventCorrelationID          [1]  CorrelationID                      OPTIONAL,
        eventSpecificInformationBCSM    [2]  EventSpecificInformationBCSM       OPTIONAL,
        legID                           [3]  LegID                              OPTIONAL,
        miscCallInfo                    [4]  MiscCallInfo                       DEFAULT
                                                                    {messageType request},
        extensions                      [5]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL
--      ...
        }

FurnishChargingInformationArg ::= FCIBillingChargingCharacteristics

HoldCallInNetworkArg ::= CHOICE {
        holdcause                       [0]  HoldCause,
        empty                           [1]  NULL
        }
-- holdcause is optional and denotes network operator specific use.

InitialDPArg ::= SEQUENCE {
        serviceKey                      [0]  ServiceKey                         OPTIONAL,
        dialledDigits                   [1]  CalledPartyNumber                  OPTIONAL,
        calledPartyNumber               [2]  CalledPartyNumber                  OPTIONAL,
        callingPartyNumber              [3]  CallingPartyNumber                 OPTIONAL,
        callingPartyBusinessGroupID     [4]  CallingPartyBusinessGroupID        OPTIONAL,
        callingPartysCategory           [5]  CallingPartysCategory              OPTIONAL,
        callingPartySubaddress          [6]  CallingPartySubaddress             OPTIONAL,
```

```
        cGEncountered                   [7]  CGEncountered                   OPTIONAL,
        iPSSPCapabilities               [8]  IPSSPCapabilities               OPTIONAL,
        iPAvailable                     [9]  IPAvailable                     OPTIONAL,
        locationNumber                  [10] LocationNumber                  OPTIONAL,
        miscCallInfo                    [11] MiscCallInfo                    OPTIONAL,
        originalCalledPartyID           [12] OriginalCalledPartyID           OPTIONAL,
        serviceProfileIdentifier        [13] ServiceProfileIdentifier        OPTIONAL,
        terminalType                    [14] TerminalType                    OPTIONAL,
        extensions                      [15] SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                  OPTIONAL,
        triggerType                     [16] TriggerType                     OPTIONAL,
        highLayerCompatibility          [23] HighLayerCompatibility          OPTIONAL,
        serviceInteractionIndicators    [24] ServiceInteractionIndicators    OPTIONAL,
        additionalCallingPartyNumber    [25] AdditionalCallingPartyNumber    OPTIONAL,
        forwardCallIndicators           [26] ForwardCallIndicators           OPTIONAL,
        bearerCapability                [27] BearerCapability                OPTIONAL,
        eventTypeBCSM                   [28] EventTypeBCSM                   OPTIONAL,
        redirectingPartyID              [29] RedirectingPartyID              OPTIONAL,
        redirectionInformation          [30] RedirectionInformation          OPTIONAL
--      ...
        }
-- OPTIONAL for iPSSPCapabilities, iPAvailable, cGEncountered, and miscCallInfo denotes network
-- operator specific use.
-- OPTIONAL for dialledDigits, callingPartyNumber, and callingPartysCategory refer to clause 3 for the
-- trigger detection point processing rules to specify when these parameters are included in the message.
-- OPTIONAL for terminalType indicates that this parameter applies only at originating or terminating
-- local exchanges if the SSF has this information.

InitiateCallAttemptArg ::= SEQUENCE {
        destinationRoutingAddress       [0]  DestinationRoutingAddress,
        alertingPattern                 [1]  AlertingPattern                 OPTIONAL,
        iSDNAccessRelatedInformation    [2]  ISDNAccessRelatedInformation    OPTIONAL,
        travellingClassMark             [3]  TravellingClassMark             OPTIONAL,
        extensions                      [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                  OPTIONAL
        serviceInteractionIndicators    [29] ServiceInteractionIndicators    OPTIONAL,
        callingPartyNumber              [30] CallingPartyNumber              OPTIONAL
--      ...
        }

MidCallArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        calledPartyBusinessGroupID      [1]  CalledPartyBusinessGroupID      OPTIONAL,
        calledPartySubaddress           [2]  CalledPartySubaddress           OPTIONAL,
        callingPartyBusinessGroupID     [3]  CallingPartyBusinessGroupID     OPTIONAL,
        callingPartySubaddress          [4]  CallingPartySubaddress          OPTIONAL,
        featureRequestIndicator         [5]  FeatureRequestIndicator         OPTIONAL,
        extensions                      [6]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                  OPTIONAL,
        carrier                         [7]  Carrier                         OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

OAnswerArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        callingPartyBusinessGroupID     [1]  CallingPartyBusinessGroupID     OPTIONAL,
        callingPartySubaddress          [2]  CallingPartySubaddress          OPTIONAL,
        callingFacilityGroup            [3]  FacilityGroup                   OPTIONAL,
        callingFacilityGroupMember      [4]  FacilityGroupMember             OPTIONAL,
        originalCalledPartyID           [5]  OriginalCalledPartyID           OPTIONAL,
        redirectingPartyID              [6]  RedirectingPartyID              OPTIONAL,
        redirectionInformation          [7]  RedirectionInformation          OPTIONAL,
```

```
        routeList                   [8]  RouteList                          OPTIONAL,
        travellingClassMark         [9]  TravellingClassMark                OPTIONAL,
        extensions                  [10] SEQUENCE SIZE(1..numOfExtensions)  OF
                                         ExtensionField                     OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

OCalledPartyBusyArg ::= SEQUENCE {
        dpSpecificCommonParameters  [0]  DpSpecificCommonParameters,
        busyCause                   [1]  Cause                              OPTIONAL,
        callingPartyBusinessGroupID [2]  CallingPartyBusinessGroupID        OPTIONAL,
        callingPartySubaddress      [3]  CallingPartySubaddress             OPTIONAL,
        callingFacilityGroup        [4]  FacilityGroup                      OPTIONAL,
        callingFacilityGroupMember  [5]  FacilityGroupMember                OPTIONAL,
        originalCalledPartyID       [6]  OriginalCalledPartyID              OPTIONAL,
        prefix                      [7]  Digits                             OPTIONAL,
        redirectingPartyID          [8]  RedirectingPartyID                 OPTIONAL,
        redirectionInformation      [9]  RedirectionInformation             OPTIONAL,
        routeList                   [10] RouteList                          OPTIONAL,
        travellingClassMark         [11] TravellingClassMark                OPTIONAL,
        extensions                  [12] SEQUENCE SIZE(1..numOfExtensions)  OF
                                         ExtensionField                     OPTIONAL,
        carrier                     [13] Carrier                            OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

ODisconnectArg ::= SEQUENCE {
        dpSpecificCommonParameters  [0]  DpSpecificCommonParameters,
        callingPartyBusinessGroupID [1]  CallingPartyBusinessGroupID        OPTIONAL,
        callingPartySubaddress      [2]  CallingPartySubaddress             OPTIONAL,
        callingFacilityGroup        [3]  FacilityGroup                      OPTIONAL,
        callingFacilityGroupMember  [4]  FacilityGroupMember                OPTIONAL,
        releaseCause                [5]  Cause                              OPTIONAL,
        routeList                   [6]  RouteList                          OPTIONAL,
        extensions                  [7]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                         ExtensionField                     OPTIONAL,
        carrier                     [8]  Carrier                            OPTIONAL,
        connectTime                 [9]  Integer4                           OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

ONoAnswerArg ::= SEQUENCE {
        dpSpecificCommonParameters  [0]  DpSpecificCommonParameters,
        callingPartyBusinessGroupID [1]  CallingPartyBusinessGroupID        OPTIONAL,
        callingPartySubaddress      [2]  CallingPartySubaddress             OPTIONAL,
        callingFacilityGroup        [3]  FacilityGroup                      OPTIONAL,
        callingFacilityGroupMember  [4]  FacilityGroupMember                OPTIONAL,
        originalCalledPartyID       [5]  OriginalCalledPartyID              OPTIONAL,
        prefix                      [6]  Digits                             OPTIONAL,
        redirectingPartyID          [7]  RedirectingPartyID                 OPTIONAL,
        redirectionInformation      [8]  RedirectionInformation             OPTIONAL,
        routeList                   [9]  RouteList                          OPTIONAL,
        travellingClassMark         [10] TravellingClassMark                OPTIONAL,
        extensions                  [11] SEQUENCE SIZE(1..numOfExtensions)  OF
                                         ExtensionField                     OPTIONAL,
        carrier                     [12] Carrier                            OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.
```

```
OriginationAttemptAuthorizedArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        dialledDigits                   [1]  CalledPartyNumber                  OPTIONAL,
        callingPartyBusinessGroupID     [2]  CallingPartyBusinessGroupID        OPTIONAL,
        callingPartySubaddress          [3]  CallingPartySubaddress             OPTIONAL,
        callingFacilityGroup            [4]  FacilityGroup                      OPTIONAL,
        callingFacilityGroupMember      [5]  FacilityGroupMember                OPTIONAL,
        travellingClassMark             [6]  TravellingClassMark                OPTIONAL,
        extensions                      [7]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        carrier                         [8]  Carrier                            OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

PlayAnnouncementArg ::= SEQUENCE {
        informationToSend               [0]  InformationToSend,
        disconnectFromIPForbidden       [1]  BOOLEAN                            DEFAULT TRUE,
        requestAnnouncementComplete     [2]  BOOLEAN                            DEFAULT TRUE,
        extensions                      [3]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL
--      ...
        }

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

ReceivedInformationArg ::= CHOICE {
        digitsResponse                  [0]  Digits,
        iA5Response                     [1]  IA5String
        }
ReleaseCallArg ::= Cause
-- A default value of decimal 31 (normal unspecified) should be coded appropriately.

RequestCurrentStatusReportArg ::= ResourceID

RequestCurrentStatusReportResultArg ::= SEQUENCE {
        resourceStatus                  [0]  ResourceStatus,
        resourceID                      [1]  ResourceID                         OPTIONAL,
        extensions                      [2]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL
--      ...
        }

RequestEveryStatusChangeReportArg ::= SEQUENCE {
        resourceID                      [0]  ResourceID,
        correlationID                   [1]  CorrelationID                      OPTIONAL,
        monitorDuration                 [2]  Duration                           OPTIONAL,
        extensions                      [3]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL
--      ...
        }
-- For correlationID OPTIONAL denotes network operator optional.
-- monitorDuration is required if outside the context of a call. It is not expected if we are in the context
-- of a call, because in that case the end of the call implicitly means the end of the monitoring.

RequestFirstStatusMatchReportArg ::= SEQUENCE {
        resourceID                      [0]  ResourceID                         OPTIONAL,
        resourceStatus                  [1]  ResourceStatus                     OPTIONAL,
        correlationID                   [2]  CorrelationID                      OPTIONAL,
```

```
        monitorDuration          [3]  Duration                          OPTIONAL,
        extensions               [4]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                      ExtensionField                    OPTIONAL,
        bearerCapability         [5]  BearerCapability                  OPTIONAL
--      ...
        }
-- For correlationID OPTIONAL denotes network operator optional.
-- monitorDuration is required if outside the context of a call. It is not expected if we are in the context
-- of a call, because in that case the end of the call implicitly means the end of the monitoring.

RequestNotificationChargingEventArg ::= SEQUENCE SIZE(1..numOfChargingEvents)
            OF ChargingEvent

RequestReportBCSMEventArg ::= SEQUENCE {
        bcsmEvents               [0]  SEQUENCE SIZE(1..numOfBCSMEvents)  OF
                                                                         BCSMEvent,
        bcsmEventCorrelationID   [1]  CorrelationID                     OPTIONAL,
        extensions               [2]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                      ExtensionField                    OPTIONAL
--      ...
        }
-- Indicates the BCSM related events for notification.
-- For correlationID OPTIONAL denotes network operator optional.

ResetTimerArg ::= SEQUENCE {
        timerID                  [0]  TimerID                           DEFAULT tssf,
        timervalue               [1]  TimerValue,
        extensions               [2]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                      ExtensionField                    OPTIONAL
--      ...
        }

RouteSelectFailureArg ::= SEQUENCE {
        dpSpecificCommonParameters [0]  DpSpecificCommonParameters,
        dialledDigits            [1]  CalledPartyNumber                 OPTIONAL,
        callingPartyBusinessGroupID [2]  CallingPartyBusinessGroupID    OPTIONAL,
        callingPartySubaddress   [3]  CallingPartySubaddress            OPTIONAL,
        callingFacilityGroup     [4]  FacilityGroup                     OPTIONAL,
        callingFacilityGroupMember [5]  FacilityGroupMember            OPTIONAL,
        failureCause             [6]  Cause                             OPTIONAL,
        originalCalledPartyID    [7]  OriginalCalledPartyID             OPTIONAL,
        prefix                   [8]  Digits                            OPTIONAL,
        redirectingPartyID       [9]  RedirectingPartyID                OPTIONAL,
        redirectionInformation   [10] RedirectionInformation            OPTIONAL,
        routeList                [11] RouteList                         OPTIONAL,
        travellingClassMark      [12] TravellingClassMark               OPTIONAL,
        extensions               [13] SEQUENCE SIZE(1..numOfExtensions)  OF
                                      ExtensionField                    OPTIONAL,
        carrier                  [14] Carrier                           OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing
-- rules to specify when these parameters are included in the message.

SelectFacilityArg ::= SEQUENCE {
        alertingPattern          [0]  AlertingPattern                   OPTIONAL,
        destinationNumberRoutingAddress [1]  CalledPartyNumber          OPTIONAL,
        iSDNAccessRelatedInformation [2]  ISDNAccessRelatedInformation  OPTIONAL,
        calledFacilityGroup      [3]  FacilityGroup                     OPTIONAL,
        calledFacilityGroupMember [4]  FacilityGroupMember              OPTIONAL,
        originalCalledPartyID    [5]  OriginalCalledPartyID             OPTIONAL,
        extensions               [6]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                      ExtensionField                    OPTIONAL
--      ...
        }
-- OPTIONAL parameters are only provided if modifications desired to basic call processing values.
```

```
SelectRouteArg ::= SEQUENCE {
        destinationRoutingAddress       [0]  DestinationRoutingAddress,
        alertingPattern                 [1]  AlertingPattern                    OPTIONAL,
        correlationID                   [2]  CorrelationID                      OPTIONAL,
        iSDNAccessRelatedInformation    [3]  ISDNAccessRelatedInformation       OPTIONAL,
        originalCalledPartyID           [4]  OriginalCalledPartyID              OPTIONAL,
        routeList                       [5]  RouteList                          OPTIONAL,
        scfID                           [6]  ScfID                              OPTIONAL,
        travellingClassMark             [7]  TravellingClassMark                OPTIONAL,
        extensions                      [8]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        carrier                         [9]  Carrier                            OPTIONAL
--      ...
        }
-- OPTIONAL parameters are only provided if modifications desired to basic call processing values.

SendChargingInformationArg ::= SEQUENCE {
        sCIBillingChargingCharacteristics [0]  SCIBillingChargingCharacteristics,
        partyToCharge                     [1]  LegID,
        extensions                        [2]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                               ExtensionField                     OPTIONAL
--      ...
        }

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

SpecializedResourceReportArg ::=  NULL

StatusReportArg ::= SEQUENCE {
        resourceStatus                  [0]  ResourceStatus                     OPTIONAL,
        correlationID                   [1]  CorrelationID                      OPTIONAL,
        resourceID                      [2]  ResourceID                         OPTIONAL,
        extensions                      [3]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL,
        reportCondition                 [4]  ReportCondition                    OPTIONAL
--      ...
        }
-- For correlationID, OPTIONAL denotes network operator optional.
-- resourceID is required when the SSF sends a report as an answer to a previous request when the
-- correlationID was present.

TAnswerArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        calledPartyBusinessGroupID      [1]  CalledPartyBusinessGroupID         OPTIONAL,
        calledPartySubaddress           [2]  CalledPartySubaddress              OPTIONAL,
        calledFacilityGroup             [3]  FacilityGroup                      OPTIONAL,
        calledFacilityGroupMember       [4]  FacilityGroupMember                OPTIONAL,
        extensions                      [5]  SEQUENCE SIZE(1..numOfExtensions)  OF
                                             ExtensionField                     OPTIONAL
--      ...
        }

TBusyArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        busyCause                       [1]  Cause                              OPTIONAL,
        calledPartyBusinessGroupID      [2]  CalledPartyBusinessGroupID         OPTIONAL,
        calledPartySubaddress           [3]  CalledPartySubaddress              PTIONAL,
        originalCalledPartyID           [4]  OriginalCalledPartyID              OPTIONAL,
        redirectingPartyID              [5]  RedirectingPartyID                 OPTIONAL,
```

```
        redirectionInformation          [6]  RedirectionInformation                 OPTIONAL,
        routeList                       [7]  RouteList                              OPTIONAL,
        travellingClassMark             [8]  TravellingClassMark                    OPTIONAL,
        extensions                      [9]  SEQUENCE SIZE(1..numOfExtensions)      OF
                                             ExtensionField                         OPTIONAL
--      ...
        }
-- For the OPTIONAL parameters, refer to clause 3 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

TDisconnectArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        calledPartyBusinessGroupID      [1]  CalledPartyBusinessGroupID             OPTIONAL,
        calledPartySubaddress           [2]  CalledPartySubaddress                  OPTIONAL,
        calledFacilityGroup             [3]  FacilityGroup                          OPTIONAL,
        calledFacilityGroupMember       [4]  FacilityGroupMember                    OPTIONAL,
        releaseCause                    [5]  Cause                                  OPTIONAL,
        extensions                      [6]  SEQUENCE SIZE(1..numOfExtensions)      OF
                                             ExtensionField                         OPTIONAL,
        connectTime                     [7]  Integer4                               OPTIONAL
--      ...
        }

TermAttemptAuthorizedArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        calledPartyBusinessGroupID      [1]  CalledPartyBusinessGroupID             OPTIONAL,
        calledPartySubaddress           [2]  CalledPartySubaddress                  OPTIONAL,
        callingPartyBusinessGroupID     [3]  CallingPartyBusinessGroupID            OPTIONAL,
        originalCalledPartyID           [4]  OriginalCalledPartyID                  OPTIONAL,
        redirectingPartyID              [5]  RedirectingPartyID                     OPTIONAL,
        redirectionInformation          [6]  RedirectionInformation                 OPTIONAL,
        routeList                       [7]  RouteList                              OPTIONAL,
        travellingClassMark             [8]  TravellingClassMark                    OPTIONAL,
        extensions                      [9]  SEQUENCE SIZE(1..numOfExtensions)      OF
                                             ExtensionField                         OPTIONAL
--      ...
        }

TNoAnswerArg ::= SEQUENCE {
        dpSpecificCommonParameters      [0]  DpSpecificCommonParameters,
        calledPartyBusinessGroupID      [1]  CalledPartyBusinessGroupID             OPTIONAL,
        calledPartySubaddress           [2]  CalledPartySubaddress                  OPTIONAL,
        calledFacilityGroup             [3]  FacilityGroup                          OPTIONAL,
        calledFacilityGroupMember       [4]  FacilityGroupMember                    OPTIONAL,
        originalCalledPartyID           [5]  OriginalCalledPartyID                  OPTIONAL,
        redirectingPartyID              [6]  RedirectingPartyID                     OPTIONAL,
        redirectionInformation          [7]  RedirectionInformation                 OPTIONAL,
        travellingClassMark             [8]  TravellingClassMark                    OPTIONAL,
        extensions                      [9]  SEQUENCE SIZE(1..numOfExtensions)      OF
                                             ExtensionField                         OPTIONAL
--      ...
        }
-- The Definition of Common Data Types

AccessCode ::= LocationNumber
-- An access code from a business group dialling plan attendant access codes, access codes to escape
-- to the public network, access code to access a private facility/network, and feature access codes.
-- Uses the LocationNumber format which is based on the Q.763 Location Number format.
-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).
-- The Numbering Plan Indicator field shall be set to "Spare" (value 000).
-- Of local significance.

AChBillingChargingCharacteristics ::= OCTET STRING (SIZE (minAChBillingChargingLength..
                                                          maxAChBillingChargingLength))
-- The AChBillingChargingCharacteristics parameter specifies the charging related information
-- to be provided by the SSF and the conditions on which this information has to be reported
-- back to the SCF with the ApplyChargingReport operation.
-- Examples of charging related information to be provided by the SSF may be: bulk counter
-- values, costs, tariff change and time of charge, time stamps, durations, etc.
```

-- Examples of conditions on which the charging related information are to be reported may be:
-- threshold value reached, timer expiration, tariff change, end of connection configuration, etc.

AdditionalCallingPartyNumber ::= Digits
-- Indicates the Additional Calling Party Number. Refer to Recommendation Q.763 for encoding.

AlertingPattern ::= OCTET STRING (SIZE(3))
-- Indicates a specific pattern that is used to alert a subscriber (e.g. distinctive ringing, tones, etc.).
-- Only applies if SSF is the terminating local exchange for the subscriber. Refer to the Q.931
-- Signal parameter for encoding.

ApplicationTimer ::=INTEGER (0..2047)
-- Used by the SCF to set a timer in the SSF. The timer is in seconds.

AssistingSSPIPRoutingAddress ::= Digits
-- Indicates the destination address of the SRF for the assist procedure.

```
BCSMEvent ::= SEQUENCE {
        eventTypeBCSM               [0]  EventTypeBCSM,
        monitorMode                 [1]  MonitorMode,
        legID                       [2]  LegID                      OPTIONAL,

        dpSpecificCriteria          [30] DpSpecificCriteria         OPTIONAL
        }
```
-- Indicates the BCSM Event information for monitoring.

```
BearerCapability ::= CHOICE {
        bearerCap                   [0]  OCTET STRING
                                         (SIZE(2..maxBearerCapabilityLength)),
        tmr                         [1]  OCTET STRING (SIZE(1))
        }
```
-- Indicates the type of bearer capability connection to the user. For bearerCapability, either
-- DSS 1 (Q.931) or the ISUP User Service Information (Q.763) encoding can be used. Refer
-- to the Q.763 Transmission Medium Requirement parameter for tmr encoding.

CalledPartyBusinessGroupID ::= OCTET STRING
-- Indicates the business group of the called party. The value of this octet string is network
-- operator specific.

CalledPartyNumber ::= OCTET STRING (SIZE (minCalledPartyNumberLength..
maxCalledPartyNumberLength))
-- Indicates the Called Party Number. Refer to Recommendation Q.763 for encoding.

CalledPartySubaddress ::= OCTET STRING
-- Indicates the Called Party Subaddress. Refer to Recommendation Q.931 for encoding.

CallingPartyBusinessGroupID ::= OCTET STRING
-- Indicates the business group of the calling party. The value of this octet string is network
-- operator specific.

CallingPartyNumber ::= OCTET STRING (SIZE (minCallingPartyNumberLength..
maxCallingPartyNumberLength))
-- Indicates the Calling Party Number. Refer to Recommendation Q.763 for encoding.

CallingPartySubaddress ::= OCTET STRING
-- Indicates the Calling Party Subaddress. Refer to Recommendation Q.931 for encoding.

CallingPartysCategory ::= OCTET STRING (SIZE(1))
-- Indicates the type of calling party (e.g. operator, payphone, ordinary subscriber).
-- Refer to Recommendation Q.763 for encoding.

CallResult ::= OCTET STRING (SIZE (minCallResultLength..
maxCallResultLength))
-- This parameter provides the SCF with the charging related information previously requested
-- using the ApplyCharging operation. This shall include the partyToCharge parameter as
-- received in the related ApplyCharging operation to correlate the result to the request.
-- The remaining content is network operator specific.

```
-- Examples of charging related information to be provided by the SSF may be: bulk counter values,
-- costs, tariff change and time of change, time stamps, durations, etc.
-- Examples of conditions on which the charging related information are to be reported may be:
-- threshold value reached, timer expiration, tariff change, end of connection configuration, etc.

Carrier ::= OCTET STRING
-- Contains the carrier selection and carrier ID fields.
-- Carrier selection is one octet and is encoded as:
-- 00000000    No indication
-- 00000001    Selected carrier code pre subscribed and not input by calling party
-- 00000010    Selected carrier identification code pre subscribed and input by calling party
-- 00000011    Selected carrier identification code pre subscribed, no indication of whether input by calling party
-- 00000100    Selected carrier identification code not pre subscribed and input by calling party
-- 00000101
--      to     Spare
-- 11111110
-- 11111111    Reserved
--
-- Carrier ID has a one octet field indicating the number of digits followed by the digits encoded using BCD.
-- Detailed coding is for further study. It is of local significance and carrying it through the ISUP is for further
-- study.

Cause ::= OCTET STRING (SIZE (minCauseLength..maxCauseLength))
-- Indicates the cause for interface related information. Refer to the Q.763 Cause parameter for
-- encoding.
-- For the use of cause and location values refer to Recommendation Q.850.

CGEncountered ::= ENUMERATED {
        noCGencountered(0),
        manualCGencountered(1),
        scpOverload(2)
        }
-- Indicates the type of automatic call gapping encountered, if any.

ChargeNumber ::= LocationNumber
-- Information sent in either direction indicating the chargeable number for the call and consisting
-- of the odd/even indicator, nature of address indicator, numbering plan indicator, and address signals.
-- Uses the LocationNumber format which is based on the Q.763 Location Number format.
-- For example, the ChargeNumber may be a third party number to which a call is billed for the 3rd party
-- billing service. In this case, the calling party may request operator assistance to charge the call to,
-- for example, their home number.

ChargingEvent ::= SEQUENCE {
        eventTypeCharging       [0]  EventTypeCharging,
        monitorMode             [1]  MonitorMode,
        legID                   [2]  LegID                  OPTIONAL
        }
-- This parameter indicates the charging event type and corresponding
-- monitor mode and LedID.

CollectedDigits ::= SEQUENCE {
        minimumNbOfDigits       [0]  INTEGER (1..127)       DEFAULT 1,
        maximumNbOfDigits       [1]  INTEGER (1..127),
        endOfReplyDigit         [2]  OCTET STRING (SIZE (1..2))  OPTIONAL,
        cancelDigit             [3]  OCTET STRING (SIZE (1..2))  OPTIONAL,
        startDigit              [4]  OCTET STRING (SIZE (1..2))  OPTIONAL,
        firstDigitTimeOut       [5]  INTEGER (1..127)       OPTIONAL,
        interDigitTimeOut       [6]  INTEGER (1..127)       OPTIONAL,
        errorTreatment          [7]  ErrorTreatment         DEFAULT
                                                              reportErrorToScf,
        interruptableAnnInd     [8]  BOOLEAN                DEFAULT TRUE,
        voiceInformation        [9]  BOOLEAN                DEFAULT FALSE,
        voiceBack               [10] BOOLEAN                DEFAULT FALSE
        }
```

-- The use of voiceBack is network operator specific.
-- The endOfReplyDigit, cancelDigit, and startDigit parameters have been designated as OCTET STRING,
-- and are to be encoded as BCD, one digit per octet only, contained
-- in the four least significant bits of each OCTET. The usage is service dependent.

```
CollectedInfo ::= CHOICE {
        collectedDigits                 [0]   CollectedDigits,
        iA5Information                  [1]   BOOLEAN
        }

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

CorrelationID ::= Digits
```
-- used by SCF for correlation with a previous operation. Refer to clause 3 for a description of the
-- procedures associated with this parameter.

```
CounterAndValue ::= SEQUENCE {
        counterID                       [0]   CounterID,
        counterValue                    [1]   Integer4
        }

CounterID ::= INTEGER (0..99)
```
-- Indicates the counters to be incremented.
-- The counterIDs can be addressed by using the last digits of the dialled number.

```
CountersValue ::= SEQUENCE SIZE(0..numOfCounters) OF CounterAndValue

CutAndPaste ::= INTEGER (0..22)
```
-- Indicates the number of digits to be deleted. Refer to 6.4.2.16/Q.1214 for additional information.

```
DateAndTime ::= OCTET STRING (SIZE(6))
```
-- Indicates, amongst others, the start time for activate service filtering. Coded as YYMMDDHHMMSS
-- with each digit coded BCD.
-- The first octet contains YY and the remaining items are sequenced following.
-- For example, 1993 September 30th, 12:15:01 would be encoded as:

-- Bits	*HGFE*	*DCBA*
-- leading octet	*3*	*9*
--	*9*	*0*
--	*0*	*3*
--	*2*	*1*
--	*5*	*1*
--	*1*	*0*

```
DestinationRoutingAddress ::= SEQUENCE SIZE(1..3) OF CalledPartyNumber
```
-- Indicates the list of Called Party Numbers (primary and alternates).

```
Digits ::= OCTET STRING (SIZE (minDigitsLength..maxDigitsLength))
```
-- Indicates the address signalling digits. Refer to the Q.763 Generic Number and Generic Digits parameters
-- for encoding. The coding of the subfields 'NumberQualifier' in Generic Number and 'TypeOfDigits' in
-- Generic Digits are irrelevant to the INAP, the ASN.1 tags are sufficient to identify the parameter.
-- The ISUP format does not allow to exclude these subfields, therefore the value is network operator specific.
-- The following parameters should use Generic Number:
-- CorrelationID for AssistRequestInstructions, AssistingSSPIPRoutingAddress for
-- EstablishTemporaryConnection, calledAddressValue for all occurrences, callingAddressValue for all
-- occurrences. The following parameters should use Generic Digits: prefix, all
-- other CorrelationID occurrences, dialledNumber filtering criteria, callingLineID filtering criteria, lineID
-- for ResourceID type, digitResponse for ReceivedInformationArg.

```
DisplayInformation ::= IA5String (SIZE (minDisplayInformationLength..
                                        maxDisplayInformationLength))
```
-- Indicates the display information.

```
DpSpecificCriteria ::= CHOICE {
        numberOfDigits          [0]  NumberOfDigits,
        applicationTimer        [1]  ApplicationTimer
        }
-- The SCF may specify the number of digits to be collected by the SSF for the CollectedInfo event.
-- When all digits are collected, the SSF reports the event to the SCF.
-- The SCF may set a timer in the SSF for the No Answer event. If the user does not answer the call
-- within the allotted time, the SSF reports the event to the SCF.

Duration ::= INTEGER (-2..86400)
-- Values are seconds.

ErrorTreatment ::= ENUMERATED {
        reportErrorToScf(0),
        help(1),
        repeatPrompt(2)
        }
-- reportErrorToScf means returning the "ImproperCallerResponse" error in the event of an error
-- condition during collection of user info.

EventSpecificInformationBCSM ::= CHOICE {
        collectedInfoSpecificInfo       [0]  SEQUENCE {
                                        calledPartynumber       [0]  CalledPartyNumber
                                        -- ... --
                                        },
        analyzedInfoSpecificInfo        [1]  SEQUENCE {
                                        calledPartynumber       [0]  CalledPartyNumber
                                        -- ... --
                                        },
        routeSelectFailureSpecificInfo  [2]  SEQUENCE {
                                        failureCause            [0]  Cause          OPTIONAL
                                        -- ... --
                                        },
        oCalledPartyBusySpecificInfo    [3]  SEQUENCE {
                                        busyCause               [0]  Cause          OPTIONAL
                                        -- ... --
                                        },
        oNoAnswerSpecificInfo           [4]  SEQUENCE {
                                        -- no specific info defined --
                                        -- ... --
                                        },
        oAnswerSpecificInfo             [5]  SEQUENCE {
                                        -- no specific info defined --
                                        -- ... --
                                        },
        oMidCallSpecificInfo            [6]  SEQUENCE {
                                        connectTime             [0]  Integer4       OPTIONAL
                                        -- ... --
                                        },
        oDisconnectSpecificInfo         [7]  SEQUENCE {
                                        releaseCause            [0]  Cause          OPTIONAL,
                                        connectTime             [1]  Integer4       OPTIONAL
                                        -- ... --
                                        },
        tBusySpecificInfo               [8]  SEQUENCE {
                                        busyCause               [0]  Cause          OPTIONAL
                                        -- ... --
                                        },
        tNoAnswerSpecificInfo           [9]  SEQUENCE {
                                        -- no specific info defined --
                                        -- ... --
                                        },
        tAnswerSpecificInfo             [10] SEQUENCE {
                                        -- no specific info defined --
                                        -- ... --
                                        },
```

```
        tMidCallSpecificInfo        [11]  SEQUENCE {
                                    connectTime             [0]   Integer4      OPTIONAL
                                    -- ... --
                                    },
        tDisconnectSpecificInfo     [12]  SEQUENCE {
                                    releaseCause            [0]   Cause         OPTIONAL,
                                    connectTime             [1]   Integer4      OPTIONAL
                                    -- ... --
                                    }
        }
-- Indicates the call related information specific to the event.
-- The connectTime indicates the duration between the received answer indication from the called party side
-- and the release of the connection for ODisconnect, OException, TDisconnect, or TException events.
-- The unit for the connectTime is 100 milliseconds.

EventSpecificInformationCharging ::= OCTET STRING (SIZE
                                    (minEventSpecificInformationChargingLength..
                                        maxEventSpecificInformationChargingLength))
-- defined by network operator.
-- Indicates the charging related information specific to the event.
-- An example data type definition for this parameter is given below:
--          chargePulses            [0]  Integer4,
--          chargeMessages          [1]  OCTET STRING (SIZE (min..max))

EventTypeBCSM ::= ENUMERATED {
        origAttemptAuthorized(1),
        collectedInfo(2),
        analysedInformation(3),
        routeSelectFailure(4),
        oCalledPartyBusy(5),
        oNoAnswer(6),
        oAnswer(7),
        oMidCall(8),
        oDisconnect(9),
        oAbandon(10),
        termAttemptAuthorized(12),
        tBusy(13),
        tNoAnswer(14),
        tAnswer(15),
        tMidCall(16),
        tDisconnect(17),
        tAbandon(18)
        }
-- Indicates the BCSM detection point event. Refer to 4.2.2.2/Q.1214 for additional information on the
-- events. Values origAttemptAuthorized and termAttemptAuthorized can only be used for TDPs.

EventTypeCharging ::= OCTET STRING (SIZE (minEventTypeChargingLength..
                                    maxEventTypeChargingLength))
-- This parameter indicates the charging event type. Its content is network operator specific.
--
-- An example data type definition for this parameter is given below:
-- EventTypeCharging ::= ENUMERATED {
--                       chargePulses (0),
--                       chargeMessages (1)
--                       }

ExtensionField ::= SEQUENCE {
        type                        INTEGER,    -- shall identify the value of an EXTENSION type
        criticality                 ENUMERATED {
                                    ignore (0),
                                    abort (1)
                                    } DEFAULT ignore,
        value                       [1]   ANY DEFINED BY type
        }
-- This parameter indicates an extension of an argument data type. Its content is network operator specific.
```

```
FacilityGroup ::= CHOICE {
        trunkGroupID                [0]  INTEGER,
        privateFacilityID           [1]  INTEGER,
        huntGroup                   [2]  OCTET STRING,
        routeIndex                  [3]  OCTET STRING
        }
-- Indicates the particular group of facilities to route the call. huntGroup and routeIndex are encoded as
-- network operator specific.

FacilityGroupMember ::= INTEGER
-- Indicates the specific member of a trunk group or multi-line hunt group.

FCIBillingChargingCharacteristics ::= OCTET STRING (SIZE (minFCIBillingChargingLength..
                                     maxFCIBillingChargingLength))
-- This parameter indicates the billing and/or charging characteristics. Its content is network operator
-- specific. An example datatype definition for this parameter is given below:
-- FCIBillingChargingCharacteristics ::= CHOICE {
--      completeChargingrecord      [0]  OCTET STRING (SIZE (min..max)),
--      correlationID               [1]  CorrelationID,
--      scenario2Dot3               [2]  SEQUENCE {
--                                  chargeParty     [0]  LegID          OPTIONAL,
--                                  chargeLevel     [1]  OCTET STRING (SIZE (min..max))
--                                                                      OPTIONAL,
--                                  chargeItems     [2]  SET OF Attribute  OPTIONAL
--                                  }
--      }
-- Depending on the applied charging scenario, the following information elements can be included
-- (refer to Q.1214 Appendix II):
-- complete charging record (scenario 2.2)
-- charge party (scenario 2.3)
-- charge level (scenario 2.3)
-- charge items (scenario 2.3)
-- correlationID (scenario 2.4)

FeatureCode ::= LocationNumber
-- The two-digit feature code preceded by "*" or "11".
-- Uses the LocationNumber format which is based on the Q.763 Location Number format.
-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).
-- The Numbering Plan Indicator field shall be set to "Spare" (value 000).
-- Used for stimulus signalling (Q.932).

FeatureRequestIndicator ::= ENUMERATED {
        hold(0),
        retrieve(1),
        featureActivation(2),
        spare1(3),
        sparen(127)
        }
-- Indicates the feature activated (e.g. a switch-hook flash, feature activation). Spare values reserved
-- for future use.

FilteredCallTreatment ::= SEQUENCE {
        sFBillingChargingCharacteristics  [0]  SFBillingChargingCharacteristics,
        informationToSend                 [1]  InformationToSend            OPTIONAL,
        maximumNumberOfCounters           [2]  MaximumNumberOfCounters      OPTIONAL,
        releaseCause                      [3]  Cause                        OPTIONAL
        }
-- If releaseCause is not present, the default value is the same as the ISUP cause value decimal 31.
-- If informationToSend is present, the call will be released after the end of the announcement
-- with the indicated or default releaseCause.
-- If maximumNumberOfCounters is not present, ServiceFilteringResponse will be sent with
-- CountersValue::= SEQUENCE SIZE (0) OF CountersAndValue.

FilteringCharacteristics ::= CHOICE {
        interval                    [0]  INTEGER (-1..32000),
        numberOfCalls               [1]  Integer4
        }
```

```
-- Indicates the severity of the filtering and the point in time when the ServiceFilteringResponse is to be sent.
-- If = interval, every interval of time the next call leads to an InitialDP and a ServiceFilteringResponse is
-- sent to the SCF. The interval is specified in seconds.
-- If = NumberOfCalls, every N calls the Nth call leads to an InitialDP and a ServiceFilteringResponse
-- is sent to the SCF.
-- If ActivateServiceFiltering implies several counters – filtering on several dialled numbers –,
-- the numberOfCalls would include calls to all the dialled numbers.

FilteringCriteria ::= CHOICE {
        dialledNumber                   [0]  Digits,
        callingLineID                   [1]  Digits,
        serviceKey                      [2]  ServiceKey,
        addressAndService               [30] SEQUENCE {
                                             calledAddressValue    [0] Digits,
                                             serviceKey            [1] ServiceKey,
                                             callingAddressValue   [2] Digits         OPTIONAL,
                                             locationNumber        [3] LocationNumber
                                                                                    OPTIONAL
                                             }
        }
-- In case calledAddressValue is specified, the numbers to be filtered are from calledAddressValue
-- up to and including calledAddressValue + maximumNumberOfCounters-1.
-- The last two digits of calledAddressvalue can not exceed 100-maximumNumberOfCounters.

FilteringTimeOut ::= CHOICE {
        duration                        [0]  Duration,
        stopTime                        [1]  DateAndTime
        }
-- Indicates the maximum duration of the filtering. When the timer expires, a ServiceFilteringResponse
-- is sent to the SCF.

ForwardCallIndicators ::= OCTET STRING (SIZE(2))
-- Indicates the Forward Call Indicators. Refer to Recommendation Q.763 for encoding.

ForwardingCondition ::= ENUMERATED {
        busy(0),
        noanswer(1),
        any(2)
        }
-- Indicates the condition that must be met to complete the connect.

GapCriteria ::= CHOICE {
        calledAddressValue              [0]  Digits,
        gapOnService                    [2]  GapOnService,
        calledAddressAndService         [29] SEQUENCE {
                                             calledAddressValue    [0] Digits,
                                             serviceKey            [1] ServiceKey
                                             },
        callingAddressAndService        [30] SEQUENCE {
                                             callingAddressValue   [0] Digits,
                                             serviceKey            [1] ServiceKey,
                                             locationNumber        [2] LocationNumber
                                                                                    OPTIONAL
                                             }
        }
-- Both calledAddressValue and callingAddressValue can be
-- incomplete numbers, in the sense that a limited amount of digits can be given.
--
-- For the handling of numbers starting with the same digit string, refer to the detailed procedure
-- of the CallGap operation in 3.3.

GapOnService ::= SEQUENCE {
        serviceKey                      [0]  ServiceKey,
        dpCriteria                      [1]  EventTypeBCSM              OPTIONAL
        }
```

```
GapIndicators ::= SEQUENCE {
        duration                    [0]  Duration,
        gapInterval                 [1]  Interval
        }
-- Indicates the gapping characteristics. No gapping when gapInterval equals 0, and gap all calls when
-- gapInterval equals 1.

GapTreatment ::= CHOICE {
        informationToSend           [0]  InformationToSend,
        releaseCause                [1]  Cause,
        both                        [2]  SEQUENCE {
                                         informationToSend    [0]  InformationToSend,
                                         releaseCause         [1]  Cause
                                         }
        }
-- The default value for Cause is the same as in ISUP.

HighLayerCompatibility ::= OCTET STRING (SIZE (highLayerCompatibilityLength))
-- Indicates the teleservice. For encoding, DSS 1 (Q.931) is used.

HoldCause ::= OCTET STRING      -- defined by network operator.
-- Indicates the cause for holding the call.

InbandInfo ::= SEQUENCE {
        messageID                   [0]  MessageID,
        numberOfRepetitions         [1]  INTEGER (1..127)       OPTIONAL,
        duration                    [2]  INTEGER (0..32767)     OPTIONAL,
        interval                    [3]  INTEGER (0..32767)     OPTIONAL
        }
-- Interval is the time in seconds between each repeated announcement. Duration is the total
-- amount of time in seconds, including repetitions and intervals.
-- The end of announcement is either the end of duration or numberOfRepetitions, whatever comes first.
-- Duration with value 0 indicates infinite duration.

InformationToSend ::= CHOICE {
        inbandInfo                  [0]  InbandInfo,
        tone                        [1]  Tone,
        displayInformation          [2]  DisplayInformation
        }

Integer4 ::= INTEGER (0..2147483647)

Interval ::= INTEGER (–1..60000)
-- Units are milliseconds. A –1 value denotes infinite.

InvokeID ::= InvokeIdType
-- Operation invoke identifier.

IPAvailable ::= OCTET STRING (SIZE (minIPAvailableLength..maxIPAvailableLength))
-- defined by network operator.
-- Indicates that the resource is available.

IPRoutingAddress ::= CalledPartyNumber
-- Indicates the routing address for the IP.

IPSSPCapabilities ::= OCTET STRING (SIZE (minIPSSPCapabilitiesLength..
                                         maxIPSSPCapabilitiesLength))
-- defined by network operator.
-- Indicates the SRF resources available at the SSP.

ISDNAccessRelatedInformation ::= OCTET STRING

-- Indicates the destination user network interface related information. Refer to the Q.763 Access
-- Transport parameter for encoding.

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

-- Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use
-- with a choice of unilateral ID assignment or bilateral ID assignment.
-- OPTIONAL for LegID also denotes the following:
-- – when only one party exists in the call, this parameter is not needed (as no ambiguity exists);
-- – when more than one party exists in the call, one of the following alternatives applies:
-- 1. LegID is present and indicates which party is concerned.
-- 2. LegID is not present and a default value is assumed (e.g. calling party in the case of the
-- ApplyCharging operation).
-- Choice between these two alternatives is kept a network operator option.

```
LegType ::= OCTET STRING (SIZE(1))

leg1 LegType ::= '01'H

leg2 LegType ::= '02'H

LocationNumber ::= OCTET STRING (SIZE (minLocationNumberLength..
                                       maxLocationNumberLength))
```

-- Indicates the Location Number for the calling party. Refer to Recommendation Q.763 (White book) for encoding.

```
MaximumNumberOfCounters ::= INTEGER (1..numOfCounters)

MessageID ::= CHOICE {
        elementaryMessageID     [0]  Integer4,
        text                    [1]  SEQUENCE {
                                     messageContent      [0]  IA5String (SIZE
                                     (minMessageContentLength..
                                maxMessageContentLength)),
                                     attributes          [1]  OCTET STRING (SIZE (
                                     minAttributesLength..
                                     maxAttributesLength))
                                                              OPTIONAL
                                     },
        elementaryMessageIDs    [29] SEQUENCE SIZE (1..numOfMessageIDs) OF Integer4,
        variableMessage         [30] SEQUENCE {
                                     elementaryMessageID [0]  Integer4,
                                     variableParts       [1]  SEQUENCE SIZE (1..5)
                                                              OF VariablePart
                                     }
        }
```

-- OPTIONAL denotes network operator specific use.

```
MiscCallInfo ::= SEQUENCE {
        messageType             [0]  ENUMERATED {
                                     request(0),
                                     notification(1)
                                     },
dpAssignment                    [1]  ENUMERATED {
                                     individualLine(0),
                                     groupBased(1),
                                     officeBased(2)
                                     }                   OPTIONAL
        }
```

-- Indicates detection point related information.

```
MonitorMode         ::= ENUMERATED {
        interrupted(0),
        notifyAndContinue(1),
        transparent(2)
        }
```

-- Indicates the event is relayed and/or processed by the SSP.
-- If this parameter is used in the context of charging events, the following definitions apply for the
-- handling of charging events:
-- Interrupted means that the SSF notifies the SCF of the charging event using
-- EventNotificationCharging, does not process the event but discard it.

-- NotifyAndContinue means that SSF notifies the SCF of the charging event using
-- EventNotificationCharging, and continues processing the event or signal without waiting for SCF
-- instructions. Transparent means that the SSF does not notify the SCF of the event. This value is used to
-- end the monitoring of a previously requested charging event. Previously requested charging events are
-- monitored until ended by a transparent monitor mode, or until the end of the connection configuration.
-- For the use of this parameter in the context of BCSM events refer to 3.3.39.

NumberingPlan ::= OCTET STRING (SIZE(1))

-- Indicates the numbering plan for collecting the user information. Refer to the Q.763 Numbering Plan.
-- Indicator field for encoding.

NumberOfDigits ::= INTEGER (1..255)

-- Indicates the number of digits to be collected

OriginalCalledPartyID ::= OCTET STRING (SIZE (minOriginalCalledPartyIDLength..
maxOriginalCalledPartyIDLength))

-- Indicates the original called number. Refer to the Q.763 Original Called Number for encoding.

RedirectingPartyID ::= OCTET STRING (SIZE (minRedirectingPartyIDLength..
maxRedirectingPartyIDLength))

-- Indicates redirecting number. Refer to the Q.763 Redirecting number for encoding.

RedirectionInformation ::= OCTET STRING (SIZE(2))

-- Indicates redirection information. Refer to the Q.763 Redirection Information for encoding.

```
ReportCondition ::= ENUMERATED {
        statusReport(0),
        timerExpired(1),
        canceled(2)
        }
```

-- ReportCondition specifies the cause of sending "StatusReport" operation to the SCF.

RequestedInformationList ::= SEQUENCE SIZE (1..numOfInfoItems) OF RequestedInformation

RequestedInformationTypeList ::= SEQUENCE SIZE (1..numOfInfoItems) OF
RequestedInformationType

```
RequestedInformation ::= SEQUENCE {
        requestedInformationType        [0]  RequestedInformationType,
        requestedInformationValue       [1]  RequestedInformationValue
        }

RequestedInformationType ::= ENUMERATED {
        callAttemptElapsedTime(0),
        callStopTime(1),
        callConnectedElapsedTime(2),
        calledAddress(3),
        releaseCause(30)
        }

RequestedInformationValue ::= CHOICE {
        callAttemptElapsedTimeValue     [0]  INTEGER (0..255),
        callStopTimeValue               [1]  DateAndTime,
        callConnectedElapsedTimeValue   [2]  Integer4,
        calledAddressValue              [3]  Digits,
        releaseCauseValue               [30] Cause
        }
```

-- The callAttemptElapsedTimeValue is specified in seconds. The unit for the
-- callConnectedElapsedTimeValue is 100 milliseconds.

```
ResourceID ::= CHOICE {
        lineID                          [0]  Digits,
        facilityGroupID                 [1]  FacilityGroup,
```

```
        facilityGroupMemberID       [2]   INTEGER,
        trunkGroupID                [3]   INTEGER
        }
```

-- Indicates a logical identifier for the physical termination resource.

```
ResourceStatus ::= ENUMERATED {
        busy(0),
        idle(1)
        }

ResponseCondition ::= ENUMERATED {
        intermediateResponse(0),
        lastResponse(1)
```

-- additional values are for further study.

```
        }
```

-- ResponseCondition is used to identify the reason why ServiceFilteringResponse operation is sent.
-- intermediateresponse identifies that service filtering is running and the interval time is expired and
-- a call is received, or that service filtering is running and the threshold value is reached.
-- lastResponse identifies that the duration time is expired and service filtering has been finished or
-- that the stop time is met and service filtering has been finished.

```
RouteList ::= SEQUENCE SIZE(1..3) OF OCTET STRING (SIZE
                                (minRouteListLength..maxRouteListLength))
```

-- Indicates a list of trunk groups or a route index. See Recommendation Q.1214 for additional information on this item.

```
ScfID ::= OCTET STRING (SIZE (minScfIDLength..maxScfIDLength))
```

-- defined by network operator.
-- Indicates the SCF identifier.

```
SCIBillingChargingCharacteristics ::= OCTET STRING (SIZE (minSCIBillingChargingLength..
                                        maxSCIBillingChargingLength))
```

-- This parameter indicates the billing and/or charging characteristics. Its content is network operator
-- specific. An example datatype definition for this parameter is given below:
-- SCIBillingChargingCharacteristics ::= CHOICE {
-- chargeLevel [0] OCTET STRING (SIZE (min..max),
-- chargePulses [1] Integer4,
-- chargeMessages [2] OCTET STRING (SIZE (min..max)
-- }
-- Depending on the applied charging scenario the following information elements
-- can be included (refer to Appendix II/Q.1214):
-- chargeLevel (scenario 3.2)
-- chargePulses (scenario 3.2)
-- chargeMessages (scenario 3.2)

```
ServiceAddressInformation ::= SEQUENCE {
        serviceKey                  [0]   ServiceKey            OPTIONAL,
        miscCallInfo                [1]   MiscCallInfo,
        triggerType                 [2]   TriggerType           OPTIONAL
        }
```

-- Information that represents the result of trigger analysis and allows the SCF to choose the appropriate
-- service logic.

```
ServiceInteractionIndicators ::= OCTET STRING (SIZE ( minServiceInteractionIndicatorsLength..
                                        maxServiceInteractionIndicatorsLength))
```

-- Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services
-- and network based services, respectively between different IN based services.
-- The contents are network specific and identified as a subject for further study with respect to INAP.
-- The following example is listed to illustrate the use of this parameter:
-- CallToBeDiverted Allowed/NotAllowed Indicator

-- If the CallToBeDiverted indicator is set to NotAllowed, the destination exchange shall not allow any
-- divertion on the subjected call. By this, each service can pass the applicable indicators to inform the
-- destination exchange of how specific services are to be handled.

ServiceKey ::= Integer4

-- Information that allows the SCF to choose the appropriate service logic.

ServiceProfileIdentifier ::= OCTET STRING

-- Indicates a particular ISDN terminal. Refer to Recommendation Q.932 for encoding.

ServingAreaID ::= LocationNumber

-- Identifies the local serving area where a network provider operates. Uses the LocationNumber
-- format which is based on the Q.763 Location Number format.
-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).
-- The Numbering Plan Indicator field shall be set to "Spare" (value 000).
-- Defined by the network operator.

SFBillingChargingCharacteristics ::= OCTET STRING (SIZE (minSFBillingChargingLength..
maxSFBillingChargingLength))

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

```
TerminalType ::= ENUMERATED {
        unknown(0),
        dialPulse(1),
        dtmf(2),
        isdn(3),
        isdnNoDtmf(4),
        spare(16)
        }
```

-- Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability
-- (voice recognition, DTMF, display capability, etc.). Since present signalling systems do not convey
-- terminal type, this parameter applies only at originating or terminating local exchanges.

```
TimerID ::= ENUMERATED {
        tssf(0)
        -- others ffs
        }
```

-- Indicates the timer to be reset.

TimerValue ::= Integer4

-- Indicates the timer value (in seconds).

```
Tone ::= SEQUENCE {
        toneID          [0]  Integer4,
        duration        [1]  Integer4          OPTIONAL
        }
```

-- The duration specifies the length of the tone in seconds, value 0 indicates infinite duration.

TravellingClassMark ::= LocationNumber

-- Indicates travelling class mark information.
-- Uses the LocationNumber format which is based on the Q.763 Location Number format.
-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).
-- The Numbering Plan Indicator field shall be set to "Spare" (value 000).
-- Maximum 2 digits.

```
TriggerType ::= ENUMERATED {
        featureActivation(0),
        verticalServiceCode(1),
        customizedAccess(2),
```

```
        customizedIntercom(3),
        emergencyService(12),
        aFR(13),
        sharedIOTrunk(14),
        offHookDelay(17),
        channelSetupPRI(18),
        tNoAnswer(25),
        tBusy(26),
        oCalledPartyBusy(27),
        oNoAnswer(29),
        originationAttemptAuthorized(30),
        oAnswer(31),
        oDisconnect(32),
        termAttemptAuthorized(33),
        tAnswer(34),
        tDisconnect(35)
        -- Private (ffs)
        }

-- The type of trigger which caused call suspension
-- 4-11: Reserved; 15,16: Reserved; 19-24: Reserved

UnavailableNetworkResource ::= ENUMERATED {
        unavailableResources(0),
        componentFailure(1),
        basicCallProcessingException(2),
        resourceStatusFailure(3),
        endUserFailure(4)
        }

-- Indicates the network resource that failed.

VariablePart ::= CHOICE {
        integer     [0] Integer4,
        number      [1] Digits,                    -- Generic digits
        time        [2] OCTET STRING (SIZE(2)),    -- HH:MM, BCD coded
        date        [3] OCTET STRING (SIZE(3)),    -- YYMMDD, BCD coded
        price       [4] OCTET STRING (SIZE(4))
        }

-- Indicates the variable part of the message.
-- BCD coded variable parts are encoded as described in the examples below.
-- For example, time = 12:15 would be encoded as:
--      Bits            HGFE        DCBA
--      leading octet   2           1
--                      5           1
-- date = 1993 September 30th would be encoded as:
--      Bits            HGFE        DCBA
--      leading octet   3           9
--                      9           0
--                      0           3
-- The Definition of range of constants Follows

highLayerCompatibilityLength        INTEGER ::= 2
minAChBillingChargingLength         INTEGER ::=     -- network specific
maxAChBillingChargingLength         INTEGER ::=     -- network specific
minAttributesLength                 INTEGER ::=     -- network specific
maxAttributesLength                 INTEGER ::=     -- network specific
maxBearerCapabilityLength           INTEGER ::=     -- network specific
minCalledPartyNumberLength          INTEGER ::=     -- network specific
maxCalledPartyNumberLength          INTEGER ::=     -- network specific
minCallingPartyNumberLength         INTEGER ::=     -- network specific
maxCallingPartyNumberLength         INTEGER ::=     -- network specific
minCallResultLength                 INTEGER ::=     -- network specific
maxCallResultLength                 INTEGER ::=     -- network specific
```

minCauseLength **INTEGER ::= 2**
maxCauseLength **INTEGER ::=** -- *network specific*
minDigitsLength **INTEGER ::=** -- *network specific*
maxDigitsLength **INTEGER ::=** -- *network specific*
minDisplayInformationLength **INTEGER ::=** -- *network specific*
maxDisplayInformationLength **INTEGER ::=** -- *network specific*
minEventSpecificInformationChargingLength **INTEGER ::=** -- *network specific*
maxEventSpecificInformationChargingLength **INTEGER ::=** -- *network specific*
minEventTypeChargingLength **INTEGER ::=** -- *network specific*
maxEventTypeChargingLength **INTEGER ::=** -- *network specific*
minFCIBillingChargingLength **INTEGER ::=** -- *network specific*
maxFCIBillingChargingLength **INTEGER ::=** -- *network specific*
minIPAvailableLength **INTEGER ::=** -- *network specific*
maxIPAvailableLength **INTEGER ::=** -- *network specific*
minIPSSPCapabilitiesLength **INTEGER ::=** -- *network specific*
maxIPSSPCapabilitiesLength **INTEGER ::=** -- *network specific*
minLocationNumberLength **INTEGER ::=** -- *network specific*
maxLocationNumberLength **INTEGER ::=** -- *network specific*
minMessageContentLength **INTEGER ::=** -- *network specific*
maxMessageContentLength **INTEGER ::=** -- *network specific*
minOriginalCalledPartyIDLength **INTEGER ::=** -- *network specific*
maxOriginalCalledPartyIDLength **INTEGER ::=** -- *network specific*
minRedirectingPartyIDLength **INTEGER ::=** -- *network specific*
maxRedirectingPartyIDLength **INTEGER ::=** -- *network specific*
minRouteListLength **INTEGER ::=** -- *network specific*
maxRouteListLength **INTEGER ::=** -- *network specific*
minScfIDLength **INTEGER ::=** -- *network specific*
maxScfIDLength **INTEGER ::=** -- *network specific*
minSCIBillingChargingLength **INTEGER ::=** -- *network specific*
maxSCIBillingChargingLength **INTEGER ::=** -- *network specific*
minServiceInteractionIndicatorsLength **INTEGER ::=** -- *network specific*
maxServiceInteractionIndicatorsLength **INTEGER ::=** -- *network specific*
minSFBillingChargingLength **INTEGER ::=** -- *network specific*
maxSFBillingChargingLength **INTEGER ::=** -- *network specific*
numOfBCSMEvents **INTEGER ::=** -- *network specific*
numOfChargingEvents **INTEGER ::=** -- *network specific*
numOfCounters **INTEGER ::= 100**
numOfExtensions **INTEGER ::=** -- *network specific*
numOfInfoItems **INTEGER ::= 5**
numOfMessageIDs **INTEGER ::=** -- *network specific*

END

2.1.4 IN CS-1 application protocol (operation and error codes)

-- This module contains the operation and error code assignments for the IN CS-1 application protocol.

IN-CS-1-Codes { ccitt recommendation q 1218 modules(0) cs-1-codes(3) version1(0) }

DEFINITIONS ::=

BEGIN

-- OPERATION AND ERROR CODE ASSIGNMENTS FOR THE IN CS-1 PROTOCOL

-- FOLLOW

IMPORTS

-- macros

APPLICATION-SERVICE-ELEMENT

FROM Remote-Operations-Notation-Extension

{joint-iso-ccitt remote-operations(4) notation-extension(2)}

-- operation types

ActivateServiceFiltering,
ActivityTest,
AnalysedInformation,
AnalyseInformation,
ApplyCharging,
ApplyChargingReport,
AssistRequestInstructions,
CallGap,
CallInformationReport,
CallInformationRequest,
Cancel,
CancelStatusReportRequest,
CollectedInformation,
CollectInformation,
Connect,
ConnectToResource,
Continue,
DisconnectForwardConnection,
EstablishTemporaryConnection,
EventNotificationCharging,
EventReportBCSM,
FurnishChargingInformation,
HoldCallInNetwork,
InitialDP,
InitiateCallAttempt,
OAnswer,
OCalledPartyBusy,
ODisconnect,
OMidCall,
ONoAnswer,
OriginationAttemptAuthorized,
PlayAnnouncement,
PromptAndCollectUserInformation,
ReleaseCall,
RequestCurrentStatusReport,
RequestEveryStatusChangeReport,
RequestFirstStatusMatchReport,
RequestNotificationChargingEvent,
RequestReportBCSMEvent,
ResetTimer,
RouteSelectFailure,
SelectFacility,
SelectRoute,
SendChargingInformation,
ServiceFilteringResponse,
SpecializedResourceReport,
StatusReport,
TAnswer,
TBusy,
TDisconnect,
TermAttemptAuthorized,
TMidCall,
TNoAnswer

FROM IN-CS-1-Operations { ccitt recommendation q 1218 modules(0) cs-1-operations(0) version1(0) }

-- error types

Canceled,
CancelFailed,
ETCFailed,
ImproperCallerResponse,
MissingCustomerRecord,
MissingParameter,
ParameterOutOfRange,
RequestedInfoError,

```
        SystemFailure,
        TaskRefused,
        UnavailableResource,
        UnexpectedComponentSequence,
        UnexpectedDataValue,
        UnexpectedParameter,
        UnknownLegID,
        UnknownResource

FROM IN-CS-1-Errors { ccitt recommendation q 1218 modules(0) cs-1-errors(1) version1(0) };

-- the operations are grouped by the identified ASEs.

-- SCF activation ASE

   initialDP                          InitialDP                          ::= localValue 0

-- Basic BCP DP ASE

   originationAttemptAuthorized       OriginationAttemptAuthorized       ::= localValue 1
   collectedInformation               CollectedInformation               ::= localValue 2
   analysedInformation                AnalysedInformation                ::= localValue 3
   routeSelectFailure                 RouteSelectFailure                 ::= localValue 4
   oCalledPartyBusy                   OCalledPartyBusy                   ::= localValue 5
   oNoAnswer                          ONoAnswer                          ::= localValue 6
   oAnswer                            OAnswer                            ::= localValue 7
   oDisconnect                        ODisconnect                        ::= localValue 8
   termAttemptAuthorized              TermAttemptAuthorized              ::= localValue 9
   tBusy                              TBusy                              ::= localValue 10
   tNoAnswer                          TNoAnswer                          ::= localValue 11
   tAnswer                            TAnswer                            ::= localValue 12
   tDisconnect                        TDisconnect                        ::= localValue 13

-- Advanced BCP DP ASE

   oMidCall                           OMidCall                           ::= localValue 14
   tMidCall                           TMidCall                           ::= localValue 15

-- SCF/SRF activation of assist ASE

   assistRequestInstructions          AssistRequestInstructions          ::= localValue 16

-- Assist connection establishment ASE

   establishTemporaryConnection       EstablishTemporaryConnection       ::= localValue 17

-- Generic disconnect resource ASE

   disconnectForwardConnection        DisconnectForwardConnection        ::= localValue 18

-- Non-assisted connection establishment ASE

   connectToResource                  ConnectToResource                  ::= localValue 19

-- Connect ASE (elementary SSF function)

   connect                            Connect                            ::= localValue 20

-- Call handling ASE (elementary SSF function)

   holdCallInNetwork                  HoldCallInNetwork                  ::= localValue 21
   releaseCall                        ReleaseCall                        ::= localValue 22

-- BCSM Event handling ASE

   requestReportBCSMEvent             RequestReportBCSMEvent             ::= localValue 23
   eventReportBCSM                    EventReportBCSM                    ::= localValue 24
```

-- *Charging Event handling ASE*

requestNotificationChargingEvent	**RequestNotificationChargingEvent**	**::= localValue 25**
eventNotificationCharging	**EventNotificationCharging**	**::= localValue 26**

-- *SSF call processing ASE*

collectInformation	**CollectInformation**	**::= localValue 27**
analyseInformation	**AnalyseInformation**	**::= localValue 28**
selectRoute	**SelectRoute**	**::= localValue 29**
selectFacility	**SelectFacility**	**::= localValue 30**
continue	**Continue**	**::= localValue 31**

-- *SCF call initiation ASE*

initiateCallAttempt	**InitiateCallAttempt**	**::= localValue 32**

-- *Timer ASE*

resetTimer	**ResetTimer**	**::= localValue 33**

-- *Billing ASE*

furnishChargingInformation	**FurnishChargingInformation**	**::= localValue 34**

-- *Charging ASE*

applyCharging	**ApplyCharging**	**::= localValue 35**
applyChargingReport	**ApplyChargingReport**	**::= localValue 36**

-- *Status reporting ASE*

requestCurrentStatusReport	**RequestCurrentStatusReport**	**::= localValue 37**
requestEveryStatusChangeReport	**RequestEveryStatusChangeReport**	**::= localValue 38**
requestFirstStatusMatchReport	**RequestFirstStatusMatchReport**	**::= localValue 39**
statusReport	**StatusReport**	**::= localValue 40**

-- *Traffic management ASE*

callGap	**CallGap**	**::= localValue 41**

-- *Service management ASE*

activateServiceFiltering	**ActivateServiceFiltering**	**::= localValue 42**
serviceFilteringResponse	**ServiceFilteringResponse**	**::= localValue 43**

-- *Call report ASE*

callInformationReport	**CallInformationReport**	**::= localValue 44**
callInformationRequest	**CallInformationRequest**	**::= localValue 45**

-- *Signalling control ASE*

sendChargingInformation	**SendChargingInformation**	**::= localValue 46**

-- *Specialized resource control ASE*

playAnnouncement	**PlayAnnouncement**	**::= localValue 47**
promptAndCollectUserInformation	**PromptAndCollectUserInformation**	**::= localValue 48**
specializedResourceReport	**SpecializedResourceReport**	**::= localValue 49**

-- *Cancel ASE*

cancel	**Cancel**	**::= localValue 53**
cancelStatusReportRequest	**CancelStatusReportRequest**	**::= localValue 54**

-- *Activity Test ASE*

activityTest	**ActivityTest**	**::= localValue 55**

-- ERROR codes

cancelled	**Cancelled**	**::= localValue 0**
cancelFailed	**CancelFailed**	**::= localValue 1**
eTCFailed	**ETCFailed**	**::= localValue 3**
improperCallerResponse	**ImproperCallerResponse**	**::= localValue 4**
missingCustomerRecord	**MissingCustomerRecord**	**::= localValue 6**
missingParameter	**MissingParameter**	**::= localValue 7**
parameterOutOfRange	**ParameterOutOfRange**	**::= localValue 8**
requestedInfoError	**RequestedInfoError**	**::= localValue 10**
systemFailure	**SystemFailure**	**::= localValue 11**
taskRefused	**TaskRefused**	**::= localValue 12**
unavailableResource	**UnavailableResource**	**::= localValue 13**
unexpectedComponentSequence	**UnexpectedComponentSequence**	**::= localValue 14**
unexpectedDataValue	**UnexpectedDataValue**	**::= localValue 15**
unexpectedParameter	**UnexpectedParameter**	**::= localValue 16**
unknownLegID	**UnknownLegID**	**::= localValue 17**
unknownResource	**UnknownResource**	**::= localValue 18**

-- APPLICATION SERVICE ELEMENTS

SCF-activation-ASE ::= APPLICATION-SERVICE-ELEMENT
-- consumer is SSF
CONSUMER INVOKES {
initialDP
}
-- supplier is SCF
SUPPLIER INVOKES

SCF-SRF-activation-of-assist-ASE ::= APPLICATION-SERVICE-ELEMENT
-- consumer is SSF/SRF
CONSUMER INVOKES {
assistRequestInstructions
}
-- supplier is SCF
SUPPLIER INVOKES

Assist-connection-establishment-ASE ::= APPLICATION-SERVICE-ELEMENT
-- consumer is SSF
CONSUMER INVOKES
-- supplier is SCF
SUPPLIER INVOKES {
establishTemporaryConnection
}

Generic-disconnect-resource-ASE ::= APPLICATION-SERVICE-ELEMENT
-- consumer is SSF
CONSUMER INVOKES
-- supplier is SCF
SUPPLIER INVOKES {
disconnectForwardConnection
}

Non-assisted-connection-establishment-ASE ::= APPLICATION-SERVICE-ELEMENT
-- consumer is SSF
CONSUMER INVOKES
-- supplier is SCF
SUPPLIER INVOKES {
connectToResource
}

Connect-ASE ::= APPLICATION-SERVICE-ELEMENT
-- consumer is SSF
CONSUMER INVOKES
-- supplier is SCF
SUPPLIER INVOKES {
connect
}

```
BCSM-event-handling-ASE          ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
            eventReportBCSM
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            requestReportBCSMEvent
            }

DP-specific-event-handling-ASE   ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
            originationAttemptAuthorized,
            collectedInformation,
            analyzedInformation,
            routeSelectFailure,
            oCalledPartyBusy,
            oNoAnswer,
            oAnswer,
            oDisconnect,
            termAttemptAuthorized,
            TBusy,
            tNoAnswer,
            tAnswer,
            tDisconnect,
            oMidCall,
            tMidCall
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            requestReportBCSMEven
            }

Charging-event-handling-ASE      ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
            eventNotificationCharging
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            requestNotificationChargingEvent
            }

SCF-call-initiation-ASE  ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES
        -- supplier is SCF
        SUPPLIER INVOKES {
            initiateCallAttempt
            }

Timer-ASE    ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF/SRF
        CONSUMER INVOKES
        -- supplier is SCF
        SUPPLIER INVOKES {
            resetTimer
            }

Billing-ASE    ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES
        -- supplier is SCF
        SUPPLIER INVOKES {
            furnishChargingInformation
            }
```

```
Charging-ASE        ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
            applyChargingReport
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            applyCharging
            }

Traffic-management-ASE          ::= APPLICATION-SERVICE-ELEMENT
        --consumer is SSF
        CONSUMER INVOKES
        -- supplier is SCF
        SUPPLIER INVOKES {
            callGap
            }

Service-management-ASE          ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
            serviceFilteringResponse
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            activateServiceFiltering
            }

Call-report-ASE     ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
            callInformationReport
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            callInformationRequest
            }

Signalling-control-ASE          ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES
        -- supplier is SCF
        SUPPLIER INVOKES {
            sendChargingInformation
            }

Specialized-resource-control-ASE        ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF/SRF
        CONSUMER INVOKES {
            specializedResourceReport
            }
        -- supplier is SCF
        SUPPLIER INVOKES {
            playAnnouncement,
            promptAndCollectUserInformation
            }

Activity-test-ASE   ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF/SRF
        CONSUMER INVOKES
        -- supplier is SCF
        SUPPLIER INVOKES {
            activityTest
            }

Basic-BCP-DP-ASE            ::= APPLICATION-SERVICE-ELEMENT
        -- consumer is SSF
        CONSUMER INVOKES {
        originationAttemptAuthorized,
        collectedInformation,
```

```
            analyzedInformation,
            routeSelectFailure,
            oCalledPartyBusy,
            oNoAnswer,
            oAnswer,
            oDisconnect,
            termAttemptAuthorized,
            tCalledPartyBusy,
            tNoAnswer,
            tAnswer,
            tDisconnect
            }
            -- supplier is SCF
            SUPPLIER INVOKES

Advanced-BCP-DP-ASE             ::= APPLICATION-SERVICE-ELEMENT
            -- consumer is SSF
            CONSUMER INVOKES {
                  oMidCall,
                  tMidCall
                  }
            -- supplier is SCF
            SUPPLIER INVOKES

SSF-Call-Processing-ASE::= APPLICATION-SERVICE-ELEMENT
            -- consumer is SSF
            CONSUMER INVOKES
            -- supplier is SCF
            SUPPLIER INVOKES {
                  collectInformation,
                  analyzeInformation,
                  selectRoute,
                  selectFacility,
                  continue
            }

Status-reporting-ASE            ::= APPLICATION-SERVICE-ELEMENT
            -- consumer is SSF
            CONSUMER INVOKES {
                  statusReport
                  }
            -- supplier is SCF
            SUPPLIER INVOKES {
                  requestCurrentStatusReport,
                  requestEveryStatusChangeReport,
                  requestFirstStatusMatchReport
                  }

Cancel-ASE                      ::= APPLICATION-SERVICE-ELEMENT
            -- consumer is SSF/SRF
            CONSUMER INVOKES
            -- supplier is SCF
            SUPPLIER INVOKES {
                  cancel,
                  cancelStatusReportRequest
                  }

Call-handling-ASE               ::=  APPLICATION-SERVICE-ELEMENT
            -- consumer is SSF
            CONSUMER INVOKES
            -- supplier is SCF
            SUPPLIER INVOKES {
                  holdCallInNetwork,
                  releaseCall
                  }

END
```

2.1.5 Application Contexts

```
APPLICATION-CONTEXT MACRO::=

BEGIN
TYPE NOTATION              ::= Symmetric | InitiatorConsumerOfResponderConsumerOf | empty
VALUE NOTATION             ::= value(VALUE OBJECT IDENTIFIER)
Symmetric                  ::= "OPERATIONS OF" "{ASEList"}"
InitiatorConsumerOf        ::= "INITIATOR CONSUMER OF" "{ASEList"}" | empty
ResponderConsumerOf        ::= "RESPONDER CONSUMER OF" "{ASEList"}" | empty
ACEList                    ::= ASE | ASEList "," ASE
ASE                        ::= type - shall reference an APPLICATION-SERVICE-ELEMENT type

END

IN-CS1-SSF-to-SCF-Generic-AC        APPLICATION-CONTEXT
-- dialogue initiated by SSF with InitialDP operation
INITIATOR CONSUMER OF {
        SCF-activation-ASE,
        Assist-connection-establishment-ASE,
        Generic-disconnect-resource-ASE,
        Non-assisted-connection-establishment-ASE,
        Connect-ASE,
        Call-handling-ASE,
        BCSM-event-handling-ASE,
        Charging-event-handling-ASE,
        SSF-call-processing-ASE,
        Timer-ASE,
        Billing-ASE,
        Charging-ASE,
        Traffic-management-ASE,
        Status-reporting-ASE,
        Call-report-ASE,
        Signalling-control-ASE,
        Specialized-resource-control-ASE,
        Cancel-ASE,
        Activity-test-ASE
        }
RESPONDER CONSUMER OF
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) generic-ssf-to-scf(0) version 1(0)};

IN-CS1-SSF-to-SCF-DPspecific-AC     APPLICATION-CONTEXT
-- dialogue initiated by SSF with DP Specific Initial Operation
INITIATOR CONSUMER OF {
        Basic-BCP-DP-ASE,
        Advanced-BCP-DP-ASE,
        Assist-connection-establishment-ASE,
        Generic-disconnect-resource-ASE,
        Non-assisted-connection-establishment-ASE,
        Connect-ASE,
        Call-handling-ASE,
        DP-specific-event-handling-ASE,
        Charging-event-handling-ASE,
        SSF-call-processing-ASE,
        Timer-ASE,
        Billing-ASE,
        Charging-ASE,
        Traffic-management-ASE,
        Status-reporting-ASE,
        Call-report-ASE,
        Signalling-control-ASE,
        Specialized-resource-control-ASE,
        Cancel-ASE,
        Activity-test-ASE
        }
RESPONDER CONSUMER OF
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) dp-specific-ssf-to-scf(1) version 1(0)};
```

```
IN-CS1-assist-handoff-SSF-to-SCF-AC          APPLICATION-CONTEXT
-- dialogue initiated by SSF with AssistRequestInstructions
INITIATOR CONSUMER OF {
        SCF-SRF-activation-of-assist-ASE,
        Generic-disconnect-resource-ASE,
        Non-assisted-connection-establishment-ASE,
        Call-handling ASE,
        Timer-ASE,
        Billing-ASE,
        Charging-ASE,
        Status-reporting-ASE,
        Specialized-resource-control-ASE,
        Cancel-ASE,
        Activity-test-ASE
        }
RESPONDER CONSUMER OF
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) assist-handoff-ssf-to-scf(2) version 1(0)};

IN-CS1-SRF-to-SCF-AC                         APPLICATION-CONTEXT
-- dialogue initiated by SRF with AssistRequestInstructions
INITIATOR CONSUMER OF {
        SCF-SRF-activation-of-assist-ASE,
        Specialized-resource-control-ASE,
        Cancel-ASE
        }
RESPONDER CONSUMER OF
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) srf-to-scf(3) version 1(0)};

IN-CS1-SCF-to-SSF-AC                         APPLICATION-CONTEXT
-- dialogue initiated by SCF with InitiateCallAttempt, Generic Case
INITIATOR CONSUMER OF
RESPONDER CONSUMER OF {
        Assist-connection-establishment-ASE,
        Generic-disconnect-resource-ASE,
        Non-assisted-connection-establishment-ASE,
        Connect-ASE,
        Call-handling-ASE,
        BCSM-event-handling-ASE,
        Charging-event-handling-ASE,
        SSF-call-processing-ASE,
        SCF-call-initiation-ASE,
        Timer-ASE,
        Billing-ASE,
        Charging-ASE,
        Status-reporting-ASE,
        Call-report-ASE,
        Signalling-control-ASE,
        Specialized-resource-control-ASE,
        Cancel-ASE,
        Activity-test-ASE
        }
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) generic-scf-to-ssf(4) version 1(0)};

IN-CS1-SCF-to-SSF-AC                         APPLICATION-CONTEXT
-- dialogue initiated by SCF with InitiateCallAttempt, DP-specific Case
INITIATOR CONSUMER OF
RESPONDER CONSUMER OF {
        Assist-connection-establishment-ASE,
        Generic-disconnect-resource-ASE,
        Non-assisted-connection-establishment-ASE,
        Connect-ASE,
        Call-handling-ASE,
        DP-specific-event-handling-ASE,
        Charging-event-handling-ASE,
        SSF-call-processing-ASE,
        SCF-call-initiation-ASE,
        Timer-ASE,
        Billing-ASE,
        Charging-ASE,
        Status-reporting-ASE,
```

```
        Call-report-ASE,
        Signalling-control-ASE,
        Specialized-resource-control-ASE,
        Cancel-ASE,
        Activity-test-ASE
        }
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) dp-specific-scf-to-ssf(5) version 1(0)};

IN-CS1-SCF-to-SSF-traffic-management-AC    APPLICATION-CONTEXT
-- dialogue initiated by SCF with CallGap
INITIATOR CONSUMER OF
RESPONDER CONSUMER OF {
        Traffic-management-ASE
        }
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) scf-to-ssf-traffic-management(6) version 1(0)};

IN-CS1-SCF-to-SSF-service-management-AC    APPLICATION-CONTEXT
-- dialogue initiated by SCF with ActivateServiceFiltering
INITIATOR CONSUMER OF
RESPONDER CONSUMER OF {
        Service-management-ASE
        }
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) scf-to-ssf-service-management(7) version 1(0)};

IN-CS1-SSF-to-SCF-service-management-AC    APPLICATION-CONTEXT
-- dialogue initiated by SSF with ServiceFilteringResponse
INITIATOR CONSUMER OF {
        Service-management-ASE
        }
RESPONDER CONSUMER OF
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) ssf-to-scf-service-management(8) version 1(0)};

IN-CS1-SCF-to-SSF-status-reporting-AC    APPLICATION-CONTEXT
-- dialogue initiated by SCF with Status Reporting Operations
INITIATOR CONSUMER OF
RESPONDER CONSUMER OF {
        Cancel-ASE,
        Status-reporting-ASE
        }
::= {ccitt recommendation q1218 scf-ssf-srf-objects(1) scf-to-ssf-status-reporting(9) version 1(0)};
```

2.2 SCF-SDF interface

2.2.1 Introduction to the IN X.500 DAP Subset

2.2.1.1 Alignment between the X.500 concepts and the IN

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

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

The SDF is the entity responsible to answer the database requests. This functional entity can be mapped onto the Directory System Agent (DSA). When an association is setup between an SCF and a SDF, an instance of a DSA is created for the length of the association.

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

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

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

In the IN environment, the service provider is responsible for the management and the administration of the data contained in an DSA. Therefore the service provider plays the administrator role. He is the administrative authority in X.500 terminology. The service provider enforces the security procedure (authentication and access control).

2.2.1.2 Use of a limited subset of X.500

The primary purpose of the X.500-Series Recommendations is to provide a directory service and not the description of the SCF-SDF interface as SG 11 wants to use them. The X.500 functionalities cover more than the functionalities needed for IN CS-1. This subclause tries to indicate which aspects of the Directory Abstract Service should be considered and supported by implementations within the scope of CS-1. It also mentions the attitude to adopt when a not-supported parameter is received. Profiling is used as a means to present the status of the different parameters.

It is important to mention that the number of parameters carried in a message should be minimized, because each of them is associated to a load in the signalling traffic and to some processing time. This is the reason why the parameters are removed unless they are absolutely necessary when they are sent. On reception removed parameters should not be treated but should be understood by the receiving entity. This allows the extensions of the profile in the future according to its actual description in the 1993 edition of the Directory.

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

2.2.1.3 Working assumptions

Several assumptions were used to design the Directory Abstract Service profile for IN CS-1. References to the assumptions used are made in 2.2.2.1, they are as follows:

- **Assumption 1**: The version of the Directory Abstract Service used for CS-1 is the 1993 version. The parameters only used for the 1988 version shall be ignored. Functionalities that might be needed in future Capabilities Sets should be at least considered if not supported.

- **Assumption 2**: Data distribution mechanisms are not considered within CS-1. The reason behind that assumption is that the distribution mechanisms use the SDF-SDF interface that is not part of CS-1. Those mechanisms include the chaining and the broadcasting of requests as well as the referral. Even though they need not to be supported, those indications of the use of such mechanisms should be considered (see assumption 1).

- **Assumption 3**: Copy mechanisms are not considered within CS-1. Like the data distribution mechanisms, they use the SDF-SDF interface. They imply complex maintenance mechanisms that are not within the scope of CS-1.

- **Assumption 4**: The operations used for CS-1 do not need to be signed. Since no security requirement was expressed toward the use of signed operations like the definition of an algorithm for the signature, this functionality is not covered within the scope of CS-1.

- **Assumption 5**: The strong authentication mechanisms are not covered by the present profile. Those mechanisms are used for a security system with public keys. Given the present status of the security work, such a functionality is not foreseen for CS-1.
- **Assumption 6**: The alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed.
- **Assumption 7**: An SCF-SDF operation cannot be abandoned. If an operation takes too much time, its timer expires and there is no need to abandon it.

2.2.2 The IN X.500 DAP Subset

2.2.2.1 Review of X.511 for use in the IN

The profiling of Recommendation X.511 provided thereafter is based upon Recommendation X.511 and the amendments made to it (DAM 2 on contexts, DAM 3 on minor extensions)

2.2.2.1.1 Information types and common procedures

2.2.2.1.1.1 CommonArguments

```
IN-CommonArguments ::= CommonArguments (
    WITH COMPONENTS {
        serviceControls          (IN-ServiceControls....),
        securityParameters       OPTIONAL,
        requestor                ABSENT,
        operationProgress        ({nameResolutionPhase notStarted}....),
        aliasedRDNs              ABSENT,
        criticalExtensions       OPTIONAL
        referenceType            ABSENT,
        entryOnly                (TRUE....),
        exclusions               ABSENT,
        nameResolveOnMaster      (FALSE....)
        operationContexts        OPTIONAL},
        ...)
```

The status of each component of **CommonArguments** parameter is as follows:

– The **serviceControls** component (see 2.2.2.1.1.3).

– The **securityParameters** component – The securityParameters component governs the security features associated with a Directory operation (i.e. the signature of an operation). Since the operations are not signed in CS-1 (assumption 4), the securityParameters component should not be present.

– The **requestor** component provides the name of the originator of a particular operation. This is mainly used for signed requests and as such should not be sent (assumption 4).

– The **operationProgress**, **referenceType**, **entryOnly**, **exclusions** and **nameResolveOnMaster** components are used when the DUA is acting on a continuation reference from a DSA. Since the distribution mechanisms are not supported (assumption 2), those parameters should not be sent.

– The **aliasedRDNs** component is present in the 1993 version only for compatibility reasons. It should always be omitted in the 1993 implementations of the Directory (assumption 1).

– The **criticalExtensions** component is used to indicated to 1988 edition implementations the criticality of a number of extensions which are available in the 1993 versions. Since the 1993 version is used within CS-1 (assumption 1), the component should not be sent. However in the IN context the following extensions should be supported: subentries, modifyRightsRequest, matchedValuesOnly, useAliasOnUpdate, useContexts, selectionOnModify, extendedEntryMods.

– The **operationContexts** component supplies a set of context assertions which are applied to attribute value assertions and entry information made within the operation. Since the use of contexts is part of the IN requirements, the component is present whenever necessary.

2.2.2.1.1.2 Common results

```
IN-CommonResults ::= CommonResults (
        WITH COMPONENTS {
            securityParameters      ABSENT,
            performer               ABSENT,
            aliasDereferenced})
```

The status of each component of **CommonResults** parameter is as follows:

- The **securityParameters** component – The operations are not signed therefore this component should not be present.
- The **performer** component provides the name of the performer of a particular operation. This is mainly used for signed results and as such should not be sent (assumption 4).
- The **aliasDereferenced** component is set to TRUE when an alias was encountered and dereferenced during the processing of an operation.

2.2.2.1.1.3 Service controls

```
IN-ServiceControls ::= ServiceControls
        (WITH COMPONENTS {
            options                 ({chainingProhibited, localScope}|
                                    ({changingProhibited, localScope})
                                    {changingProhibited, localScope, dontDereferenceAliases, subentries}|
                                    {chainingProhibitecd, localScope, subentries}),
            priority                ,
            timeLimit               ABSENT,
            sizeLimit               ABSENT,
            scopeOfReferral         ABSENT,
            attributeSizeLimit      ABSENT})
```

The status of each component of **ServiceControls** parameter is as follows:

- The **options** component contains an number of indications:
  - **preferChaining** should be absent because it implies the use of distribution (assumption 2);
  - **chainingProhibited** and **localScope** should always be set, because chaining is not allowed in CS1 (assumption 2);
  - **dontUseCopy** and **copyShallDo** refer to copy mechanisms and should not be present (assumption 3);
  - **dontDereferenceAliases** should only be included when the object of interest is an alias (e.g. to remove an alias);
  - **subentries** should be set depending on whether the entries or the subentries need to be addressed.
- The **priority** component is used to specified at which priority the service is to be provided. This parameter could be used to manage congestion in the DSA and may be needed.
- The **timeLimit** component indicates the maximum elapsed time to fulfil a request. It is redundant with the operation timers of TCAP and therefore is not needed.
- The **sizeLimit** and the **attributeSizeLimit** sets some size limits on the results either in terms of objects or in terms of attributes. This is useful when requests are expected to be general (the requestor does not know the structure of the DSA), but in the case of IN, this type of limitation does not seem applicable.
- The **scopeOfReferral** component indicates the scope relevant for a referral. Since referral is not one of the features considered in IN CS1 (asssumption 2), the parameter should not be present.

2.2.2.1.1.4 Entry information selection

```
IN-EntryInformationSelection ::= EntryInformationSelection
        (WITH COMPONENTS {
            attributes              ,
            infoTypes               (attributeTypesAndValues....),
            extraAttributes         OPTIONAL
```

```
        contextSelection        OPTIONAL,
        returnContexts          },
        ...)
```

The status of each component of **EntryInformationSelection** parameter is as follows:

- The **attributes** component specifies the attributes that should be returned in a retrieval service. The **allUserAttributes** option is kept even though it is advised to service specifiers to avoid its use which generates more traffic than needed. They should use instead the **select** option that precisely name the requested attributes.
- The **infoTypes** component specifies whether the attribute types and values should be returned or only the types. In IN services are mainly interested in the attribute values that are relevant to the processing of the service. This component should be absent given its default value.
- The **extraAttributes** component has a similar use as the **attributes** component. The only difference is that the option **allUserAttributes** is replaced by **allOperationalAttributes**.
- The **contextSelection** component is used to specify which attribute values should be returned of the attributes selected by **attributes** when contexts are used. Therefore this component is of interest for IN specifications.
- The **returnContexts** component is to request the SDF to return attribute values with their associated context lists. The component can be used for some IN services.

2.2.2.1.1.5 EntryInformation

```
IN-EntryInformation ::= EntryInformation
    (WITH COMPONENTS {
        name                    PRESENT,
        fromEntry               (TRUE....),
        information             (WITH COMPONENTS {
                                    attributeType   ABSENT,
                                    attribute       PRESENT}....) OPTIONAL,
        incompleteEntry         ,
        partialNameResolution   (FALSE....)},
        ...)
```

The status of each component of **EntryInformation** parameter is as follows:

- The **name** component gives the name of the entry returned. It is a mandatory component.
- The **fromEntry** component indicates if a copy or the entry itself is returned. Since IN CS-1 does not use copy mechanisms (assumption 3), only the default value of this component should be used.
- The **information** parameter contains the relevant information which is returned. Given the choice made for the **infoTypes** component (see 2.2.2.1.1.4), only the **attribute** option should be used.
- The **incompleteEntry** parameter indicates when the returned result is not complete due to some access rights limitations for example. It is necessary to support such a parameter.
- The **partialNameResolution** component indicates whether the name resolution was partially or totally completed. This component does not correspond to any IN requirements and is not included.

2.2.2.1.1.6 Optionally-signed parameters

```
UNSIGNED {Type} ::= OPTIONALLY-SIGNED {Type}
    (WITH COMPONENTS {
        unsigned        PRESENT,
        signed          ABSENT})
```

The operations are not signed for CS-1 (assumption 4). This means that only the unsigned option should be used.

2.2.2.1.2 Bind operation

```
in-DirectoryBind OPERATION ::= {
        ARGUMENT        IN-DirectoryBindArgument
        RESULT          IN-DirectoryBindResult
        ERRORS          {in-DirectoryBindError}}
```

The DirectoryBind and DirectoryUnbind operations are used at the beginning and end of a particular period of accessing the Directory.

2.2.2.1.2.1 Bind arguments and results

```
IN-DirectoryBindArgument ::= DirectoryBindArgument
        (WITH COMPONENTS  {
            credentials         (IN-Credentials) OPTIONAL,
            versions            })

IN-Credentials ::= Credentials
        (WITH COMPONENTS {
            simple                  ,
            strong                  ABSENT,
            externalProcedure       })

IN-DirectoryBindResult ::= IN-DirectoryBindArgument
```

The status of each parameter of **DirectoryBindArgument** is as follows:

- The **credentials** parameter is used to establish the identity of the user. It is necessary for services that requires authentication. However, the **strong** credentials should not be used because they correspond to the use of a public key system (assumption 5).
- The **versions** parameter identifies the version of the DirectoryService which is to be used.

The same considerations apply to the DirectoryBindResult.

2.2.2.1.2.2 Bind errors

```
in-DirectoryBindError ERROR ::= {
PARAMETER IN-DirectoryBindErrorParameter}

IN-DirectoryBindErrorParameter ::= DirectoryBindErrorParameter
        (WITH COMPONENTS {
            versions            ,
            error           (WITH COMPONENTS {
                                securityError       (SecurityProblem (1/2/7....)),
                                serviceError        (ServiceProblem (2))})})
```

The **DirectoryBindError** remains as defined in the Directory Abstract Service. In reception, all the possible errors should be supported to understand a Bind error.

2.2.2.1.3 Search operation

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

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

2.2.2.1.3.1 Search arguments and errors

```
IN-SearchArgument ::= UNSIGNED {SearchArgument
        (WITH COMPONENTS {
            baseObject              PRESENT,
            subset                  ,
            filter                  ,
            searchAliases           (TRUE....),
            selection               (IN-EntryInformationSelection....),
            pagedResults            ABSENT,
```

```
        matchedValuesOnly       ,
        extendedFilter          ABSENT,
        checkOverspecified      },
        ...)}
```

The status of each parameter of **SearchArgument** is as follows:

- The **baseObject** parameter identifies the object entry where the search operation should take place. It is a mandatory parameter.
- The **subset** parameter indicates which part of the DIT should be involved in the search operation. This parameter is needed even though the use of the wholeSubtree should be avoided to prevent the operation timer to expire.
- The **filter** parameter is used to eliminate entries from the search space. However the **extendedFilter** parameter was added in the 1993 version of the Directory for compatibility reasons and should therefore not be sent. Only the **filter** parameter should be sent.
- The **searchAliases** parameter indicates whether the aliases encountered in the search space (except the base object) should be considered. Since in IN aliases are always dereferenced when searching, this parameter should be used only with its default value.
- The **selection** parameter indicates what information from the entries, e.g. types and values, is requested (see 2.2.2.1.1.4).
- The **pagedResults** parameter is used to request a page by page result. The **pagedResults** parameter is used to present the results of a search operation in a page format. This type of information is not needed in CS-1 since the SCF treats the results.
- The **matchedValuesOnly** is a facility used to return only the values that fulfil the filter if any. This might be needed for IN CS-1.
- The **checkOverspecified** parameter is used to request the SDF to return an **overspecFilter** item in the Search result when appropriate. The IN services might need an **overspecFilter** item and therefore this parameter should be included.

Concerning errors, the **abandoned** and the **referral** errors are not supported.

2.2.2.1.3.2 Search results

```
IN-SearchResult ::= UNSIGNED {SearchResult
        (WITH COMPONENTS {
        searchInfo          (WITH COMPONENTS {
            name                ,
            entries             (WITH COMPONENT (IN-EntryInformation...)...),
                partialOutcomeQualifier (PartialOutcomeQualifier
                    (WITH COMPONENTS {
                        limitProblem                OPTIONAL,
                        unexplored                  ABSENT,
                        unavailableCriticalExtensions ,
                        unknownErrors               ABSENT,
                        queryReference              ABSENT,
                        overspecFilter              OPTIONAL},
                        ...) ....)},
                ...),
        uncorrelatedSearchInfo  ABSENT},
        ...)}
```

The status of each parameter of **SearchResult** is as follows:

- The **name** parameter indicates the name of the returned entries if different from the name of the **baseObject** name. It is needed when the search operation has entries other than the base object as a result.
- The **entries** parameter contains the entries that satisfy the filter (see 2.2.2.1.1.5).

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

  - The **limitProblem** parameter indicates which type of limit was reached (administrative). It is necessary to carry such indications.
  - The **unexplored** parameter indicates where the search operation was stopped and gives indications about the state of the operation (object considered and progress indication). If the operation is to be resent, such a parameter is needed. This is equivalent to a referral and as such should be absent (assumption 2).
  - The **unavailableCriticalExtensions** parameter indicates that some critical extensions were not available.
  - The **unknownErrors** parameter corresponds to the reception of unknown errors from other SDF. It implies the use of a chaining mechanism that is not available in CS-1 (assumption 2) and is not used.
  - The **queryReference** parameter is used when paged results were requested and therefore is not needed.
  - The **overspecFilter** parameter is used when the returned Search result is empty as a consequence of over-specified filtering. The filter returned corresponds to a part of the initial filter for which some entries are still available. This functionalities could be used for IN services.

– The **uncorrelatedSearchInfo** contains signed results that could not be unsigned. Since the operations are not signed in IN CS-1, this parameter should not be used.

2.2.2.1.4 AddEntry operation

```
in-AddEntry OPERATION ::= {
        ARGUMENT  IN-AddEntryArgument
        RESULT    AddEntryResult
        ERRORS    {nameError | in-ServiceError | in-SecurityError | attributeError |
                  in-UpdateError}
        CODE      id-opcode-in-addEntry}
```

The addEntry operation is used to add a leaf entry to the DIT.

2.2.2.1.4.1 AddEntry arguments, results and errors

```
IN-AddEntryArgument ::= UNSIGNED {AddEntryArgument
        (WITH COMPONENTS {
            object          PRESENT,
            entry           PRESENT,
            targetSystem    ABSENT})}
```

The status of each parameter of **AddEntryArgument** is as follows:

– The **object** parameter identifies the entry to be added. It is a mandatory parameter.

– The **entry** parameter describes the entry to be added. It is also a mandatory parameter.

– The **targetSystem** parameter contains the name of the SDF where the entry was added. When no distribution is considered (assumption 2), the SDF is the same as the one where the operation was sent. The parameter is therefore unnecessary.

Concerning errors, the **referral** error is not supported.

2.2.2.1.5 RemoveEntry operation

```
in-RemoveEntry OPERATION ::= {
        ARGUMENT    IN-RemoveEntryArgument
        RESULT      RemoveEntryResult
        ERRORS      {nameError | in-ServiceError | in-SecurityError | in-UpdateError}
        CODE        id-opcode-in-removeEntry}
```

The removeEntry operation is used to remove a leaf entry from the DIT.

2.2.2.1.5.1 RemoveEntry arguments, results and errors

IN-RemoveEntryArgument ::= UNSIGNED {RemoveEntryArgument}

The parameters are only the name of the object to be deleted. It is mandatory. The argument of the operation should be kept as it is described in the Directory.

Concerning errors, the **referral** error is not supported.

2.2.2.1.6 ModifyEntry operation

```
in-ModifyEntry OPERATION ::= {
        ARGUMENT        IN-ModifyEntryArgument
        RESULT          IN-ModifyEntryResult
        ERRORS          {nameError | in-ServiceError | in-SecurityError | attributeError|
                        in-UpdateError}
        CODE            id-opcode-in-modifyEntry}
```

The modifyEntry operation is used to perform a series of modifications to a single entry.

2.2.2.1.6.1 ModifyEntry arguments and errors

```
IN-ModifyEntryArgument ::= UNSIGNED {ModifyEntryArgument
        (WITH COMPONENTS  {
            object          PRESENT,
            changes         PRESENT,
            selection       (IN-EntryInformationSelection)})}
```

The parameters are the name of the object to be modified, the list of changes and the **selection** parameter that specifies some attributes and values to be returned. The first two are mandatory and the third one corresponds to 2.2.2.1.1.4. Since all the changes available in the **changes** parameter are useful for IN CS-1, the argument of the operation should be kept as it is described in the Directory except the selection parameter.

Concerning errors, the **referral** error is not supported.

2.2.2.1.6.2 ModifyEntry results

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

If no information was to be retrieved with the modifyEntry operation, the **null** result is returned. Otherwise the information is to be returned in the **entry** component of the **information** result. For IN CS-1 this component is specified in 2.2.2.1.15.

2.2.2.1.7 Errors

The precedence rule defined in the Directory should apply.

The **abandoned**, **abandonFailed** and **referral** errors are not considered because not supported by CS-1 (assumption 1 and assumption 7).

2.2.2.1.7.1 Attribute error

All the attribute problems cited in the **attributeError** can be used in CS-1 and therefore this error should not be modified and directly taken from the Directory.

2.2.2.1.7.2 Name error

All the naming problems cited in the nameError can be used in CS-1 and therefore this error should be modified and directly taken from the Directory.

2.2.2.1.7.3 Security Error

```
in-SecurityError ERROR ::= {
        PARAMETER       IN-SecurityErrorParameter
        CODE            id-errcode-in-securityError}

IN-SecurityErrorParameter ::= SecurityErrorParameter
        (WITH COMPONENTS  {
            problem     (SecurityProblem (1 | 2 | 3 | 6))})
```

The **securityError** reports a security problem that was encountered during the execution of an operation.

It contains only the **problem** parameter that indicates which type of problem was met. Among the possible problems, the **invalidSignature** and **protectionRequired** ones should not be sent, because they are used in conjunction with signed operations (assumption 4).

2.2.2.1.7.4 Service error

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

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

The **serviceError** reports a problem related to the provision of the service.

It contains only the **problem** parameter that indicates which type of problem was encountered. Among the possible problems, several should not be sent because they are linked to distributed operation (assumption 2), to the incompatibility of versions (assumption 1) and to the use of paged results. That are the **chainingRequired**, **invalidReference**, **timeLimitExceeded**, **outOfScope** and **invalidQueryReference** problems.

2.2.2.1.7.5 Update error

```
in-UpdateError ERROR ::= {
        PARAMETER       IN-UpdateErrorParameter
        CODE            id-errcode-in-updateError}

IN-UpdateErrorParameter::= UpdateErrorParameter
        (WITH COMPONENTS  {
            problem     (UpdateProblem (1 | 2 | 3 | 4 | 5 | 7))})
```

The UpdateError reports problems related to attempts to add, delete, or modify in formation in the DIB. It contains only the **problem** parameter that indicates which type of problem was encountered. Among the possible problems, the **affectsMultipleDSAs** problem should not be sent because it is linked to distributed operation (assumption 2).

2.2.2.2 Directory Access Protocol Subset

2.2.2.2.1 Protocol overview

2.2.2.2.1.1 Remote Operations – Specification and SS No. 7 realization

ITU-T Rec. X.880 | ISO/IEC 13712-1 defines several information object classes that are useful in the specification of ROS-based application protocols such as the various Directory protocols defined in this Directory Specification. A number of these classes are used in this and subsequent subclauses. The specification techniques provided in ITU-T Rec. X.880 | ISO/IEC 13712-1 are used to define a generic protocol between objects. When realized as an SS No. 7 application layer protocol, the concepts of ITU-T Rec. X.880 | ISO/IEC 13712-1 are mapped to SS No. 7 concepts in Recommendation Q.775. The SS No. 7 application layer protocol defined in this Recommendation, is a protocol to provide communication between a pair of application processes. In the SS No. 7 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. All the services provided by the Directory ASEs are contained in a single AE. The Component Handler (CHA) of the Transaction Capabilities (TC) supports the request/reply paradigm of the operation. The Directory ASEs provide the mapping function of the abstract-syntax notation of the directory operation packages onto the services provided by TC. The Dialogue handler (DHA) of the TC supports the establishment and release of an application – association called "dialogue" between a pair of AEs. Dialogues between a DUA and a DSA may be established only by the DUA.

2.2.2.2.1.2 Directory ROS-Objects and Contracts

ITU-T Rec. X.511 | ISO/IEC 9594-3 defines the abstract service between a DUA and the Directory which provides an access point to support a user accessing Directory services. Subclause 2.2.2 defines the sub-set of this abstract service used in the context of Intelligent Networks.

The **dua** class of ROS-object describes a DUA, being an instance of this class, as the initiator of the contract **dapContract**. This contract is referred to in these Directory Specifications as the Directory Abstract Service. It is specified as a ROS-based information object in 2.2.2.2.1.3.

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

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

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

The Directory is further modeled as being represented to a DUA by a DSA which support the particular access point concerned. In the context of Intelligent Networks, each DSA is potentially an access point to the Directory.

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

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

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

2.2.2.2.1.3 DAP Contract and Packages

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

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

When a DUA and DSA from different IN physical entities, this association contract shall be realized as an SS No. 7 application layer protocol, referred to as the IN Directory Access Protocol (DAP). The definition of this protocol in terms of an SS No. 7 application context is provided in 7.2.

The **dapContract** is composed of a **connection package**, **dapConnectionPackage**, and two operation packages, **searchPackage** and **modifyPackage**. The **connection package**, **dapConnectionPackage**, is defined as an information object of class CONNECTION-PACKAGE. The bind and unbind operations of this connection package, **directoryBind** and **directoryUnbind**, are defined in ITU-T Rec. X.511 | ISO/IEC 9594-3.

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

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

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

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

Since the DUA is the initiator of the **dapContract**, it assumes the role of consumer of the operation packages of the contract. This means that only the DUA can invoke operations in this contract and its SS No. 7 realization.

2.2.2.2.1.4 Use of underlying services

The DAP protocol makes use of underlying services as described below.

2.2.2.2.1.4.1 Use of component handling services

The Directory ASEs are users of the TC component handling services except for the TC-L-REJECT and TC-L-CANCEL services which are used by the application-process. Receipt of a TC-L-REJECT-Ind or TC-L-CANCEL-Ind, leads the application-process to abandon the dialogue (i.e. it issues a TC-U-ABORT-Request primitive).

The TC-U-CANCEL service is never used.

2.2.2.2.1.4.2 Use of dialogue handling services

Dialogue handling services are used to support the DirectoryBind and DirectoryUnbind operations and to trigger the sending of the APDUs associated with the operations involved in the Directory packages.

Component grouping is performed under the control of the application-process through an appropriate usage of the TC-BEGIN and TC-CONTINUE service.

The TC-END service is solely used to support the unbind procedure (i.e. it is never used to trigger the sending of components).

On receipt of an empty TC-CONTINUE.req primitive, the SDF should ignore the primitive.

On receipt of a TC-END.req with a database request, the SDF should not perform the database request and consider the requested TC-END service as an unbind procedure. The dialogue is then terminated (see 2.2.2.2.3.1.2).

On receipt of a TC-U-REJECT.ind in the SDF, this primitive should be ignored.

On receipt of a TC-R-REJECT.ind in the SDF, the dialogue should be released with a TC-U-ABORT.req.

If reject situations are detected in the SDF, a TC-U-REJECT.req should be sent followed by a TC-CONTINUE.req.

The pre-arranged termination procedure is never used.

The application-process is the sole user of the TC-P-ABORT service and TC-NOTICE service.

The receipt of a TC-U-ABORT-Ind or TC-P-ABORT-Ind on a dialogue terminates all request processing. It is the application-process responsibility to confirm if requested modifications occurred.

It is an application-process responsibility to provide in the TC-BEGIN-req primitive a destination address which can be used by the underlying SCCP to route the message to the proper SDF.

2.2.2.2.2 Directory protocol abstract syntax

2.2.2.2.2.1 Abstract syntaxes

This version of the Directory Access Protocol requires the support of three abstract syntaxes:

a) the abstract-syntax of TC dialogue control protocol data units, **dialogue-abstract-syntax**, which is needed to establish the dialogues between the SCFs and the SDFs and is specified in Recommendation Q.773;

b) the abstract-syntax for conveying the protocol data units for invoking **directoryBind** and **directoryUnbind** operations and reporting their outcome;

c) the abstract-syntax for conveying the protocol data units for invoking the operations involved in the operation packages specified in 2.2.2.2.1.3 and reporting their outcome.

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

The ASN.1 type from which the values of the last abstract syntax are derived is specified using the parameterized types **TCAPMessage {}** defined in Recommendation Q.773.

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

2.2.2.2.2.1.1 DAP Abstract Syntax

The Directory ASEs that realize the operation packages specified in 2.2.2.2.1.3 share a single abstract syntax, **directoryOperationsAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectoryOperationsAbstractSyntax     ABSTRACT-SYNTAX ::= {
        BasicDAP-PDUs
        IDENTIFIED BY                   id-as-directoryOperationsAS}

BasicDAP-PDUs ::= TCAPMessage {{DAP-Invokable},{DAP-Returnable}}

DAP-Invokable       OPERATION       ::= {search | addEntry | removeEntry | modifyEntry}

DAP-Returnable      OPERATION       ::= {search | addEntry | removeEntry | modifyEntry}
```

The realization of the connection package specified in 2.2.2.2.1.3 uses a separate abstract syntax, **directoryBindingAbstractSyntax**. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
InDirectoryBindingAbstractSyntax        ABSTRACT-SYNTAX  ::= {
        DAPBinding-PDUs
        IDENTIFIED BY                       id-as-directoryBindingAS}

DAPBinding-PDUs ::= CHOICE {
        bind        Bind {directoryBind},
        unbind      Unbind {directoryUnbind}}
```

2.2.2.2.2 Directory application contexts

2.2.2.2.2.1 Directory Access Application Context

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

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

2.2.2.2.2.3 Operation Codes

The operations involved in the packages defined in this Recommendation are specified in Recommendation X.519 where the assigned operation codes are imported from Recommendation X.519.

2.2.2.2.2.4 Error Codes

The errors involved in the packages defined in this Recommendation are specified in Recommendation X.519 where the assigned error codes are imported from Recommendation X.519.

2.2.2.2.2.5 Versions and the rules for extensibility

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

2.2.2.2.2.5.1 Version negotiation

When accepting an association, i.e. binding, utilizing the DAP, the version negotiated shall only affect the point-to-point aspects of the protocol exchanged between the DUA and the DSA to which it is connected. Subsequent requests or responses on the association shall not be constrained by the version negotiated.

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

2.2.2.2.2.5.2 DUA side

2.2.2.2.2.5.2.1 Request and response processing at the DUA side

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

NOTE 1 – If the information, the extension replaced in a CHOICE, ENUMERATED or INTEGER (used as ENUMERATED) type would be essential for proper operation in a DSA implemented according to an earlier edition of this Specification, it is recommended that the extension be marked critical.

When processing a response, a DUA shall:

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

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

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

NOTE 2 – Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

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

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

2.2.2.2.2.5.2.2 Extensibility rules for error handling at the DUA side

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

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

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

When processing an unknown error type, a DUA shall:

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

b) optionally report the error to the user.

2.2.2.2.2.5.3 Request processing at the DSA side

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

NOTE 1 – If a **criticalExtensions** string with one or more zero values is received, this indicates either that the extensions corresponding to the values are not present or are not critical. The presence of a zero value in a **criticalExtensions** string shall not be inferred as either the presence or absence of the corresponding extension in the APDU.

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

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

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

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

NOTE 2 – Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

2.2.2.2.3 Mapping onto used services

This subclause defines the mapping of the DAP onto TC services.

2.2.2.2.3.1 Mapping onto dialogue services

This subclause defines the mapping of the DirectoryBind and DirectoryUnbind services onto the services of the TC dialogue handling services defined in Recommendation Q.771.

2.2.2.2.3.1.1 Bind

The DirectoryBind service is mapped onto TC-services as follows:

a) The TC-BEGIN service is used to invoke the **DirectoryBind** operation.

b) The TC-CONTINUE service is used to report the success of the **DirectoryBind** operation.

c) The TC-U-ABORT service is used to report the failure of the **DirectoryBind** operation.

The use of the parameter of these services is qualified in the following subclauses.

2.2.2.2.3.1.1.1 Mapping onto TC-BEGIN

2.2.2.2.3.1.1.1.1 Quality of Service

This parameter shall be used as specified in Recommendation Q.771. No specific constraints are imposed by this Recommendation.

2.2.2.2.3.1.1.1.2 Destination Address

This parameter shall contain the address of the SDF which plays the role of the DSA.

2.2.2.2.3.1.1.1.3 Application-Context-Name

This parameter shall take the value of the application-context-name field of the **iNdirectoryAccessAC** object.

2.2.2.2.3.1.1.1.4 Originating Address

This parameter shall contain the address of the SCF in which the DUA resides.

2.2.2.2.3.1.1.1.5 Dialogue Id

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.1.1.6 User Information

This parameter shall contain a value of type **DirectoryBindArgument**.

2.2.2.2.3.1.1.1.7 Component Present

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.1.2 Mapping onto TC-CONTINUE

2.2.2.2.3.1.1.2.1 Quality of Service

This parameter shall be used as specified in Recommendation Q.771. No specific constraints are imposed by this Recommendation.

2.2.2.2.3.1.1.2.2 Originating Address

This parameter shall be used as specified in Recommendation Q.771. If, present it shall contain the address of the SDF which plays the role of a DSA.

2.2.2.2.3.1.1.2.3 Application-Context-Name

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.1.2.4 Dialogue Id

This parameter shall be used as specified in Recommendation Q.771.

The Authorized Relationship Id can be mapped onto the TCAP Dialogue Id.

2.2.2.2.3.1.1.2.5 User Information

This parameter shall contain a value of type **DirectoryBindResult**.

2.2.2.2.3.1.1.2.6 Component Present

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.1.3 Mapping onto TC-U-ABORT

2.2.2.2.3.1.1.3.1 Quality of Service

This parameter shall be used as specified in Recommendation Q.771. No specific constraints are imposed by this Recommendation.

2.2.2.2.3.1.1.3.2 Dialogue Id

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.1.3.3 Abort Reason

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.1.3.4 Application-Context-Name

This parameter shall be used as specified in Recommendation Q.771. When the SDF refuses a dialogue because the application-context-name it receives is not supported, this parameter shall have the value of the &application-context-name field of the **iNdirectoryAccessAC** object.

2.2.2.2.3.1.1.3.5 User information

When the abort reason parameter has the value "dialogue-refused", this parameter shall contain a value of type **DirectoryBindError**. Otherwise it shall be absent.

2.2.2.2.3.1.2 Unbind

The **DirectoryUnbind** service is mapped onto the TC-END service. The use of the parameter of the TC-END service is qualified in the following subclauses.

2.2.2.2.3.1.2.1 Quality of Service

This parameter shall be used as specified in Recommendation Q.771. No specific constraints are imposed by this Recommendation.

2.2.2.2.3.1.2.2 Application-Context-Name

This parameter is not used in this phase of a dialogue.

2.2.2.2.3.1.2.3 Dialogue Id

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.1.2.4 User Information

This parameter shall be empty.

2.2.2.2.3.1.2.5 Component Present

This parameter shall be used as specified in Recommendation Q.771.

2.2.2.2.3.2 Mapping onto component handling services

The Directory ASE services are mapped onto the TC component handling services. The mapping of operations and errors onto TC services is defined in Recommendation Q.774.

The timeout parameter of the TC-INVOKE-Req primitives is set according to Table 3.

TABLE 3/Q.1218

TC timer values of DAP operations

Operation	Timeout
search	ffs
modifyEntry	ffs
addEntry	ffs
removeEntry	ffs

2.2.2.2.4 Conformance

This subclause defines the requirements for conformance to this Specification.

2.2.2.2.4.1 Conformance by SCFs

An SCF implementation claiming conformance to this Specification shall satisfy the requirements specified in 2.2.2.2.4.1.1 through 2.2.2.2.4.1.3.

2.2.2.2.4.1.1 Statement requirements

The following shall be stated:

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

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

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

2.2.2.2.4.1.2 Static requirements

An SCF shall:

a) have the capability of supporting the **iNdirectoryAccessAC** application-context as defined by its abstract syntax in 2.2.2.2.2;

b) conform to the extensions for which conformance was claimed in 2.2.2.2.4.1.1 c).

2.2.2.2.4.1.3 Dynamic requirements

An SCF shall:

a) conform to the mapping onto used services defined in 2.2.2.2.3;

b) shall conform to the rules of extensibility procedures defined in 2.2.2.2.2.5.2.

2.2.2.2.4.2 Conformance by SDFs

An SDF implementation claiming conformance to this Specification shall satisfy the requirements specified in 2.2.2.2.4.2.1 through 2.2.2.2.4.2.3.

2.2.2.2.4.2.1 Statement requirements

The following shall be stated:

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

NOTE – An application context shall not be divided except as stated herein; in particular, conformance shall not be claimed to particular operations.

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

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

d) The object classes, for which conformance is claimed.

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

f) Whether conformance is claimed for collective attributes as defined in 8.8 of ITU-T Rec. X.501 | ISO/IEC 9594-2 and 7.6, 7.8.2 and 9.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

g) Whether conformance is claimed for hierarchical attributes as defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

h) The operational attribute types defined in ITU-T Rec. X.501 | ISO/IEC 9594-2 and any other operational attribute types for which conformance is claimed.

i) Whether conformance is claimed for return of alias names as described in 7.7.1 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

j) Whether conformance is claimed for indicating that returned entry information is complete, as described in 7.7.6 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

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

l) Whether conformance is claimed to Basic Access Control.

m) Whether conformance is claimed to Simplified Access Control.

n) The name bindings for which conformance is claimed.

o) Whether the SDF is capable of administering collective attributes, as defined in ITU-T Rec. X.501 | ISO/IEC 9594-2

p) Whether conformance is claimed for contexts.

2.2.2.2.4.2.2 Static requirements

An SDF shall:

a) have the capability of supporting the application-contexts for which conformance is claimed as defined by their abstract syntax in 2.2.2.2.2;

b) have the capability of supporting the information framework defined by its abstract syntax in ITU-T Rec. X.501 | ISO/IEC 9594-2;

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

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

e) conform to the extensions for which conformance was claimed in 2.2.2.2.4.2.1;

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

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

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

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

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

2.2.2.2.4.2.3 Dynamic requirements

An SDF shall:

a) conform to the mapping onto used services defined in 2.2.2.2.3;

b) conform to the rules of extensibility procedures defined in 2.2.2.2.2.5.3;

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

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

2.2.2.3 X.501 profile

The Recommendation X.501 provides a generic information model that is needed to support the service provided by the Directory. In the context of IN, the generic information model should conform to the part of the Recommendation going from clauses 1 to 7 of the Recommendation X.501. However certain aspects of this Recommendation need not to be supported. This includes the DIT content rules whose use is a local matter.

Some other points are outside the scope of this Recommendation. This concerns the distributed procedures, the replications procedures and other items associated with capabilities not covered by IN CS-1. Therefore the following parts of Rec. X.501 are not applicable:

– Subclause 3.3 Distributed operation definitions;

– Subclause 3.4 Replication definitions;

– last paragraph of 6.3.2;

– paragraphs f), h) and i) in 16.2.3;

– paragraph a) in 16.2.4. The compare operation is not used, the search operation is used instead. Therefore the FilterMatch permission replaces the Compare permission.

In this subclause when it is referred to Recommendation X.501, it refers to the modified version of the Recommendation X.501 including the amendments made to it (DAM2 on contexts and DAM3 on minor extensions).

2.2.2.4 ASN.1 Profile of the Directory Abstract Service for the IN CS-1

2.2.2.4.1 ASN.1 Module for IN X.500 profile

This subclause contains the full ASN.1 profiling of the Directory Abstract Service. It has been successfully compiled.

```
IN-DirectoryAbstractService {ccitt recommendation q 1218 modules (0) abstractService (15) version 1 (0)}

DEFINITIONS ::=

BEGIN

IMPORTS
attributeError, ServiceControls, EntryInformation, EntryInformationSelection,  DirectoryBindArgument, Credentials,
SearchArgument, SearchResult, SearchInfo, PartialOutcomeQualifier, AddEntryArgument, RemoveEntryArgument,
ModifyEntryArgument, ServiceProblem, UpdateProblem, SecurityProblem, directoryBindError, AddEntryResult,
BindErrorParameter, ModifyEntryResult, RemoveEntryResult, SecurityErrorParameter, ServiceErrorParameter,
AttributeErrorParameter, UpdateErrorParameter,
        FROM DirectoryAbstractService {joint-iso-ccitt ds (5) module (1) directoryAbstractService (2) 2}

Code, OPERATION, ERROR
        FROM Remote-Operations-Information-Objects {joint-iso-ccitt remote-operations (4) informationObjects (5)
version1 (0)};

IN-ServiceControls ::= ServiceControls
        (WITH COMPONENTS {
                options                 ({chainingProhibited, localScope}|
                                        {chainingProhibited, localScope, dontDereferenceAliases, subentries}|
                                        {chainingProhibited, localScope, dontDereferenceAliases}|
                                        {chainingProhibited, localScope, subentries}...),
                priority                ,
                timeLimit               ABSENT,
                sizeLimit               ABSENT,
                scopeOfReferral         ABSENT,
                attributeSizeLimit      ABSENT},
                ...)

IN-EntryInformationSelection ::= EntryInformationSelection
        (WITH COMPONENTS {
                attributes              ,
                infoTypes               (attributeTypesAndValues ....),
                extraAttributes         OPTIONAL,
                contextSelection        OPTIONAL,
                returnContexts          },
                ...)

IN-EntryInformation ::= EntryInformation
        (WITH COMPONENTS {
                name                    PRESENT,
                fromEntry               (TRUE ....),
                information             (WITH COMPONENTS {
                                attributeType   ABSENT,
                                attribute       PRESENT},
                                ...) OPTIONAL,
                incompleteEntry         ,
                partialNameResolution   (FALSE ...)},
                ...)

UNSIGNED {Type} ::= OPTIONALLY-SIGNED {Type}
        (WITH COMPONENTS {
                unsigned        PRESENT,
                signed          ABSENT} ...)

in-DirectoryBind OPERATION ::= {
        ARGUMENT    IN-DirectoryBindArgument
        RESULT      IN-DirectoryBindResult
        ERRORS      {in-DirectoryBindError}}

IN-DirectoryBindArgument ::= DirectoryBindArgument
        (WITH COMPONENTS {
                credentials     (IN-Credentials ....) OPTIONAL,
                versions        } ....)

IN-Credentials ::= Credentials
        (WITH COMPONENTS {
                simple                  ,
                strong                  ABSENT,
                externalProcedure       } ....)
```

```
IN-DirectoryBindResult ::= IN-DirectoryBindArgument

in-DirectoryBindError ERROR ::= {
        PARAMETER     IN-DirectoryBindErrorParameter}

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

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

IN-SearchArgument ::= UNSIGNED {SearchArgument
        (WITH COMPONENTS {
                baseObject              PRESENT,
                subset                  ,
                filter                  ,
                searchAliases           (TRUE ....),
                selection               (IN-EntryInformationSelection ....),
                pagedResults            ABSENT,
                matchedValuesOnly       ,
                extendedFilter          ABSENT
                serviceControls         (IN-ServiceControls ....),
                securityParameters      OPTIONAL,
                requestor               ABSENT,
                operationProgress       ({nameResolutionPhase notStarted} ....),
                aliasedRDNs             ABSENT,
                criticalExtensions      OPTIONAL,
                referenceType           ABSENT,
                entryOnly               (TRUE ....),
                exclusions              ABSENT,
                nameResolveOnMaster     (FALSE ....),
                checkOverespecified     },
                ...)}

IN-SearchResult ::= UNSIGNED {SearchResult
        (WITH COMPONENTS {
                searchInfo      (WITH COMPONENTS {
                        name                    ,
                        entries                 (WITH COMPONENT (IN-EntryInformation ....) ....),
                        partialOutcomeQualifier PartialOutcomeQualifier
                                (WITH COMPONENTS {
                                        limitProblem                    OPTIONAL,
                                        unexplored                      ABSENT
                                        unavailableCriticalExtensions   ,
                                        unknownErrors                   ABSENT,
                                        queryReference                  ABSENT,
                                        overspectFilter                 OPTIONAL},
                                        ...) ....),
                        securityParameters      ABSENT,
                        performer               ABSENT,
                        aliasDereferenced       },
                        ...),
                uncorrelatedSearchInfo  ABSENT},
                ...)}

in-AddEntry OPERATION ::= {
        ARGUMENT    IN-AddEntryArgument
        RESULT      AddEntryResult
        ERRORS      {nameError | in-ServiceError | in-SecurityError | attributeError | in-UpdateError}
        CODE        id-opcode-in-addEntry}
```

```
IN-AddEntryArgument ::= UNSIGNED {AddEntryArgument
        (WITH COMPONENTS {
                object                  PRESENT,
                entry                   PRESENT,
                targetSystem            ABSENT,
                serviceControls         (IN-ServiceControls ....),
                securityParameters      OPTIONAL,
                requestor               ABSENT,
                operationProgress       ({nameResolutionPhase notStarted} ....),
                aliasedRDNs             ABSENT,
                criticalExtensions      OPTIONAL,
                referenceType           ABSENT,
                entryOnly               (TRUE ....),
                exclusions              ABSENT,
                nameResolveOnMaster     (FALSE ....)},
                ...)}

in-RemoveEntry OPERATION ::= {
        ARGUMENT    IN-RemoveEntryArgument
        RESULT      RemoveEntryResult
        ERRORS      {nameError | in-ServiceError | in-SecurityError  | in-UpdateError}
        CODE        id-opcode-in-removeEntry}

IN-RemoveEntryArgument ::= UNSIGNED {RemoveEntryArgument
        (WITH COMPONENTS {
                object                  PRESENT,
                serviceControls         (IN-ServiceControls ....),
                securityParameters      OPTIONAL,
                requestor               ABSENT,
                operationProgress       ({nameResolutionPhase notStarted} ....),
                aliasedRDNs             ABSENT,
                criticalExtensions      OPTIONAL,
                referenceType           ABSENT,
                entryOnly               (TRUE ....),
                exclusions              ABSENT,
                nameResolveOnMaster     (FALSE ....)},
                ...)}

in-ModifyEntry OPERATION ::= {
        ARGUMENT    IN-ModifyEntryArgument
        RESULT      ModifyEntryResult
        ERRORS      {nameError | in-ServiceError | in-SecurityError | attributeError | in-UpdateError}
        CODE        id-opcode-in-modifyEntry}

IN-ModifyEntryArgument ::= UNSIGNED {ModifyEntryArgument
                (WITH COMPONENTS {
                object                  PRESENT,
                changes                 PRESENT,
                selection               (IN-EntryInformationSelection ....),
                serviceControls         (IN-ServiceControls ....),
                securityParameters      OPTIONAL,
                requestor               ABSENT,
                operationProgress       ({nameResolutionPhase notStarted} ....),
                aliasedRDNs             ABSENT,
                criticalExtensions      OPTIONAL,
                referenceType           ABSENT,
                entryOnly               (TRUE ....),
                exclusions              ABSENT,
                nameResolveOnMaster     (FALSE ....)},
                ...)}

IN-ModifyEntryResult ::= ModifyEntryResult
        (WITH COMPONENTS {
                null                ,
                information         UNSIGNED {Information
                    (WITH COMPONENTS {
```

```
                    entry                   (IN-EntryInformation ....),
                    securityParameters      OPTIONAL,
                    performer               ABSENT,
                    aliasDereferenced       },
                    ...)}};
                    ...)

in-SecurityError ERROR ::= {
        PARAMETER    IN-SecurityErrorParameter
        CODE         id-errcode-in-securityError}

IN-SecurityErrorParameter ::= SecurityErrorParameter
                (WITH COMPONENTS {
                problem                 (SecurityProblem (1 | 2 | 3 | 6|7 ....) ....)} ....)

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

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

in-UpdateError ERROR ::= {
        PARAMETER    IN-UpdateErrorParameter
        CODE         id-errcode-in-updateError}

IN-UpdateErrorParameter::= UpdateErrorParameter
        (WITH COMPONENTS {
                problem                 (UpdateProblem (1|2|3|4|5|7 ....) ....)} ....)

id-errcode-in-serviceError      Code ::= local:2
id-errcode-in-securityError     Code ::= local:3
id-errcode-in-updateError       Code ::= local:4

id-opcode-in-modifyEntry        Code ::=local:1
id-opcode-in-addEntry           Code ::=local:2
id-opcode-in-removeEntry        Code ::=local:3
id-opcode-in-search             Code ::=local:4

END
```

2.2.2.4.2 ASN.1 Modules for the DAP

This subclause includes all of the ASN.1 type and value definitions contained in this Directory Specification, in the form of the ASN.1 module, "DirectoryAccessProtocol". It also includes all of the ASN.1 Object Identifiers assigned in this Specification, in the form of a ASN.1 module, "ProtocolObjectIdentifiers".

```
INDirectoryAccessProtocol {ccitt recommendation q 1218 module(0) indap(12) version1( 0)}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        directoryAbstractService
                FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}

        ROS-OBJECT-CLASS, CONTRACT, OPERATION-PACKAGE, CONNECTION-PACKAGE,
        Code, OPERATION
                FROM Remote-Operations-Information-Objects
                        {joint-iso-ccitt remote-operations(4) informationObjects(5) version1(0)}

        Bind{}, Unbind{}, InvokeID
                FROM Remote-Operations-Generic-ROS-PDUs
                        {joint-iso-ccitt remote-operations(4) generic-ROS-PDUs(6) version1(0)}
```

```
        TCAPMessage {}
                FROM TCAPMessages
                        {ccitt recommendation q 773 modules(2) messages(1) version3(3)}

        APPLICATION-CONTEXT
                FROM TC-Notation-Extension
                        {ccitt recommendation q 775 modules(2) notation-extensions (4) version1(1)}

        dialogue-as-id
                FROM DialoguePDUs
                        {ccitt recommendation q 773 modules(2) dialoguePDUs (2) version1(1)}

        id-ac-directoryAccessAC, id-rosObject-dua, id-rosObject-directory, id-rosObject-dapDSA,
        id-contract-dap, id-package-dapConnection, id-package-search, id-package-modify,
        id-as-directoryOperationsAS, id-as-directoryBindingAS
                FROM SDFProtocolObjectIdentifiers
                        {ccitt recommendation q 1218 modules(0) sdfProtocolObjectIdentifiers (10) version1 (0)}

        directoryUnbind FROM DirectoryAbstractService directoryAbstractService ;

        directoryBind, search, addEntry, removeEntry, modifyEntry
                FROM ExtendedDirectoryAbstractService
                        {ccitt recommendation q 1218 modules(0) informationFramework (11) version1( 0)}

-- application contexts --

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

-- ROS-objects --

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

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


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

-- contracts --

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

```
-- connection package --

dapConnectionPackage        CONNECTION-PACKAGE ::=
{
        BIND                                    directoryBind
        UNBIND                                  directoryUnbind
        ID                                      id-package-dapConnection
}

-- search package --

searchPackage               OPERATION-PACKAGE  ::=
{
        CONSUMER INVOKES                {search }
        ID                              id-package-search
}

-- modify package --

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

-- abstract-syntaxes --

inDirectoryOperationsAbstractSyntax   ABSTRACT-SYNTAX ::=    {
        BasicDAP-PDUs
        IDENTIFIED BY                   id-as-directoryOperationsAS }

BasicDAP-PDUs ::= TCAPMessage {{ DAP-Invokable }, { DAP-Returnable } }

DAP-Invokable      OPERATION      ::=      {search | addEntry | removeEntry | modifyEntry}

DAP-Returnable     OPERATION      ::=      {search | addEntry | removeEntry | modifyEntry}

inDirectoryBindingAbstractSyntax      ABSTRACT-SYNTAX ::=    {
        DAPBinding-PDUs
        IDENTIFIED BY                   id-as-directoryBindingAS }

DAPBinding-PDUs ::= CHOICE
{
        bind                            Bind { directoryBind },
        unbind                          Unbind { directoryUnbind }
}

END

SDFProtocolObjectIdentifiers
        {ccitt recommendation q 1218 module(0) sdfProtocolObjectIdentifiers(10) version1 (0)}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

IMPORTS
-- useful definitions

in-ds                           OBJECT IDENTIFIER ::=
                {ccitt recommendation q 1218 sdf-objects (10)}

id-rosObject                    OBJECT IDENTIFIER       ::=     {in-ds 25}
id-contract                     OBJECT IDENTIFIER       ::=     {in-ds 26}
id-package                      OBJECT IDENTIFIER       ::=     {in-ds 27}
id-ac                           OBJECT-IDENTIFIER       ::=     {in-ds 3}
id-as                           OBJECT IDENTIFIER       ::=     {in-ds 5}
```

```
-- ROS Objects --

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

-- contracts --

id-contract-dap             OBJECT IDENTIFIER   ::=   {id-contract 1}

-- packages --

id-package-search           OBJECT IDENTIFIER   ::=   {id-package 2}
id-package-modify           OBJECT IDENTIFIER   ::=   {id-package 3}
id-package-dapConnection    OBJECT IDENTIFIER   ::=   {id-package 10}

-- application contexts --

id-ac-directoryAccessAC     OBJECT IDENTIFIER   ::=   {id-ac 1}

-- abstract syntaxes --

id-as-directoryOperationsAS OBJECT IDENTIFIER   ::=   {id-as 1}
id-as-directoryBindingAS    OBJECT IDENTIFIER   ::=   {id-as 2}

END
```

3 Semantics

3.1 Definition of procedures and entities

3.1.1 SSF application entity procedures

3.1.1.1 General

This clause provides the definition of the SSF Application Entity (AE) procedures related to the SSP-SCP interface. The procedures are based on the use of Common Channel Signalling System No. 7 (CCSS No. 7); other signalling systems can be used (e.g. DSS 1-layer 3).

Capabilities not explicitly covered by these procedures may be supported in an implementation dependent manner in the SSP, while remaining in line with clause 2.

The AE, following the architecture defined in the Recommendations Q.700, Q.771 and Q.1400, includes Transaction Capabilities Application Part (TCAP) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE which interfaces with TCAP using the primitives specified in Recommendation Q.771; other signalling systems, such as DSS 1 (layer 3), may be used.

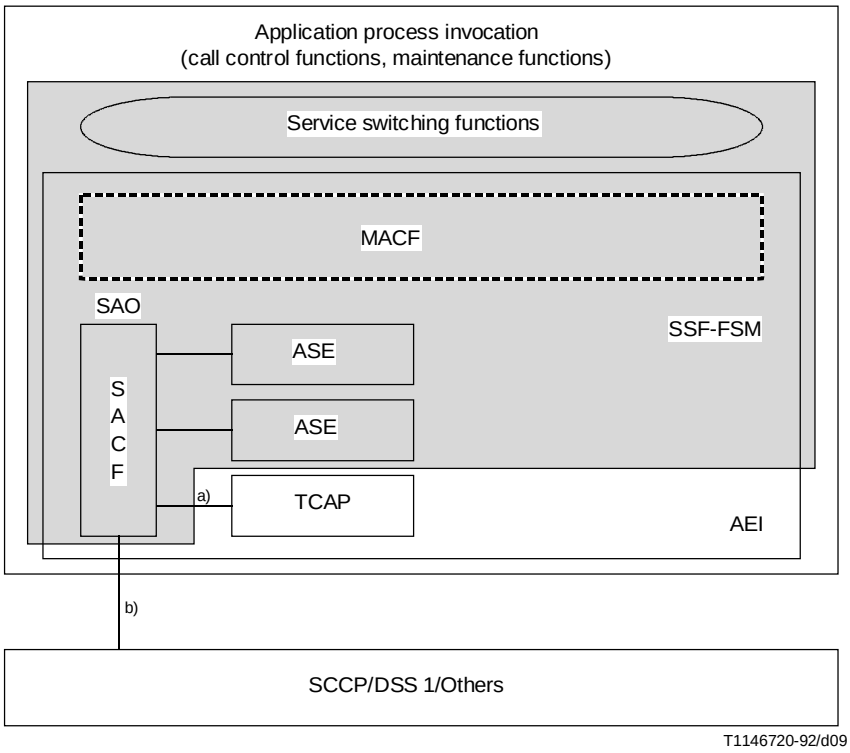
The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed subclauses 3.3 and 3.4 shall be followed.

3.1.1.2 Model and interfaces

The functional model of the AE-SSF is shown in Figure 9; the ASEs interface to TCAP to communicate with the SCF, and interface to the call control function (CCF) and the maintenance functions already defined for switching systems. The scope of this Recommendation is limited to the shaded area in Figure 9.

The interfaces shown in Figure 9 use the TC-user ASE primitives specified in Recommendation Q.771 and N-Primitives specified in Recommendation Q.711. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clause 2.



AEI Application Entity Invocation
SSF Service Switching Functions
FSM Finite State Machine
MACF Multiple Association Control Function
SACF Single Association Control Function
SAO Single Association Object

a) TC-Primitives
b) N-Primitives

NOTE – SSF-FSM includes several finite state machines.

FIGURE 9/Q.1218

Functional model of SSF-AE

3.1.1.3 Relations between SSF-FSM and the CCF and maintenance functions

The primitive interface between the SSF-FSM and the CCF/maintenance functions is an internal interface and is not subject to standardization in CS-1. Nevertheless this interface should be in line with the BCSM defined in 4.2.1.2/Q.1214.

The relationship between the BCSM and the SSF-FSM may be described as follows for the case of a call/attempt initiated by an end user, and the case of a call/attempt initiated by IN service logic:

– When a call/attempt is initiated by an end user and processed at an exchange, an instance of a BCSM is created. As the BCSM proceeds, it encounters detection points (DPs, see 4.2/Q.1214). If a DP is armed as a Trigger DP (TDP) an instance of an SSF-FSM is created.

– If an InitiateCallAttempt is received from the SCF, an instance of a BCSM is created, as well as an instance of an SSF-FSM.

The SSF logic should:

- perform the DP processing actions specified in 4.2.2.7/Q.1214, including if DP criteria are met;
- check if traffic mechanisms are active;
- check for SCF accessibility;
- handle service feature interactions.

The SSF hands control back to the CCF at least in the following cases:

- If call gapping is in effect – The SSF logic will instruct the CCF to terminate the call with the appropriate treatment.
- If service filtering is in effect – The call is counted (if required) and the SSF logic instructs the CCF to handle the call with the appropriate treatment.
- If a trigger (TDP) criteria match is not found (e.g. insufficient information to proceed) – The SSF logic returns call control to the CCF.
- If the call is abandoned – The SSF logic returns call control to the CCF and continues processing as described in 3.1.1.5.
- If the destination SCF is not accessible – The SSF logic will instruct the CCF to route the call if possible (e.g. default routing to a terminating announcement).
- If there is an existing control relationship for the call and the DP is armed as an EDP-R – The SSF returns call control to the CCF.

The management functions related to the execution of operations received from the SCF are executed by the SSF Management Entity (SSME). The SSME comprises a SSME-Control and several instances of SSME FSMs. The SSME-control interfaces the different SSF FSMs and SSME FSMs respectively and the Functional Entity Access Manager (FEAM). Figure 10 shows the SSF Interfaces.

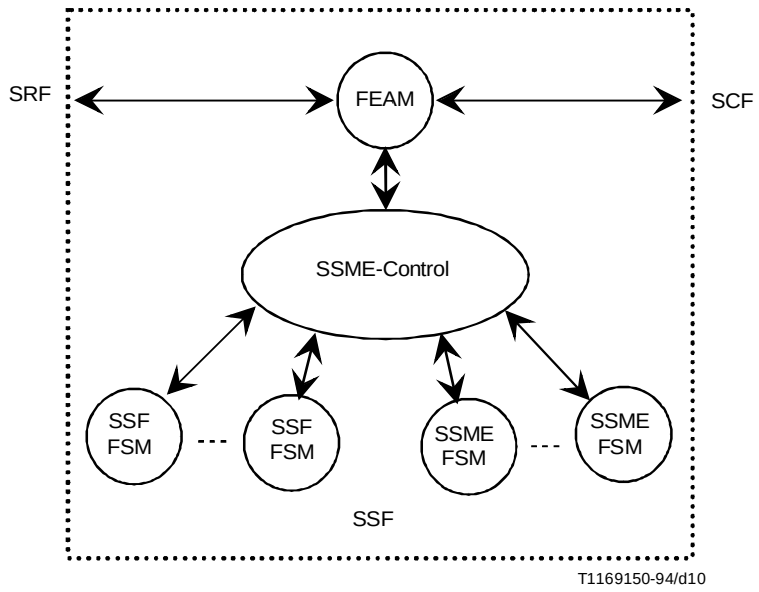


FIGURE 10/Q.1218

SSF interfaces

The Functional Entity Access Manager (FEAM) provides the low level interface maintenance functions including the following:

1) establishing and maintaining the interfaces to the SCF and SRF;

2) passing and queuing (when necessary) the messages received from the SCF and SRF to the SSME-Control;

3) formatting, queuing (when necessary), and sending the messages received from the SSME-Control to the SCF and SRF.

The SSME-control maintains the dialogues with the SCF and SRF on behalf of all instances of the SSF Finite State Machine (FSM). These instances of the SSF FSM occur concurrently and asynchronously as calls occur, which explains the need for a single entity that performs the task of creation, invocation and maintenance of the SSF-FSMs. In particular the SSME-control performs the following tasks:

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

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

3) Captures asynchronous (with call processing) activities related to management or supervisory functions in the SSF and creates an instance of a SSME-FSM. For example, the SSME provides 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.

The different contexts of the SSME-FSMs may be distinguished based on the address information provided in the initiating operations. In the case of service filtering this address information is given by filteringCriteria, i.e. all ActivateServiceFiltering operations using the same address, address the same SSME-FSM handling this specific service filtering instance. For example ActivateServiceFiltering operations providing different filtering Criteria cause the invocation of new SSME-FSMs.

The SSF-FSM passes call handling instructions to the related instances of the BCSM as needed. DPs may be dynamically armed as Event DPs, requiring the SSF-FSM to remain active. At some point, further interaction with the SCF is not needed, and the SSF-FSM may be terminated while the BCSM continues to handle the call as needed. A later TDP in the BCSM may result in a new instance of the SSF-FSM for the same call.

Consistent with the single-ended control characteristic of IN service features for CS-1, the SSF-FSM only applies to a functionally separate call portion (e.g. the originating BCSM or the terminating BCSM in a two-party call, but not both).

3.1.1.4 SSF Management Finite State Machine (SSME-FSM)

The SSME-FSM state diagram is described in Figure 11.

The SSME-FSM is independent of the individual SSF-FSMs.

The non-call associated treatment state is entered from the IdleManagement state when one of the following non-call associated operations is received (transition em1):

– RequestCurrentStatusReport;

– RequestEveryStatusChangeReport;

– RequestFirstStatusMatchReport;

– ActivateServiceFiltering;

– CallGap;

– ActivityTest.

The CallGap operation may be received inside as well as outside a call context transaction. The ActivityTest operation applies to call context transactions only. The ActivateServiceFiltering operation may be received outside a call context only.

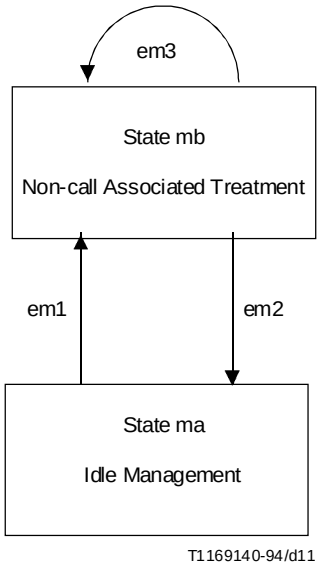


FIGURE 11/Q.1218

SSME-FSM state diagram

During this state the following events can occur:

- given that service filtering is active, the SSF needs to send a service filtering response to the SCF: the SSME-FSM remains in this state (transition em3);

- given that service filtering is active, the SSF needs to increment a counter: the SSME FSM remains in this state (transition em3);

- given that service filtering is active and the service filtering duration expires: the SSME should move to the Idle Management state (transition em2) and send a ServiceFilteringResponse operation to the SCF;

- given that status report is active previously requested by a RequestEveryStatusChangeReport operation, the SSF needs to send a StatusReport operation: the SSF remains in this state (transition em3);

- given that status report is active previously requested by a RequestFirstStatusMatchReport operation, the SSF needs to send a StatusReport operation: the SSF transits to Idle Management state (transition em2);

- given that status report is active previously requested by a RequestFirstStatusMatchReport or a RequestEveryStatusMatchReport operation, and the CancelStatusReportRequest operation is received by the SSF or the status report duration expires, the SSME-FSM should move to the Idle Management state (transition em2);

- given that status report is active previously requested by a RequestCurrentStatusReport operation, the SSF sends the StatusReport operation, the SSME-FSM should move to the Idle Management state (transition em2);

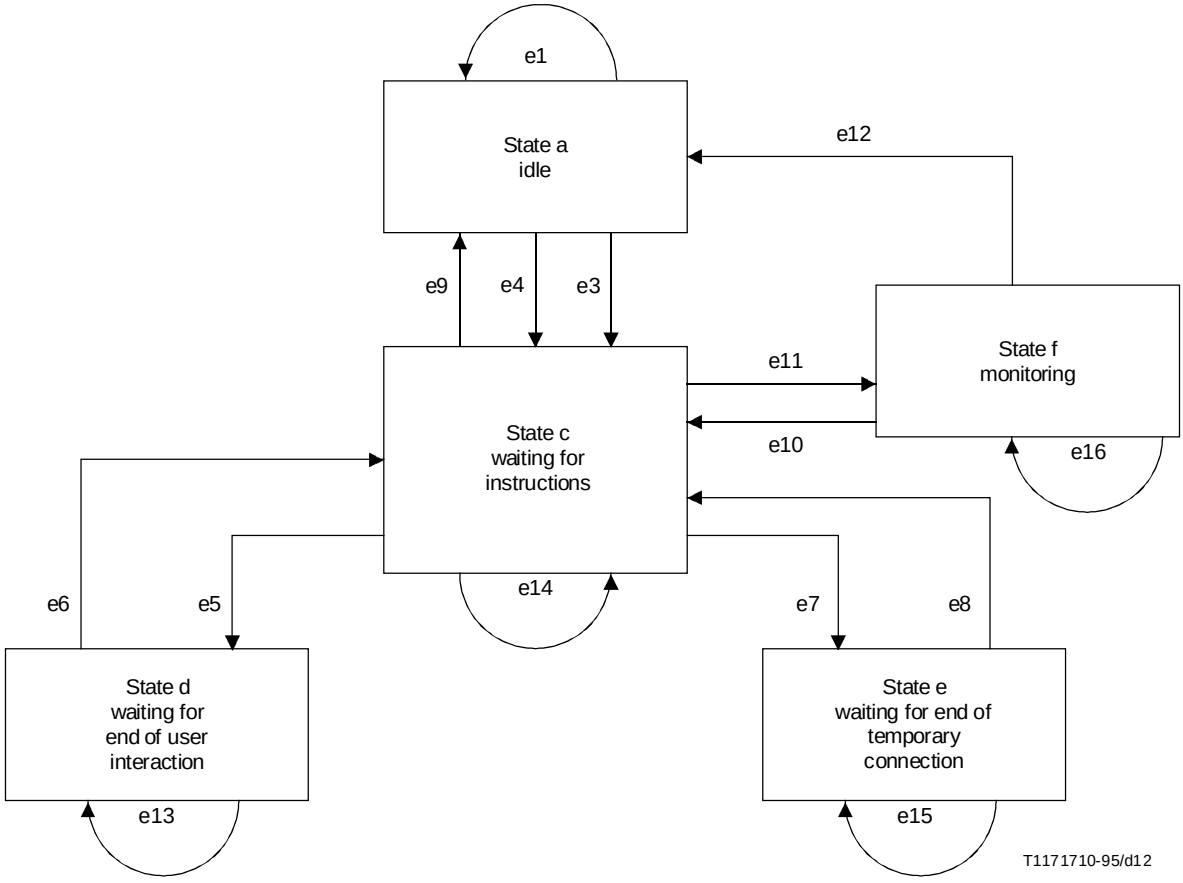
- if call gap related duration timer expires, the SSME-FSM should move to the Idle Management state (transition em2);

- given that call gap/service filtering is active, another CallGap/ActivateServiceFiltering operation could be received by the SSF, which has the same gapping/filtering criteria: the second “filter” or “gap” replace the first one (transition em3) unless the duration timer value is equal to zero, in which case the SSME-FSM should move to the Idle Management state (transition em2).

All other operations have no effect on the SSME-FSMs; the operations are passed by the SSME-Control to the relevant SSF-FSM.

3.1.1.5 SSF state transition diagram

Figure 12 shows the state diagram of the SSF part of the SSP during the processing of an IN call/attempt.



NOTE - The abandon and disconnect transition is not shown.

FIGURE 12/Q.1218

SSF finite state machine

Each state is discussed in the following subclauses. General rules applicable to more than one state are addressed here.

One or a sequence of components received in one or more TCAP messages may include a single operation or multiple operations, and is processed as follows:

- Process the operations in the order in which they are received.

- Each operation causes a state transition independent of whether or not a single operation or multiple operations are received in a message.

– The SSF examines subsequent operations in the sequence. As long as sequential execution of these operations would leave the FSM in the same state, it will execute them (e.g. RequestReportBCSMEvent). If a subsequent operation causes a transition out of the state, then the following operations should be buffered until the current operation has been executed. In all other cases, await an event that would cause a transition out of the current state (such an event would be the completion of operation being executed, or reception of an external event). An example of this is as follows:

  • The SSF receives the operations FurnishChargingInformation, ConnectToResource, and PlayAnnouncement in a component sequence inside a single TCAP message. Upon receipt of this message, these operations are executed up to and including ConnectToResource while the SSF is in the Waiting For Instructions state. As the ConnectToResource operation is executed (and when, or after the FurnishChargingInformation operation has been completed), the SSF-FSM will transit to the Waiting For End Of User Interaction state. The PlayAnnouncement operation is relayed to the SRF while the SSF is in Waiting For End Of User Interaction state.

– If there is an error in processing one of the operations in the sequence, the SSF-FSM processes the error (see below) and discards all remaining operations in the sequence.

– If an operation is not understood or is out of context (i.e. violates the SACF rules defined by the SSF-FSM) as described above, ABORT the interaction. For example, when the SSF-FSM applies to an originating BCSM, then receiving SelectFacility operation would be out of context since this applies only to the terminating half of the BCSM.

In any state, if there is an error in a received operation, the maintenance functions are informed and the SSF-FSM remains in the same state as when it received the erroneous operation; depending on the class of the operation, the error could be reported by the SSF to the SCF using the appropriate component (see Recommendation Q.774).

In any state (except idle), if the calling party abandons the call before it is answered (i.e. before the Active PIC in the BCSM), then the SSF-FSM should instruct the CCF to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows:

– If the Abandon DP is not armed and there is no call information request pending, then transition to the idle state.

– If the Abandon DP is not armed and there is a CallInformationRequest pending, send a CallInformationReport, then transit to the Idle state.

– If the Abandon DP is armed as an EDP-R, send an EventReportBCSM or a DP specific operation, then transit to the Waiting For Instructions state. If a CallInformationRequest is pending, a CallInformationRequest shall be sent before the corresponding EventReportBCSM or a DP specific operation is sent.

– If the Abandon DP is armed as an EDP-N and there is no CallInformationRequest pending, send an EventReportBCSM or a DP specific operation, then transit to the Idle state.

– If the Abandon DP is armed as an EDP-N and there is a CallInformationRequest pending, send an EventReportBCSM or a DP specific operation, followed by a CallInformationReport, then transit to the Idle state.

– Other pending requests that should be treated in the same way as the CallInformationRequest in the above list is the ApplyCharging.

In any state (except Idle), if a call party disconnects from a stable call (i.e. from the Active PIC in the BCSM), then the SSF-FSM should process this event as follows:

– If the Disconnect DP is not armed for that specific leg and there is no CallInformationRequest pending, transit to the Idle state.

– If the Disconnect DP is not armed and there is a CallInformationRequest pending, send a CallInformationReport and transit to the Idle state.

– If the Disconnect DP is armed as an EDP-R for that specific leg, send an EventReportBCSM or a DP specific operation, then transit to the Waiting For Instructions state. If a Call Information Request is pending, a CallInformationReport shall be sent before the corresponding EventReportBCSM or a DP specific operation is sent.

– If the Disconnect DP is armed as an EDP-N and there is no CallInformationRequest pending, send an EventReportBCSM or a DP specific operation, then transit to the Idle state.

– If the Disconnect DP is armed as an EDP-N and there is a CallInformationRequest pending, send an EventReportBCSM or a DP specific operation and a CallInformationReport, then transit to the Idle state.

– Other pending requests that should be treated in the same way as the CallInformationRequest in the above list is the ApplyCharging.

In any state (except Idle), an EventNotificationCharging can be sent to the SCF, if previously requested by a RequestNotificationCharging and if the charging event has been detected by the CCF. In this case no state transition takes place.

The SSF has an application timer, T_{SSF}, whose purpose is to prevent excessive call suspension time and to guard the association between the SSF and the SCF.

Timer T_{SSF} is set in the following cases:

– When the SSF sends an InitialDP, DP specific (refer to 3.1.1.5.2 State c: Waiting For Instructions), or AssistRequestInstructions operation (refer to 3.1.1.6.2 State b: Waiting For Instructions in assisting/handoff case). While waiting for the first response from the SCF, the timer T_{SSF} can be reset only once by a ResetTimer operation. Subsequent to the first response, the timer can be reset any number of times.

– When the SSF enters the Waiting For Instructions state (refer to 3.1.1.5.2) under any other condition than the ones listed in the previous case. In this case the SCF may reset the T_{SSF} timer using the ResetTimer operation any number of times.

– When the SSF receives a HoldCallInNetwork operation (refer to 3.1.1.5.2 State c: Waiting For Instructions). In this case the SCF may reset T_{SSF} using the ResetTimer operation any number of times.

– When the SSF enters the Waiting For End of User Interaction state or the Waiting For End of Temporary Connection state (refer to 3.1.1.5.3 and 3.1.1.5.4). In these cases the SCF may reset T_{SSF} using the ResetTimer operation any number of times.(OPTIONAL).

NOTE – This "OPTIONAL" means that the application timer T_{SSF} is optionally set. Whether it is used or not depends on implementation. But it must be synchronized with $T_{SCF\text{-}SSF}$ in the SCSM.

In the four above cases T_{SSF} may respectively have four different values as defined by the application.

When receiving or sending any operation which is different from the above, the SSF should reset T_{SSF} to the last used set value. This value is either one associated to the four different cases as listed above, or received in a ResetTimer operation, whatever occurred last. In the monitoring state (refer to 3.1.1.5.5) T_{SSF} is not used.

On expiration of T_{SSF} the SSF-FSM transitions to the idle state, aborts the interaction with the SCF and the CCF progresses the BCSM if possible.

The SSF state diagram contains the following transitions (events):

e1 TDP-N encountered

e3 InitiateCallAttempt received

e4 TDP-R encountered

e5 User interaction requested

e6 User interaction ended

e7 Temporary connection created

e8 Temporary connection ended

e9 Idle return from wait for instruction

e10 EDP-R encountered

e11 Routing instruction received

e12 EDP-N last encountered or ReleaseCall received or Cancel(allRequests) received

e13 Waiting for End of User Interaction state no change

e14 Waiting for Instructions state no change

e15 Waiting for End of Temporary Connection state no change

e16 Monitoring state no change

e17 Abandon (from any state) (not shown in the SSF state diagram)

e18 Disconnect (from any state) (not shown in the SSF state diagram)

e19 Non-Call Associated Treatment from any state (not shown in the SSF state diagram)

The SSF state diagram contains the following states:

State a Idle

State c Waiting for Instructions

State d Waiting for End of User Interaction

State e Waiting for End of Temporary Connection

State f Monitoring

3.1.1.5.1 State a – Idle

The SSF-FSM enters the Idle state under a variety of conditions, as described below.

The SSF-FSM enters the Idle state when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state.

The SSF-FSM enters the Idle state when one of the following occurs:

- when the call is abandoned or one or more call parties disconnect in any other state under the conditions identified in 3.1.1.5;
- when a Connect or proceed call processing operation is processed in the Waiting for Instructions state, and no EDPs are armed and there are no outstanding report requests (transition e9);
- when the application timer T_{SSF} expires in one of the states: Waiting for Instructions, Waiting for End of User Interaction or Waiting for End of Temporary Connection;
- when a ReleaseCall operation is processed in Waiting for Instructions (transition e9) or Monitoring (transition e12);
- when the last EDP-N is encountered in the Monitoring state, and there are no EDP-Rs armed and no monitoring is active for charging events (transition e12);
- when the last charging event is encountered in the Monitoring state, and there are no EDPs armed (transition e12).

When transiting to the Idle state, if there is a CallInformationRequest pending (see 3.1.1.5), the SSF sends a CallInformationReport operation to the SCF before returning to Idle. Once in the Idle state, if status reporting is still active the SSF deactivates it, any outstanding responses to send to the SCF are discarded.

During this state the following call-associated events can occur:

- indication from the CCF that an armed TDP is encountered related to a possible IN call/service attempt, the SSF-FSM acts as described below:
  - if the DP is a TDP-N, send a generic InitialDetectionPoint or DP-specific operation (see Note) to the SCF, as determined from DP processing; there is no resulting transition to a different state (transition e1);

- if the DP is a TDP-R, send a generic InitialDetectionPoint or DP-specific operation (see Note) to the SCF, as determined from DP processing, and transit to the Waiting for Instructions state (transition e4);
- the rules for DP processing are described in Recommendation Q.1214, section "DP Processing";

– a message related to a new transaction containing an InitiateCallAttempt operation is received from the SCF: in this case the SSF-FSM moves to the state Waiting for Instructions (transition e3).

Any other operation received from the SCF while the SSF is in Idle state should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TCAP (see Recommendation Q.774).

NOTE – DP specific operations are the following (refer to clause 2): TAnswer, TDisconnect, TermAttemptAuthorized, TMidCall, TNoAnswer, AnalysedInformation, TBusy, CollectedInformation, OAnswer, OCalledPartyBusy, ODisconnect, OMidCall, ONoAnswer, OriginationAttemptAuthorized, RouteSelectFailure.

3.1.1.5.2 State c – Waiting for Instructions

This state is entered from the Idle state, as indicated above (transition e4), directly from the Idle state on receipt at the SSF of a TC-BEGIN indication primitive containing an InitiateCallAttempt operation from the SCF (transition e3), from the state Monitoring on detection of an EDP-R (transition e10), from the state Waiting for End of User Interaction on occurrence of call disconnect from/to the SRF (transition e6), or from the state Waiting for End of Temporary Connection on occurrence of disconnection of temporary connection (transition e8).

In this state the SSF-FSM is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) should be set on entering this state.

During this state the following events can occur:

– The user dials additional digits (applies for open-ended numbering plans) – The CCF should store the additional digits dialled by the user.

– The user abandons or disconnects. This should be processed in accordance with the general rules in 3.1.1.5.

– The application Timer T_{SSF} expires – The SSF-FSM moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.

– An operation is received from the SCF – The SSF-FSM acts according to the operation received as described below.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e14):

– HoldCallInNetwork;

– RequestReportBCSMEvent;

– RequestNotificationChargingEvent;

– ResetTimer;

– FurnishChargingInformation;

– ApplyCharging;

– CallInformationRequest;

– SendChargingInformation;

– Cancel.

The following operations may be received from the SCF and processed by the SSF, causing a state transition to Waiting for End of User Interaction state (transition e5):

– ConnectToResource.

The following operations may be received from the SCF and processed by the SSF, causing a state transition to Waiting for End of Temporary Connection state (transition e7):

– EstablishTemporaryConnection

The following operations may be received from the SCF and processed by the SSF, causing a state transition to either Monitoring state (if any EDPs were armed or any reports were requested) (transition e11) or Idle state (transition e9):

- Connect;
- CollectInformation;
- AnalyseInformation;
- SelectRoute;
- SelectFacility;
- Continue;
- ReleaseCall.

ReleaseCall operation may be received from the SCF. In this case, the SSF-FSM should instruct the CCF to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows:

- if neither CallInformationRequest nor ApplyChargingReport operation has been requested, the SSF-FSM transits to the Idle state (transition e9);
- if CallInformationRequest or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the SSF-FSM transits to the Idle state (transition e9).

When processing the above operations, any necessary call handling information is provided to the Call Control Function (CCF).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.5.3 State d – Waiting for End of User Interaction

The SSF enters this state from the Waiting for Instructions state (transition e5) on the reception of one of the following operations:

- ConnectToResource.

During this state the following events can occur:

- A valid SCF-SRF operation [i.e. PlayAnnouncement, PromptAndCollectUserInformation, and Cancel(invokeID)] for relaying is received and is correct, the operation is transferred to the SRF for execution. The SSF-FSM remains in the Waiting for End of User Interaction state (transition e13).
- A valid SRF-SCF operation (i.e. SpecializedResourceReport and ReturnResult from PromptAndCollectUserInformation) for relaying is received and is correct, the operation is transferred to the SCF. The SSF-FSM remains in the Waiting for End of User Interaction state (transition e13).
- The application timer T_{SSF} expires (if it was set) – The SSF-FSM moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF – The SSF-FSM acts according to the operation received as described below.
- The user abandons – This should be processed in accordance with the general rules in 3.1.1.5.

The following operations may be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e13):

- RequestNotificationChargingEvent;
- ResetTimer;
- FurnishChargingInformation;
- ApplyCharging;
- Send Charging Information.

The DisconnectForwardConnection operation may be received from the SCF and processed by the SSF in this state. Call disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the Waiting for Instructions state. The disconnection is not transferred to the other party (transition e6).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.5.4 State e – Waiting for End of Temporary Connection

The SSF enters this state from the Waiting for Instructions state (transition e7) upon receiving an EstablishTemporaryConnection operation.

The call is routed to the assisting SSF-SRF and call handling is suspended while waiting for the end of the assisting procedure. The timer T_{SSF} is active in this state (whether it is used or not is optional).

During this state the following events can occur:

- The application timer T_{SSF} expires (if it was set) – The SSF-FSM moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.
- The receipt of an indication of disconnection of forward connection from the CCF – In this case, the SSF moves to the Waiting for Instructions state (transition e8). The disconnection is not transferred to the calling party.
- The user abandons – This should be processed in accordance with the general rules in 3.1.1.5
- An operation is received from the SCF – The SSF acts according to the operation received as described below.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e15):

- RequestNotificationChargingEvent;
- ResetTimer;
- FurnishChargingInformation;
- ApplyCharging;
- SendChargingInformation.

The DisconnectForwardConnection operation may be received from the SCF and processed by the SSF in this state. Call disconnect can also be received from the SRF. In both cases this causes the release of the connection to the SRF and the transition to the Waiting for Instructions state. The disconnection is not transferred to the other party (transition e8).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.5.5 State f – "Monitoring"

The SSF enters this state from the Waiting For Instructions state (transition e11) upon receiving a Connect, CollectInformation, AnalyseInformation, SelectRoute, SelectFacility, Continue, operation or processing an Initiate CallAttempt operation when one or more EDPs are armed or/and there are other reports pending (see 3.1.1.5).

In this state the timer T_{SSF} is not used, i.e. the expiration of T_{SSF} does not have any impact on the SSF-FSM.

During this state the following events can occur:

- An EDP-N should be reported to the SCF by sending an EventReportBCSM operation or a DP specific operation; the SSF-FSM should remain in the Monitoring state (transition e16) if one or more EDPs are armed or there are report requests pending. The SSF-FSM should move to the Idle state (transition e12) if there are no remaining EDPs armed or there are no report requests pending.

  If the event causing a CallInformationReport is also detected by an armed EDP-N, then the CallInformationReport shall be sent immediately before the corresponding EventReportBCSM or a DP specific operation is sent.

– An EDP-R should be reported to the SCF by sending an EventReportBCSM operation or a DP specific operation; the SSF-FSM should move to the Waiting For Instructions state (transition e10).

– If the event causing a CallInformationReport is also detected by an armed EDP-R, then the CallInformationReport shall be sent immediately after the corresponding EventReportBCSM or a DP specific operation is sent.

– The receipt of an END or ABORT primitive from TCAP has no effect on the call; the call may continue or be completed with the information available. In this case, the SSF-FSM transits to the Idle state (transition e12), disassociating the SSF-FSM from the call.

– An operation is received from the SCF: The SSF-FSM acts according to the operation received as described below.

– The user abandons or disconnects. This should be processed in accordance with the general rules in 3.1.1.5.

The following operations can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition e16):

– RequestNotificationChargingEvent;
– SendChargingInformation;
– FurnishChargingInformation;
– ApplyCharging.

The RequestReportBCSMEvent operation may be received from the SCF. In this case, the SSF-FSM transits as follows:

– if one or more EDPs are armed or a part of armed EDPs are disarmed, the SSF-FSM transits back to the same state (transition e16);

– if all of armed EDPs are disarmed and there is outstanding CallInformationRequest, EventNotificationCharging or ApplyChargingReport, the SSF-FSM transits back to the same state (transition e16);

– if all of armed EDPs are disarmed and there is no outstanding CallInformationRequest, EventNotificationCharging or ApplyChargingReport, the SSF-FSM transits to the Idle (transition e12).

The Cancel(allRequests) operation may be received from the SCF and processed by the SSF, causing a state transition to the Idle state (transition e12).

The ReleaseCall operation may be received from the SCF. In this case, the SSF-FSM should instruct the CCF to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows:

– if neither CallInformationRequest nor ApplyChargingReport operation has been requested, the SSF-FSM transits to the Idle state (transition e12);

– if CallInformationRequest or ApplyChargingReport operation has been requested, the SSF sends each operation which has been requested from SCF, and then the SSF-FSM transits to the Idle state (transition e12).

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

3.1.1.6 Assisting/Handoff SSF-FSM

This subclause describes the SSF-FSM related to the Assisting/Handoff SSF. The assisting SSF is structured as defined in 3.1.1.1 through 3.1.1.5. The Handoff FSM for CS-1 applies only to the case where final treatment is to be applied.

Within this subclause, the term "Assisting SSF" refers to both Assisting and Handoff SSFs unless an explicit indication to one or the other is provided.

The Assisting/Handoff SSF state diagram contains the following transitions (events) (see Figure 13):

ea1 Assist/Handoff detected

ea2 Assist/Handoff fail/success

ea3 User interaction requested

ea4 User interaction ended

ea5 Waiting For Instruction state no change

ea6 Waiting For End Of User Interaction state no change

ea7 Initiating bearer channel disconnect

The Assisting/Handoff SSF state diagram contains the following states:

State a Idle

State b Waiting for Instructions

State c Waiting for End of User Interaction

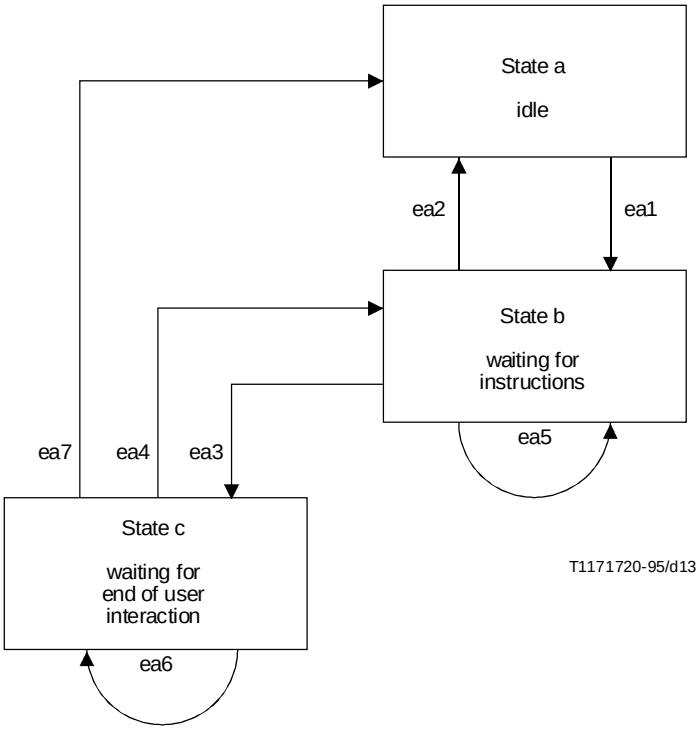


FIGURE 13/Q.1218

Assisting/Handoff SSF finite state machine

3.1.1.6.1 State a – Idle

The SSF-FSM enters the Idle state when one of the following occurs:

– when sending or receiving an ABORT TCAP primitive due to abnormal conditions in any state;

– given a temporary connection between an initiating SSF and the Assisting SSF, when a bearer channel disconnect is received from the initiating SSF (transition ea2).

Once in the Idle state, if there are any outstanding responses to send to the SCF, they are discarded by the Assisting SSF.

The Assisting SSF-FSM transits from the Idle state to the Waiting for Instructions state on receipt of an assist indication at the assisting SSF from another SSF (transition ea1).

Any operation received from the SCF while the Assisting SSF is in Idle state should be treated as an error. The event should be reported to the maintenance functions and the transaction should be aborted according to the procedure specified in TCAP (see Recommendation Q.774).

3.1.1.6.2 State b – Waiting for Instructions

This state is entered from the Idle state on receipt of a connect at an SSF from another SSF indicating that an assist is required, based on an implementation dependent detection mechanism (transition ea1).

Before entering this state, the SSF sends an AssistRequestInstructions operation to the SCF. In this state the Assisting SSF-FSM is waiting for an instruction from the SCF; call handling is suspended and an application timer (T_{SSF}) should be set on entering this state.

During this state the following events can occur:

- The application timer T_{SSF} expires – The Assisting SSF-FSM moves to the Idle state (transition ea2) and the expiration is reported to the maintenance functions and the transaction is aborted.
- An operation is received from the SCF – The SSF-FSM acts according to the operation received as described below.
- A bearer channel disconnect is received from the initiating SSF and the FSM moves to the Idle state (transition ea2).

The following operations may be received from the SCF and processed by the Assisting SSF with no resulting transition to a different state (transition ea5):

- ResetTimer;
- FurnishChargingInformation;
- ApplyCharging;
- SendChargingInformation.

The following operations can be received from the SCF and processed by the Assisting SSF, causing a state transition to Waiting for End of User Interaction state (transition ea3):

- ConnectToResource.

In the case of Handoff SSF the ReleaseCall operation can be received from the SCF. The Handoff SSF-FSM should instruct the CCF to clear the call and ensure that any CCF resources allocated to the call have been de-allocated, then continue processing as follows:

- if ApplyChargingReport operation is not requested, the Handoff SSF-FSM transits to the Idle state (transition ea2);
- if ApplyChargingReport operation has been requested, the Handoff SSF sends ApplyChargingReport to SCF and then the Handoff SSF-FSM transits to the Idle state (transition ea2).

Note that the above operation is allowed only in the Handoff SSF. In the case where an implementation is not capable of differentiating between a Handoff or an assisting SSF case, it may execute the ReleaseCall operation in the assisting SSF.

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

Note that multiple Handoff procedures are not covered by this Recommendation.

3.1.1.6.3 State c – Waiting for End of User Interaction

The Assisting SSF enters this state from the Waiting for Instructions state (transition ea3) on the reception of one of the following operations:

- ConnectToResource.

During this state the following events can occur:

– A valid SCF-SRF operation [i.e. Play announcement, prompt and collect user information, and cancel (announcement)] for relaying is received and is correct, the operation is transferred to the SRF for execution. The SSF-FSM remains in the Waiting for End of User Interaction state (transition ea6).

– When the SRF indicates to the assisting SSF the end of user interaction by initiating disconnection the assisting SSF-FSM returns to the Waiting for Instructions state (only in handoff case) (transition ea4).

– The application timer T_{SSF} expires – The SSF-FSM moves to the Idle state, the CCF routes the call if possible (e.g. default routing to a terminating announcement), the T_{SSF} expiration is reported to the maintenance functions and the transaction is aborted.

– An operation is received from the SCF – The SSF-FSM acts according to the operation received as described below.

– A bearer channel disconnect is received from the initiating SSF – The assisting SSF-FSM moves to the Idle state, the connection to the SRF is released and the transaction is terminated (transition ea7).

The following operation can be received from the SCF and processed by the SSF with no resulting transition to a different state (transition ea6):

– ResetTimer.

The DisconnectForwardConnection operation may be received from the SCF and processed by the assisting/handoff SSF in this state, causing a transition to the Waiting for Instructions state (transition ea4). This procedure is only valid if a ConnectToResource was previously processed to cause a transition into the Waiting for End of User Interaction state.

Any other operation received in this state should be processed in accordance with the general rules in 3.1.1.5.

3.1.2 SCF application entity procedures

3.1.2.1 General

This subclause provides the definition of the SCF Application Entity (AE) procedures related to the SCF-SSF/SRF/SDF interfaces. The procedures are based on the use of Signalling System No. 7 (SS No.7); other signalling systems can also be used.

In addition, other capabilities may be supported in an implementation-dependent manner in the SCP, AD or SN.

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes Transaction Capabilities Application Part (TCAP) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE and SACF/MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.774.

The procedure may equally be used with other message-based signalling systems supporting the Application Layer structures defined. By no means is this text intended to dictate any limitations to Service Logic Programs (SLPs).

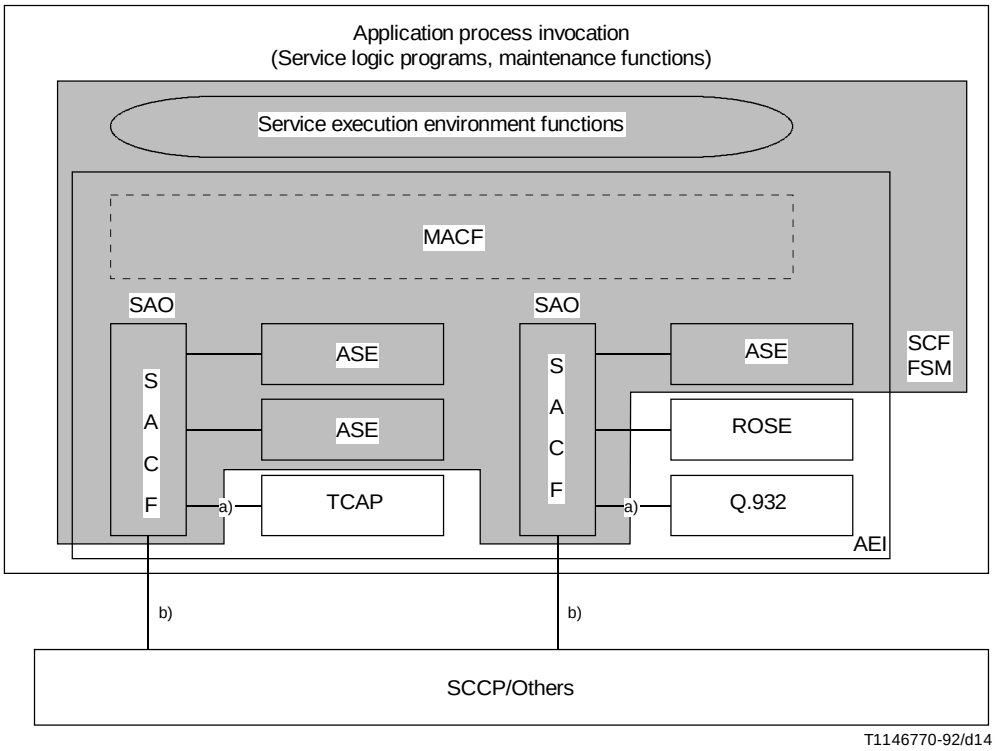
In case interpretations for the AE procedures defined in the following differ from detailed procedures and the rules for using the TCAP service, the statements and rules contained in 3.3 and 3.4 shall be followed.

3.1.2.2 Model and interfaces

The functional model of the AE-SCF is shown in Figure 14; the ASEs interface with supporting protocol layers to communicate with the SSF, SRF and SDF, and interface to the service logic programs and maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 14.

NOTE – The SCF-FSM includes several Finite State Machines.

The interfaces shown in Figure 14 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and N-primitives specified in Recommendation Q.711 [interface (2)]. The operations and parameters of INAP are defined in clause 2.



AEI Application Entity Invocation
SCF Service Control Functions
FSM Finite State Machine
MACF Multiple Association Control Function
SACF Single Association Control Function
SAO Single Association Object

a) TC-primitives or Q.932-primitives.
b) N-primitives.

NOTE – The SCF-FSM includes several finite state machines.

FIGURE 14/Q.1218

Functional model of SCF-AE

3.1.2.3 Relationship between the SCF-FSM and the SLPs/maintenance functions

The primitive interface between the SCF-FSM and the SLPs/maintenance functions is an internal interface and is not a subject of standardization in CS1.

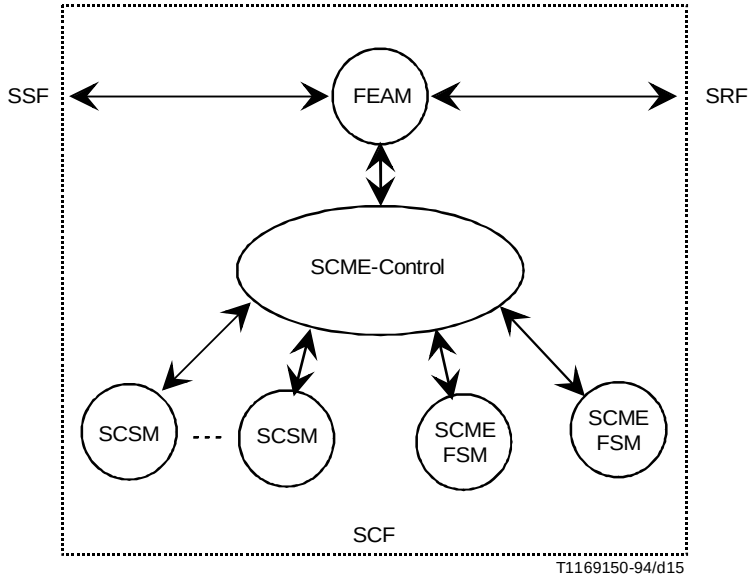
The relationship between the SLP and the SCF-FSM may be described as follows (for both cases where a call is initiated by an end user and by IN service logic):

– If a request for IN call processing is received from the SSF, an instance of an SCF Call State Model (SCSM) is created, and the relevant SLP is invoked.

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

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

The management functions related to the execution of operations received from the SCF are executed by the SCF Management Entity (SCME). The SCME is comprised of the SCME-Control and multiple instances of SCME FSMs. The SCME-Control interfaces different SCF Call State Models (SCSMs) and the Functional Entity Access Manager (FEAM). Figure 15 shows the SCF-FSM structure.



FEAM	Functional Entity Access Manager
SCME	SCF Management Entity
SCSM	SCF Call State Model

FIGURE 15/Q.1218

SCF-FSM structure

The following text systematically describes the procedural aspects of the interface between the SCF and other functional entities, with the main goal of specifying the proper order of operations rather than the functional capabilities of the entities. Consequently, this text describes only a subset of the SCF functional capabilities.

The procedural model associates an SCSM with each query from the SSF. The SCSM maintains dialogues with the SSF, SRF and SDF on behalf of service logic.

Multiple requests may be executed concurrently and asynchronously by the SCF, which explains the need for a single entity that performs the tasks of creation, invocation and maintenance of the SCF-FSM objects. This entity is called the SCF Management Entity-Control (SCME-Control). In addition to the above tasks, the SCME maintains the dialogues with the SSF, SDF and SRF on behalf of all instances of the SCF-FSMs. In particular, the SCME-Control:

1) interprets the input messages from other FEs and translates them into corresponding SCSM events;

2) translates the SCSM outputs into corresponding messages to other FEs;

3) performs some asynchronous (with call processing) activities (one such activity is flow control). It is the responsibility of the SCME-control to detect nodal overload and send the Overload Indication (e.g. Automatic Call Gap) tot the SSF to place flow control on queries. Other such activities include non-call associated treatment due to changes in Service Filtering, Call Gapping, or Resource Monitoring status and also provision of resource status messages to the SSF;

4) supports persistent interactions between the SCF and other FEs; and

5) performs asynchronous (with call processing) activities related to management and supervisory functions in the SCF and creates an instance of a SCME-FSM. For example, the SCME provides the non-call associated treatment due to changes in Service Filtering. Therefore, the SCME-Control separates the SCSM from the Service Filtering by creating instances of SCME-FSMs for each context of related operations.

The different contexts of the SCME-FSMs may be distinguished based on the address information provided in the initiating operations. In the case of service filtering, this address information is given by Filtering Criteria, i.e. all ActivateServiceFiltering operations using the same address, address the same SCME-FSM handling this specific service filtering instance. For example, ActivateServiceFiltering operations providing different Filtering Criteria cause the invocation of new SCME-FSMs.

Finally, the FEAM relieves the SCME of low-level interface functions. The functions of the FEAM include:

1) establishing and maintaining the interfaces to the SSF, SRF and SDF;

2) passing (and queueing when necessary) the messages received from the SSF, SRF and SDF to the SCME; and

3) formatting, queueing (when necessary), and sending messages received from the SCME to the SSF, SRF and SDF.

Note that although the SCSM includes a state and procedures concerning queue management, this type of resource management only represents one way of managing network call queues. Another alternative is to let the SSF-CCF manage call queues; however, the technical details of how the SSF-CCF performs queue management is beyond the scope of IN. As such, the RCO (see 3.1.2.4.5) and the queueing state of the SCSM (State 2.2), along with its relevant sub-states, events and procedures, are only required and applicable in the case where queue management is performed in the SCF.

3.1.2.4 Partial SCF Management Entity (SCME) State Transition Diagram

The two key parts of SCF Management Entity (SCME) State Diagram are described in Figures 16 and 17.

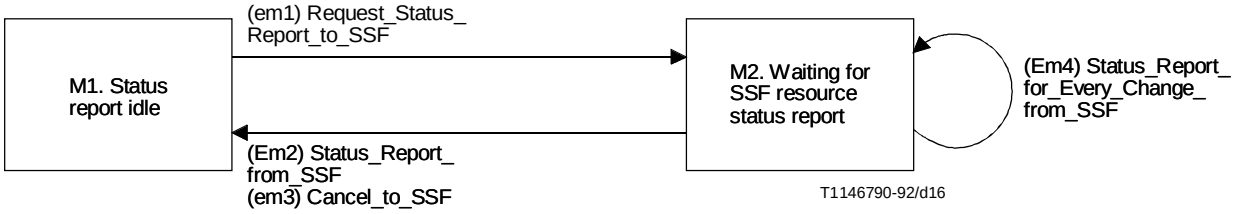


FIGURE 16/Q.1218

The status report FSM in the SCME

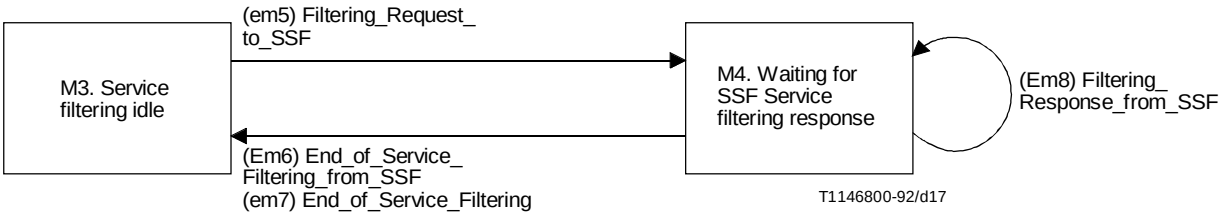


FIGURE 17/Q.1218

The service filtering FSM in the SCME

The SCME handles the following operations:

- **RequestCurrentStatusReport**;
- **RequestEveryStatusChangeReport**;
- **RequestFirstStatusMatchReport**;
- **CancelStatusReport (including the Invoke ID previously used for request FirstStatusMatchReport or RequestEveryStateChangeReport operations)**;
- **StatusReport**;
- **ActivateServiceFiltering**;
- **ServiceFilteringReport**;
- **CallGap**; and
- **ActivityTest**.

Issuing the **CallGap** and **ActivityTest** operations does not cause state transitions in the SCME. The procedures for the rest of the above operations are described below.

The operations that are not listed above do not affect the state of the SCME; these operations are passed to the relevant SCSM.

3.1.2.4.1 State M1 – "Status report idle"

The following event[1)] is considered in this state:

- (em1) Request_Status_Report_to_SSF – This is an internal event, caused by a decision to transmit one of the following operations:
  - request current status report;
  - request first status match report;
  - request every status change report.

This event causes a transition to state M2, Waiting for SSF response status report.

3.1.2.4.2 State M2 – "Waiting for SSF response status report"

The following events are considered in this state:

- (Em2) Status_Report_from_SSF – This is an external event, caused by the reception of the response to the request current status report or request first status match report operation previously issued to the SSF. This event causes a transition out of this state to state M1, Status report idle;

1) All events are enumerated, and the number of an event is prefixed with either the letter "E" (for external events) or "e" (for internal ones) and included in parentheses in the beginning of the event name. The scope of event names and numbers is defined by the state machine in which these events appear; the same applies to state names.

– (em3) Cancel_to_SSF – This is an internal event, caused by the service logic's need to end status monitoring of the resources in the SSF, and by transmission of cancel status report operation to the SSF. This event takes place only for previously issued request first status match report or request every status change report operations. This event causes a transition to state M1, Status report idle; and

– (Em4) Status_Report_for_Every_Change_from_SSF – This is an external event, caused by reception of the response to the request every status change report operation previously issued to the SSF. This event does not cause a transition out of this state, so the SCSM still remains in state M2, Waiting for the SSF resource status report.

3.1.2.4.3 State M3 – "Service filtering idle"

The following event is considered in this state:

– (em5) Filtering_Request_to_SSF – This is an internal event, caused by service logic's need to filter service requests to the SSF, and by transmission of the activate service filtering operation. This event causes a transition to state M4, Waiting for SSF service filtering response.

3.1.2.4.4 State M4 – "Waiting for SSF service filtering response"

In this state, the SCF is waiting for the service filtering response from the SSF. The following events are considered in this state:

– (Em6) End_of_Service_Filtering_Response_from_SSF – This is an external event, caused by reception of the response to the request service filtering previously issued to the SSF at the end of the service filtering duration. This event causes a transition out of this state to state M3, Service filtering idle.

– (em7) End_of_Service_Filtering – This is an internal event, caused by the expiration of service filtering duration timer in the SCF. This event causes a transition to state M3, Service filtering idle.

– (Em8) Filtering_Response_from_SSF – This is an external event, caused by reception of the response to the request service filtering operation previously issued to the SSF. This event does not cause a transition out of this state, and the SCSM remains in state M4, Waiting for SSF service filtering response.

When service filtering is active, another service filtering operation could be sent to the SSF that has the same filtering criteria; this second "filter" replaces the first one.

3.1.2.4.5 The Resource Control Object

The Resource Control Object (RCO) is part of the SCF management entity that controls data relevant to resource information.

The RCO consists of:

1) a data structure that (by definition) resides in the SDF and can be accessed only via the RCO's methods; and

2) the RCO methods.

For purposes of this Recommendation, no implementation constraints are placed on the structure. The only requirement to the structure is that, for each supported resource, it:

1) stores the resource's status (e.g. busy or idle); and

2) maintains the queue of SCSMs that are waiting for this resource. For continuous monitoring, the RCO maintains its knowledge of the status of the resources through use of the request every status change report operation.

The following three methods are defined for the RCO:

1) Get_Resource – This method is used to obtain the address of an idle line on behalf of an SCSM. If the resource is busy, the SCSM is queued for it;

2) Free_Resource – This method is used when a disconnect notification from the SSF is received. The method either advances the queue (if it is not empty) or makes the resource free (otherwise); and

3) Cancel – This method is used when either the queueing timer has expired or the call has been abandoned.

3.1.2.5 The SCF Call State Model (SCSM)

Figure 18 shows the general State Diagram of the SCSM as relevant to the procedures concerning the SCF-FSM part of the SCP/AD/SN during the processing of an IN call. Each state is discussed in one of the following subclauses. Each state may have internal sub FSMs composed of the sub-states.

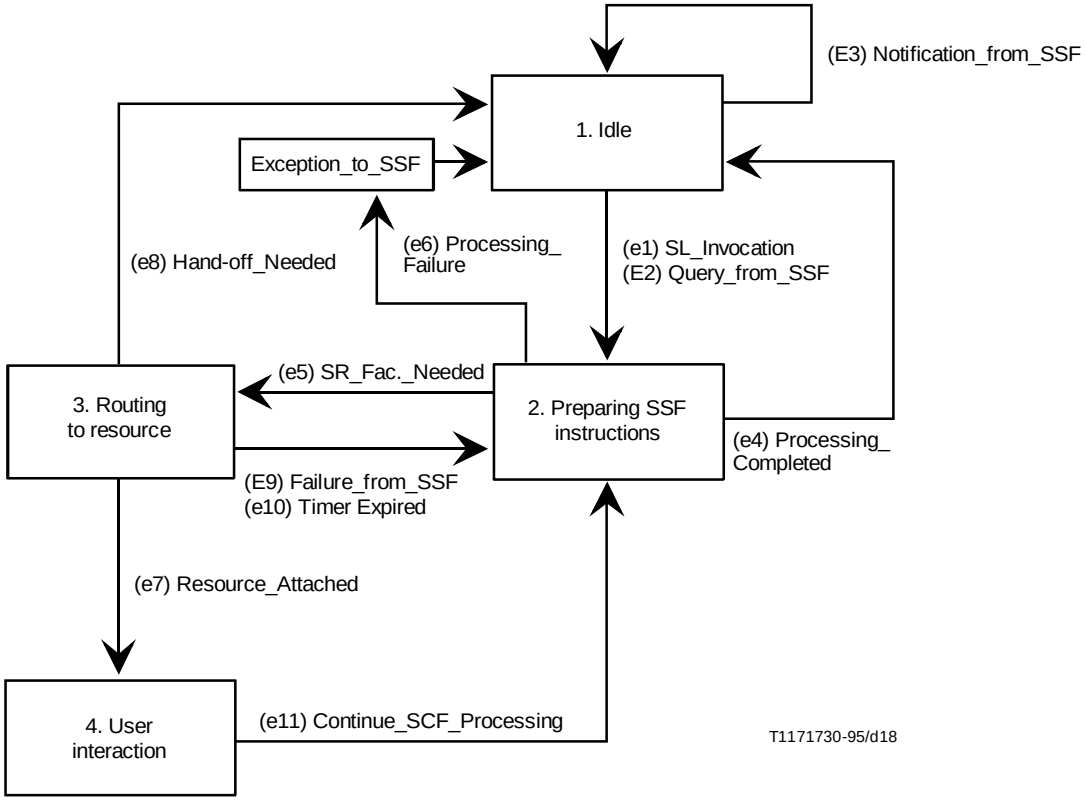


FIGURE 18/Q.1218

The SCSM finite state machine

General rules applicable to more than one state are as follows:

In every state, if there is an error in a received operation, the SLP and the maintenance functions are informed. Generally, the SCSM remains in the same state; however, different error treatment is possible in specific cases as described in 3.3. Depending on the class of the operation, the error can be reported to the SSF, SRF, or SDF (see Recommendation Q.774).

It also holds that, in every state, if the SCSM is informed that the dialogue with the SSF is terminated, then it informs the SLP and returns to the Idle state. In this case, all resources allocated to that call, including those required for relevant dialogues with the other functions, shall be de-allocated. To simplify the diagram, such transitions are not demonstrated in the figures.

When the SLP requests call information, the SCSM transmits the **Call InformationRequest** operation to the SSF, and then **CallInformationReport** is outstanding.

In any state (except Idle), the SCSM may receive the **CallInformationReport** operation from the SSF when the **CallInformationReport** is outstanding. Other pending requests that should be treated in the same way as the **CallInformationRequest** is the **ApplyCharging** operation.

From any state (except Idle), if **CallInformationReport** is outstanding and the SLP indicates that the processing has been completed, the SCSM remains in the same state until it receives the **CallInformationReport** operation.

The general rules for one or a sequence of components sent in one or more TCAP messages, which may include a single operation or multiple operations, are specified in 3.1.1.5 and are not described here.

The SCSM has an application timer, $T_{SCF\text{-}SSF}$, whose purpose is to reset the timer T_{SSF}, which is used to prevent excessive call suspension time and to guard the association between the SSF and the SCF.

Timer $T_{SCF\text{-}SSF}$ is set in the following cases:

- when the SCF receives an **InitialDP, TAnswer, TDisconnect, TermAttemptAuthorized, TMidCall, TNoAnswer, AnalysedInformation, TBusy, CollectedInformation, OAnswer, OCalledPartyBusy, ODisconnect, OMidCall, ONoAnswer, OriginationAttemptAuthorized, RouteSelectFailure,** or **AssistRequestInstructions** operation (see 3.1.2.5.2.1 state 2.1: "**Preparing SSF Instructions**", and 3.1.2.5.2.2.1 state 2.2.1: "**Preparing SSF Instructions**"). In this case, this timer is reset when the first request, other than **ResetTimer** operation, is sent to the SSF. Subsequent to the sending of the first response, the SCF may reset T_{SSF} any number of times, using the **ResetTimer** operation, and also reset $T_{SCF\text{-}SSF}$. Prior to the sending of the first response, on the expiration of $T_{SCF\text{-}SSF}$, the SCF may reset T_{SSF} once, using the **ResetTimer** operation, and also reset $T_{SCF\text{-}SSF}$. On second expiration of $T_{SCF\text{-}SSF}$, the SCSM informs SLP and the maintenance functions, and the SCSM transits to the **Idle** state;

- when the SCF enters the Preparing SSF Instructions state after sending **InitiateCallAttempt** operation or **DisconnectForwardConnection** operation or after receiving a report on EDP-R from the SSF. In this case, the SCF may reset T_{SSF} using the **ResetTimer** operation any number of times;

- when the SCF enters the Queueing sub-state (see 3.1.2.5.2.2, State 2.2.2: "**Queueing**"). In this case, on the expiration of timer $T_{SCF\text{-}SSF}$, the SCF may reset T_{SSF} using the **ResetTimer** operation any number of times; and

- when the SCF enters the "**Waiting for Assist Request Instructions**" state or the "**User Interaction**" state (see 3.1.2.5.3.2 and 3.1.2.5.4). In these cases, on the expiration of $T_{SCF\text{-}SSF}$, the SCF may reset T_{SSF} using the **ResetTimer** operation any number of times (OPTIONAL).

  NOTE – The word “OPTIONAL” refers to the use of the application timer $T_{SCF\text{-}SSF}$. Whether it is used or not depends on an implementation, but, if used, it must be synchronized with T_{SSF} in the SSF-FSM.

In all four cases, $T_{SCF\text{-}SSF}$ may respectively have four different values as defined by the application. The value of $T_{SCF\text{-}SSF}$ are smaller than the respective values of T_{SSF}.

In all other cases, when receiving or sending any operation, the SCF should reset $T_{SCF\text{-}SSF}$. In the "**Waiting for Notification or Request**" state (see 3.1.2.5.2.3), $T_{SCF\text{-}SSF}$ is not used.

The SCSM also has an application timer, $T_{ASSIST/HANDOFF}$, whose purpose is to prevent excessive assist/handoff suspension time. The SCSM sets the timer $T_{ASSIST/HANDOFF}$ when the SCSM sends the **EstablishTemporaryConnection** or **Select Route/Connect** operation with a correlation ID. This timer is stopped when the SCSM receives the **AssistRequestInstructions** operation from the assisting/handed-off SSF. On expiration of $T_{ASSIST/HANDOFF}$, the SCSM informs service logic and the maintenance functions. Then, in the Assist case, the SCSM transits to the **Preparing SSF Instructions** state, and in the Handoff case, the SLPI will be released.

The call-control-related operations relevant to the SCF-SSF interface (except the SCME related operations) are categorized into:

1) call-processing-related operations; and
2) non-call-processing-related operations.

Call-processing-related operations are grouped into the following two sets:

– **CollectInformation**;
– **AnalyseInformation**;
– **SelectFacility**;
– **SelectRoute**;
– **Connect**;
– **Continue**.

and

– **InitiateCallAttempt**;
– **ConnectToResource**;
– **DisconnectForwardConnection**;
– **ReleaseCall**;
– **EstablishTemporaryConnection**.

For the first set of call-processing-related operations, the SCF may not send two operations of the same set in a series of TCAP messages or in a component sequence to the SSF, but send them only one at a time. Two operations of the first set shall be separated by at least one EDP-R message received by the SCSM. The same applies for any operation of the first set followed by ConnectToResource or Establish Temporary Connection.

The non-call-processing operations include the rest of the operations at the SCF-SSF interface (but not the SCME related operations). When the service logic needs to send operations in parallel, they are sent in the component sequence.

In what follows, each state is described in a separate subclause together with the events that cause a transition out of this state. The outputs are presented within smaller rectangles than the states are; unlike the states and events, the outputs are not enumerated.

3.1.2.5.1 State 1 – "Idle"

The following events are considered in this state:

- (e1) SL_Invocation – This is an internal event caused by the need of the service logic to start a call. The SCSM issues the **InitiateCallAttempt** operation to the SSF.
- (E2) Query_from_SSF – This is an external event, caused by a reception of one of the following operations:
  – **InitialDP**;
  – **AssistRequestInstructions** (for the Service Handoff case);
  – **TAnswer**;
  – **TDisconnect**;
  – **TermAttemptAuthorized**;
  – **TMidCall**;
  – **TNoAnswer**;
  – **AnalysedInformation**;
  – **TBusy**;
  – **CollectedInformation**;
  – **OAnswer**;
  – **OCalledPartyBusy**;
  – **ODisconnect**;
  – **OMidCall**;
  – **ONoAnswer**;
  – **OriginationAttemptAuthorized**; and
  – **RouteSelectFailure**.

In the case of handoff, the timer $T_{ASSIST/HANDOFF}$ is stopped.

Both events cause a transition to State 2, **PreparingSSFInstructions.**

- (E3) Notification_from_SSF – This is an external event caused by the reception of **InitialDP** or **TAnswer, TDisconnect, TermAttemptAuthorized, TMidCall, TNoAnswer, AnalysedInformation, TBusy, CollectedInformation, OAnswer, OCalledPartyBusy, ODisconnect, OMidCall, ONoAnswer, OriginationAttemptAuthorized, RouteSelectFailure** operation that notifies the detection of TDP_N in the SSF. This event causes a transition back to the same state.

3.1.2.5.2 State 2 – "Preparing SSF Instructions"

In this state, the SCF determines how to further process.

The following events are considered in this state:

- (e4) Processing_Completed – This is an internal event. In this case, the SCF has completed the processing of the instructions to the SSF. This event causes a response to be sent to the SSF and a transition to State 1, **Idle**.
- (e5) SR_Facilities_Needed – This is an (internal) event caused by the need of the service logic for additional information from the call party; hence is the necessity to set up a connection between the call party and the SRF. This event causes a transition to State 3, **Routing to Resource**.
- (e6) Processing_Failure – This (internal) event causes an appropriate exception processing and a transition back to State 1, **Idle**.

NOTE – Here and further in this Recommendation, the exception processing is not defined. It is assumed, however, that it has to include releasing all the involved resources and sending an appropriate response message to the SSF.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following three subclauses (this subdivision is illustrated in Figure 19).

3.1.2.5.2.1 State 2.1 – "Preparing SSF Instructions"

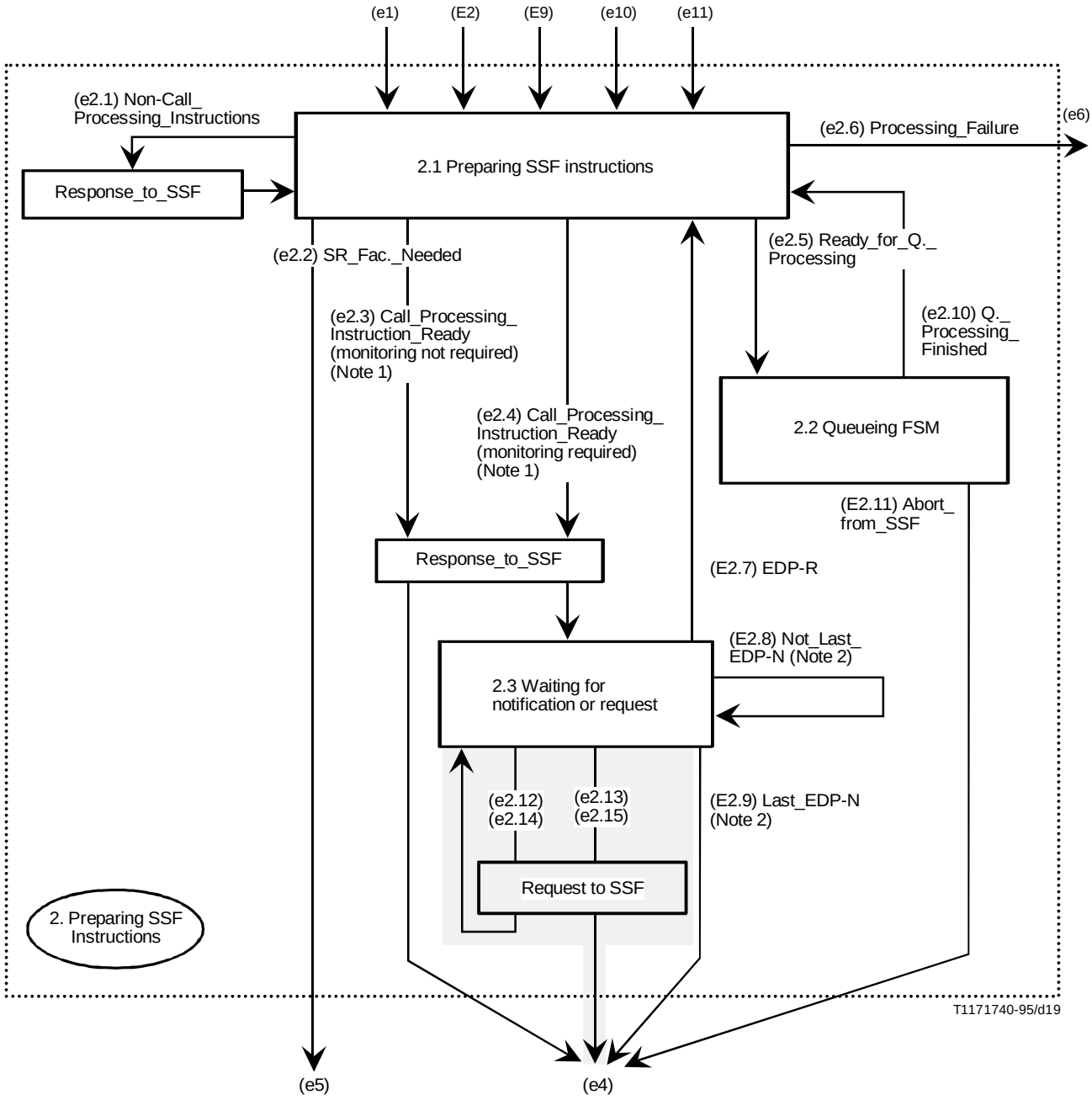
State 2.1, **Preparing SSF Instructions**, is where the initial decision is made on whether the SDF information or a Specialized Resource is needed, whether queueing is supported, etc. In addition, the EDP-R-related processing is also performed in this state.

On entering this state, the SCSM starts or resets the timer $T_{SCF\text{-}SSF}$.

The following events are considered in this state:

- (e2.1) Non-Call_Processing_Instructions – This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes one or more of the following operations to be issued to the SSF:
  - **ApplyCharging**;
  - **CallInformationRequest**;
  - **Cancel (allRequests)**;
  - **FurnishChargingInformation**;
  - **RequestReportBCSMEvent**;
  - **RequestNotificationChargingEvent**;
  - **ResetTimer**; and
  - **SendChargingInformation**.

  This event causes a transition back to State 2.1, **Preparing SSF Instructions**.
- (e2.2) SR_Facilities_Needed – This is an internal event, caused by the service logic when there is a need to use the SRF. This event maps into the SCSM event (e5).
- (e2.3) Call_Processing_Instruction_Ready (Monitoring not required) – This is an internal event caused by the service logic when the final call-processing-related operation is ready and there is no armed EDP and no outstanding **CallInformationReport**, **EventNotificationCharging** or **ApplyChargingReport** operation. It causes one of the following operations to be issued to the SSF:
  - **AnalyseInformation**;
  - **Connect**;
  - **Continue**;
  - **ReleaseCall**;
  - **SelectFacility**; and
  - **SelectRoute**.



NOTES

1 Including Call Information Request, Apply Charging with report request and Request Notification Charging Event.

2 Including Call Information Report, Apply Charging Report and Event Notification Charging.

(e2.12) Notification_or_Request_Continuing_Instruction

(e2.13) Monitoring_Cancel_Instruction

(e2.14) Release_Call_Instruction (Call Information Report or Apply Charging Report has been requested)

(e2.15) Release_Call_Instruction (Neither Call Information Report nor Apply Charging Report has been required)

FIGURE 19/Q.1218

Partial expansion of the State 2 FSM

In addition, one or more of the following operations may be issued to the SSF prior to the operations listed above:

– **Cancel (allRequests)**;
– **RequestReportBCSMEvent** (to disarm all armed EDPs);
– **FurnishChargingInformation**, and
– **SendChargingInformation**.

This event maps into the SCSM event (e4).

- (e2.4) Call_Processing_Instruction_Ready (Monitoring required) – This is an internal event caused by the service logic when a call-processing-related operation is ready and the monitoring of the call is required (e.g. an EDP is set, or there is an outstanding **CallInformationReport** or **ApplyChargingReport**, or there is a need to issue such a request). It causes one of the following operations to be issued to the SSF:

  – **AnalyseInformation**;
  – **CollectInformation**;
  – **Connect**;
  – **Continue**;
  – **ReleaseCall**;
  – **SelectFacility**; and
  – **SelectRoute**.

  In addition, one or more of the following operations may be issued to the SSF prior to issuing one of the operations listed above:

  – **ApplyCharging**;
  – **CallInformationRequest**;
  – **FurnishChargingInformation**;
  – **RequestReportBCSMEvent**;
  – **RequestNotificationChargingEvent**; and
  – **SendChargingInformation**.

  This event causes a transition into State 2.3, **Waiting for Notification or Request.**

- (e2.5) Ready_for_Queueing_Processing – This is an internal event caused by the service logic when queuing of the call is required. This event causes a transition into State 2.2, **Queueing FSM.**
- (e2.6) Processing_Failure – This is an internal event, and it maps into the SCSM event (e6) Processing_Failure.

3.1.2.5.2.2 State 2.2 – "Queueing FSM"

When the SCF is processing the query from the SSF-CCF, it may find that the resource to which the call shall be routed is unavailable. One possible reason causing the resource to be unavailable is the "busy" condition.

NOTE – The manner in which the status of the resources is maintained is described in 3.1.2.4.5.

Such a resource may be an individual line or trunk or a customer-defined group of lines or trunks. In the latter case, the word "busy" means that all lines or trunks in the group are occupied; and the word "idle" means that at least one line or trunk in the group is idle.

If the resource is busy, the SCF may put the call on queue and resume it later when the resource is idle. The following operations can be sent in this state:

– **HoldCallInNetwork**;
– **ApplyCharging**;
– **CallInformationRequest**;
– **FurnishChargingInformation**;
– **RequestReportBCSMEvent**;
– **RequestNotificationChargingEvent**;
– **ResetTimer**; and
– **SendChargingInformation**.

The following events are considered in this state:

- (e2.10) Queueing_Processing_Finished – This is an internal event caused by the SLP when it is ready to prepare the call-related operation to the SSF. This event causes a transition to State 2.1, **Preparing SSF Instructions**.

- (E2.11) Abort_from_SSF – This is an external event caused by the reception of the **Abort** message from the SSF (on call abandonment), and it causes a transition that maps into the SCSM event (e4).

This state further expands into an FSM, which is depicted in Figure 20.

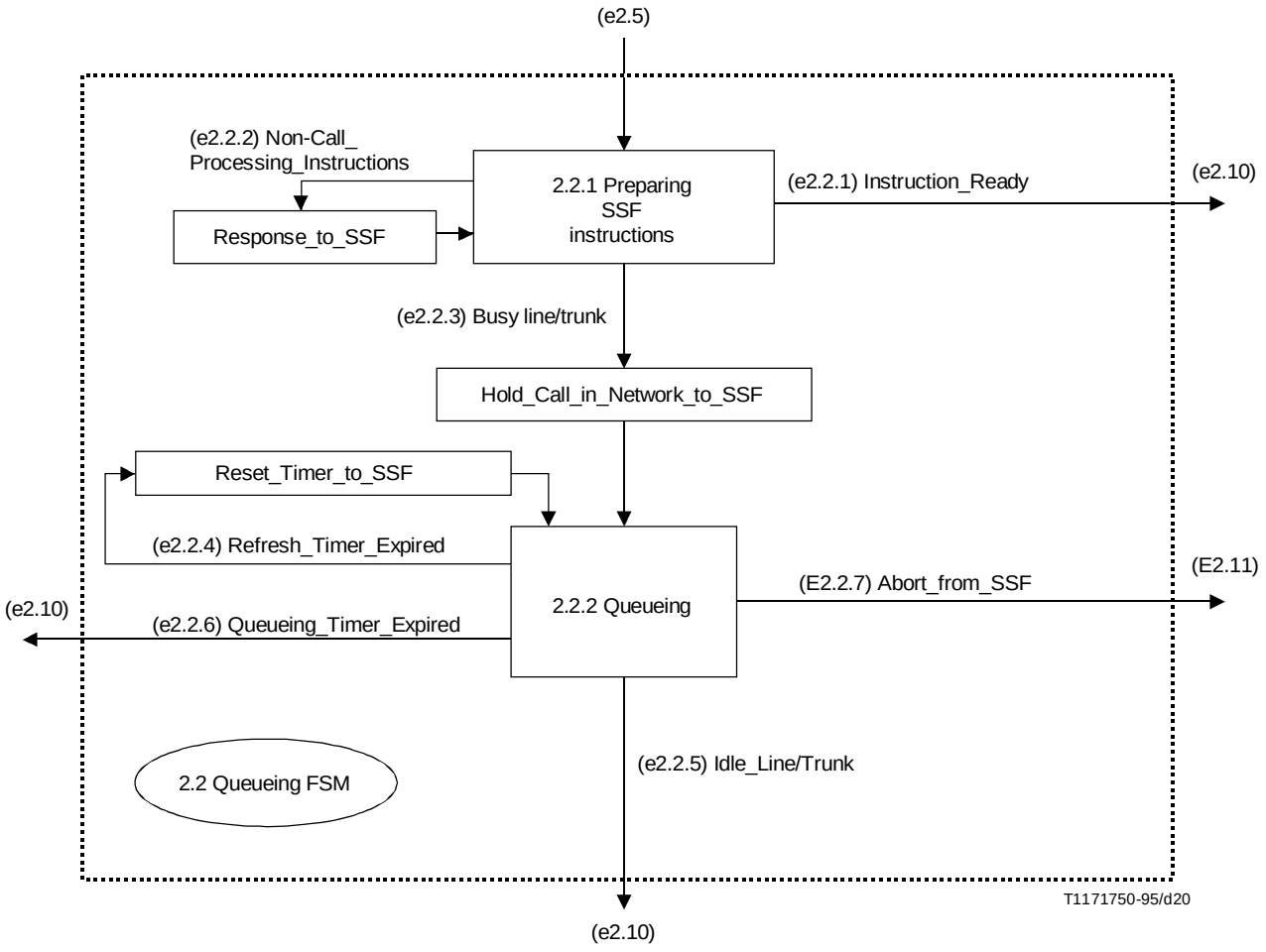


FIGURE 20/Q.1218

Partial expansion of the State 2 FSM as related to queueing

This FSM does not explicitly describe all possible combinations of resource monitoring functions used for queueing. The following possibilities may be used in implementations:

– **RequestFirstStatusMatchReport** by means of SCME;

– **RequestCurrentStatusChangeReport** by means of the SCME;

– **RequestEveryStatusChangeReport** by means of the SCME; and

– monitoring based on issuing by the SCSM of the **RequestEventReportBCSM** operation and subsequent reception of the **EventReportBCSM** operation to report the availability of the resource. Both the Request and Report occur in a single different call context. In this case, operations to the SDF or equivalent SCF functionality may be used for scanning the status of resources.

In the rest of this subclause, the state-by-state description of the FSM is followed by the description of the supporting mechanisms of the SCME.

3.1.2.5.2.2.1 State 2.2.1 – "Preparing SSF Instructions"

In this state, the SCSM prepares the instructions for the SSF to complete the call. The following events are considered in this state:

- (e2.2.1) Instruction_Ready – This is an internal event that takes place only when the required resource is available. In this case, the SCSM has obtained the address of the free resource via the Get_Resource method of the RCO (see 3.1.2.4.5). This event causes a transition to the State 2.1, **PreparingSSFInstructions** [transition (e2.10)].

- (e2.2.2) Non-Call_Processing_Instructions – This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes one or more of the following operations to be issued to the SSF:

  – **ApplyCharging**;
  – **CallInformationRequest**;
  – **FurnishChargingInformation**;
  – **RequestReportBCSMEvent**;
  – **RequestNotificationChargingEvent**;
  – **ResetTimer**; and
  – **SendChargingInformation**.

  This event causes a transition back to State 2.2.1, **Preparing SSF Instructions.**

- (e2.2.3) Busy_Line/Trunk – This is an internal event caused by the RCO when no terminating line/trunk is available. This event causes the operation, with a suitable queueing timervalue, to be sent to the SSF, and a transition to State 2.2.2, **Queueing.**

3.1.2.5.2.2.2 State 2.2.2 – "Queueing"

In this state, the SCSM is awaiting an indication from the RCO to proceed with routing a call to an idle trunk/line. The support of playing various announcements is also provided when the SCSM is in this state. In this Recommendation, the relevant further expansion of the state is not provided; it is, however, not different from that of the SCSM States 3 and 4. Nevertheless, if announcements are completed before the call is dequeued and the SSF-FSM has transitioned to state **Waiting for Instructions**, the operation should be sent to set the T_{SSF} with an appropriate value. Once the SCSM enters this state, the Queueing Timer is started, and the $T_{SCF\text{-}SSF}$ is reset. The respective roles of these timers are as follows:

1) The Queueing Timer limits the time that a call can spend in the queue, and its value may be customer-specific.

2) The $T_{SCF\text{-}SSF}$ signals when the **ResetTimer** operation shall be sent to the SSF-CCF lest the latter abandons the call. Hence, the value of this timer is set in agreement with that of the relevant timer T_{SSF} within the SSF-CCF.

The **FurnishChargingInformation** operation may also be sent to the SSF at this time to indicate the initiation of queueing for call logging purposes.

The following events are considered in this state:

- (e2.2.4) Refresh_Timer_Expired – This is an internal event, which results in sending the **ResetTimer** operation to the SSF-CCF and a transition back to the same state.

- (e2.2.5) Idle_Line/Trunk – This is an internal event, which maps into the State 2 event (e2.10).

- (e2.2.6) Queueing_Timer_Expired – This is an internal event, which results in processing the Cancel method of the RCO and causes a transition out of this state to the State 2.1, **Preparing SSF Instructions** [Transition (e2.10)] [following procedures depends on decision of the service logic that may play (or not play) the terminating announcement].
- (E2.2.7) Abort_from_SSF – This is an external event caused by the reception of the Abort message from the SSF (on call abandonment), and it causes a transition that maps into the State 2 event (E2.11). The Service Control Managing Entity (SCME) takes care of updating the queueing data via the Cancel method of the RCO.

3.1.2.5.2.3 State 2.3 – "Waiting for Notification or Request"

In this state, the SCSM waits for a notification or a request from the SSF.

On entering this state the SCSM stops the timer $T_{SCF\text{-}SSF}$.

The following events are considered in this state:

- (E2.7) EDP-R – This is an external event, caused by a reception of the following operations:
  - **EventReportBCSM** (for EDP_R);
  - **TAnswer**;
  - **TDisconnect**;
  - **TermAttemptAuthorized**;
  - **TMidCall**;
  - **TNoAnswer**;
  - **AnalysedInformation**;
  - **TBusy**;
  - **CollectedInformation**;
  - **OAnswer**;
  - **OCalledPartyBusy**;
  - **ODisconnect**;
  - **OMidCall**;
  - **ONoAnswer**;
  - **OriginationAttemptAuthorized**; and
  - **RouteSelectFailure**.

  This event causes a transition to State 2.1, **Preparing SSF Instructions.**
- (E2.8) Not_Last_EDP-N – This is an external event, caused by a reception of one of the following operations:
  - **ApplyChargingReport**;
  - **CallInformationReport**;
  - **EventReportBCSM (**for EDP_N);
  - **EventNotificationCharging**;
  - **TAnswer**;
  - **TDisconnect**;
  - **TermAttemptAuthorized**;
  - **TMidCall**;
  - **TNoAnswer**;
  - **AnalysedInformation**;
  - **TBusy**;
  - **CollectedInformation**;
  - **OAnswer**;
  - **OCalledPartyBusy**;
  - **ODisconnect**;
  - **OMidCall**;
  - **ONoAnswer**;
  - **OriginationAttemptAuthorized**; and
  - **RouteSelectFailure**.

  In this case, there is still an outstanding armed EDP or pending **CallInformationReport** or **ApplyChargingReport** operation. This event causes a transition back to State 2.3, **Waiting for Notification or Request.**

- (E2.9) Last_EDP-N – This is an external event, caused by a reception of one of the following operations:
  - **ApplyChargingReport**;
  - **CallInformationReport**;
  - **EventReportBCSM** (for EDP_N);
  - **TAnswer**;
  - **TDisconnect**;
  - **TermAttemptAuthorized**;
  - **TMidCall**;
  - **TNoAnswer**;
  - **AnalysedInformation**;
  - **TBusy**;
  - **CollectedInformation**;
  - **OAnswer**;
  - **OCalledPartyBusy**;
  - **ODisconnect**;
  - **OMidCall**;
  - **ONoAnswer**;
  - **OriginationAttemptAuthorized**; and
  - **RouteSelectFailure**.

  In this case, there is no outstanding armed EDP and no pending **CallInformationReport** or **ApplyChargingReport**. This event maps into the SCSM event (e4).
- (e2.12) Notification_or_Request_Continuing_Instruction – This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes one of the following operations to be issued to the SSF:
  - **RequestReportBCSMEvent** (for the purpose of which:
    i) one or more EDPs will be armed;
    ii) some armed EDPs will be disarmed; or
    iii) all armed EDPs will be disarmed when there are other pending requests).
  - **FurnishChargingInformation**;
  - **ApplyCharging**;
  - **RequestNotificationChargingEvent**; and
  - **SendChargingInformation**.

  This event causes a transition to State 2.3, **Waiting for Notification or Request**.
- (e2.13) Monitoring_Cancel_Instruction – This is an internal event caused by the service logic when there is a need to send this operation to the SSF. It causes the following operation to be issued to the SSF:
  - **Cancel** (all Requests); and
  - **RequestReportBCSMEvent** (for the purpose of which all armed EDPs will be disarmed when there are no more other pending requests).

  This event maps into the SCSM event (e4).
- (e2.14) Release_Call_Instruction (**CallInformationReport** or **ApplyChargingReport** has been requested) – This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes the following operation to be issued to the SSF:
  - **ReleaseCall** (when there is an outstanding **CallInformationReport** or **ApplyChargingReport)**.

  This event causes a transition to State 2.3, **Waiting for Notification or Request**.
- (e2.15) Release_Call_Instruction (neither **CallInformationReport** nor **ApplyChargingReport** has been requested) – This is an internal event caused by the service logic when there is a need to send such an operation to the SSF. It causes the following operation to be issued to the SSF:
  - **ReleaseCall** (when there is no outstanding **CallInformationReport** or **ApplyChargingReport**).

  This event maps into the SCSM event (e4).

This concludes the description of State 2, **Preparing SSF Instructions**.

3.1.2.5.3 State 3 – "Routing to Resource"

The resource is any SRF facility (e.g. Intelligent Peripheral).

In this state, interactions with the SSF are necessary. Accordingly, the following events cause transitions out of this state:

- (e7) Resource_Attached – The SRF is available. This event causes a transition to State 4, **User Interaction**;
- (e8) Handoff_Needed – When the Handoff procedure is initiated, the SCSM terminates the interaction with the initiating SSF. This event causes a transition to the State 1, **Idle**. When the **AssistRequestInstructions** operation from the Handed-off SSF is received by the SCME-Control, the SCME-Control creates the new SCSM. The SCF shall maintain sufficient information in order to correlate the subsequent AssistRequestInstructions operation (from the Handoff SSF) to the existing Service Logic Program Instance (SLPI);
- (E9) Failure_from_SSF – The inability of the SSF to connect to requested resources causes a transition to State 2, **Preparing SSF Instructions**; and
- (e10) Timer_Expired – This event takes place when $T_{ASSIST/HANDOFF}$ expires. This event causes a transition to State 2, **Preparing SSF Instructions**.

To further describe the procedures relevant to this state, the state is divided into three sub-states, which are described in the following two subclauses (this subdivision is illustrated in Figure 21).

3.1.2.5.3.1 State 3.1 – "Determine Mode"

In this state, the SCSM determines the User Interaction mode to connect the call to SRF. The following events are considered in this state:

- (e3.1) Instruction_Ready – This is an internal event that takes place only in the Initiating SSF relay case. In this case, the SCSM sends the **ConnectToResource** operation accompanied by **PlayAnnouncement** or **PromptAndCollectUserInformation** operation to the Initiating SSF, and transits out to the State 4, **User Interaction**. This transition maps into the event (e7);
- (e3.2) Assist_Needed – This event is an internal event that takes place when the Assisting SSF is needed or Direct SCF-SRF relation is needed. In this case, the SCSM sends the **EstablishTemporaryConnection** operation to the Initiating SSF with the Assisting SSF address or SRF address, and transits to the State 3.2, **WaitingForAssistRequestInstructions**; and
- (e3.3) Handoff_Needed – This is an internal event that takes place only in the Handoff case. In this case, the SCSM sends the **Connect** or **SelectRoute** operation with the Handed-off SSF address to the Initiating SSF, sets the $T_{ASSIST/HANDOFF}$ timer, and transits to the State 1, **Idle**. This transition maps into the event (e8). The SCF shall maintain sufficient information in order to correlate the subsequent **AssistRequestInstructions** operation (from the handoff SSF) to the existing SLPI.

3.1.2.5.3.2 State 3.2 – "Waiting for Assist Request Instructions"

In this state, the SCSM waits for the **AssistRequestInstructions** operation from the Assisting SSF (SSF relay case) or from the SRF (Direct SCF-SRF case). On entering this state the SCSM starts the Timer $T_{ASSIST/HANDOFF}$, and resets the timer $T_{SCF-SSF}$ (if it is used). The following events are considered in this state:

- (E3.4) Assist_Request_Instructions_from_SSF (Assisting SSF relay case) – This is an external event caused by the receipt of **AssistRequestInstructions** from the Assisting SSF. In this case, the SCSM transmits the **ConnectToResource** operation accompanied by **PlayAnnouncement** or **PromptAndCollectUserInformation** operation to the Assisting SSF, and transits out to the State 4, **User Interaction**. This transition maps into the event (e7);
- (E3.5) Assist_Request_Instructions_from_SRF (Direct SCF-SRF case) – This is an external event caused by the receipt of **AssistRequestInstructions** from the SRF. In this case, the SCSM transmits the **PlayAnnouncement** or **PromptAndCollectUserInformation** operation to the SRF, and transits out to the State 4, **User Interaction**. This transition maps into the event (e7);

- (e3.6) Refresh_Timer_Expired (OPTIONAL: see the Note in 3.1.2.5) – This is an internal event that takes place on the expiration of $T_{SCF-SSF}$. In this case, the SCSM transmits the **ResetTimer** operation to the Initiating SSE, and transits back to the same state;

- (e3.7) Assist_Timer_Expired – This is an internal event that takes place on the expiration of $T_{ASSIST/HANDOFF}$. In this case, the SCSM informs the SCME and SLP, and transits to State 2, **Preparing SSF Instructions**, after sending the **DisconnectForwardConnection** operation to the SSF. This event maps into the event (e10); and

- (E3.8) Initial_SSF_Failure – This is an external event caused by the reception of SSF failure. This event causes a transition that maps into the SCSM event (E9).

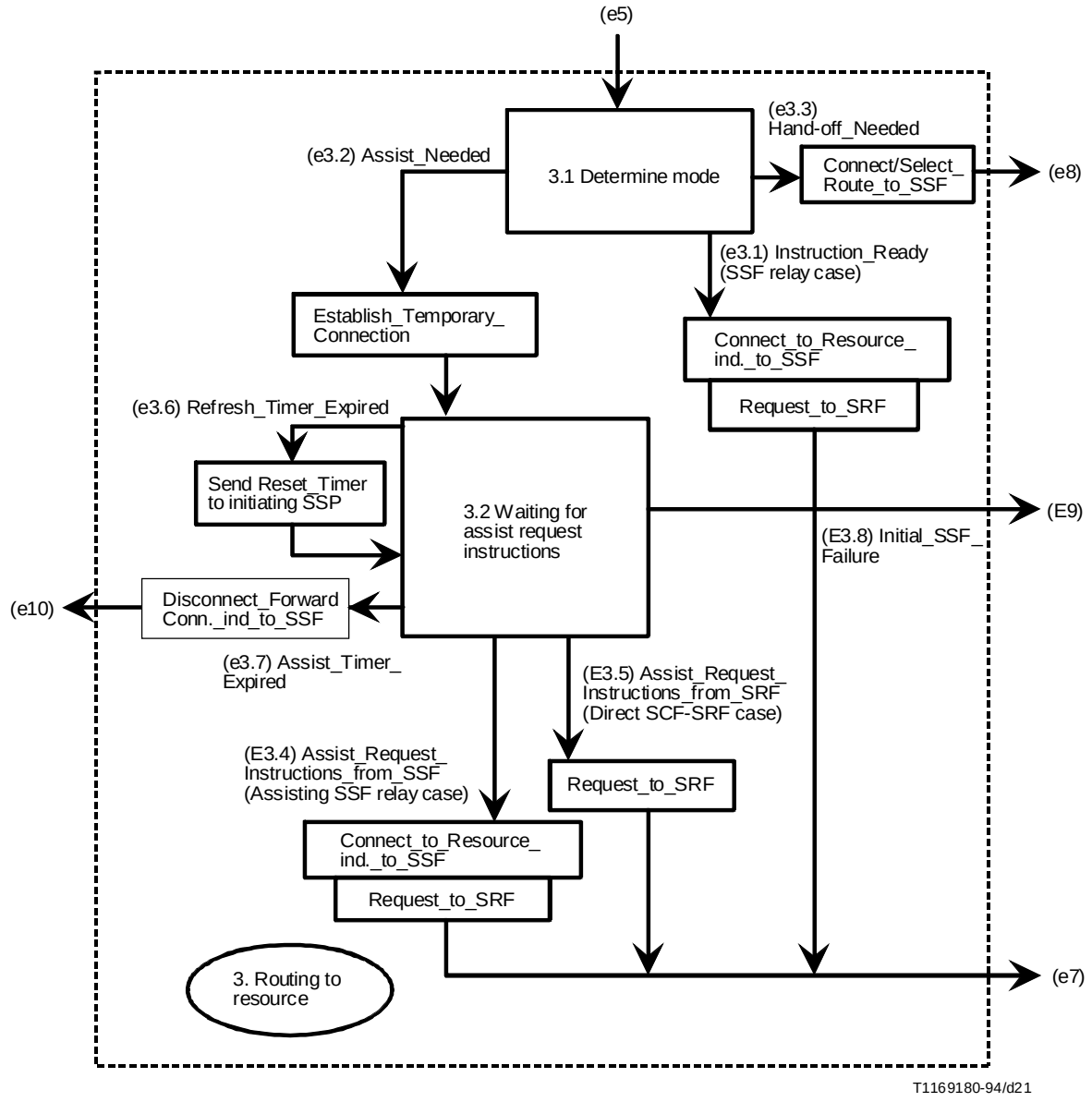


FIGURE 21/Q.1218

The State 3 FSM

3.1.2.5.4 State 4 – "User Interaction"

In this state, the SCF requests the SRF to provide user interaction (e.g. collect additional information and/or play announcements). When an interaction is finished the SCF can instruct the SSF to disconnect the bearer between SSF and SRF. Alternatively, it can send a user interaction operation to the SRF containing an indication that allows the SRF initiated disconnect.

On entering this state the SCSM resets the timer $T_{SCF\text{-}SSF}$ (if it is used).

One event causes a transition out of this state:

- (e11) Continue_SCF_Processing – In this case, the SCF has obtained all the information from the SRF, that is needed to instruct the SSF to complete the call. This event causes a transition to State 2, **Preparing SSF Instructions**.

To consider the processing of this state in more detail, it is expanded into a separate FSM depicted in Figure 22A.

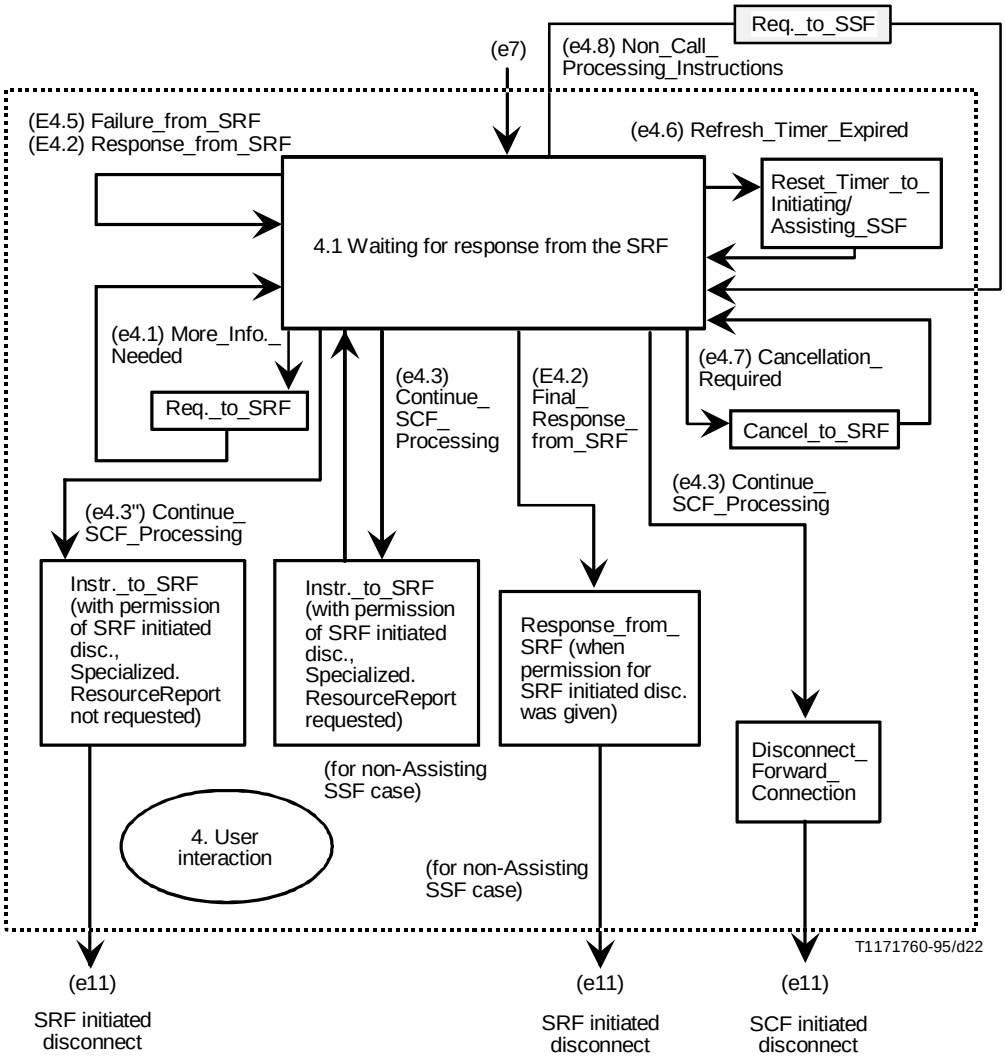


FIGURE 22A/Q.1218

The State 4 FSM

3.1.2.5.4.1 State 4.1 – "Waiting for Response from the SRF"

In this state, the SCF waits for the response to the previously sent operation and evaluates this response. The following events are considered in this state:

- (e4.1) More_Information_Needed results in issuing yet another operation to the SRF; it causes a transition back to State 4.1;

- (E.4.2) Response_from_SRF – This is an external event caused by the reception of **SpecializedResourceReport** or Return Result from **PromptAndCollectUserInformation** operation. On the receipt of this operation, the SCSM transits back to the same state;

- (E4.2′) Final_Response_from_SRF – This is an external event caused by the reception of SpecializedResourceReport opperation, as requested by the SCF, in response to the previous PlayAnnouncement or Return Result from PromptAndCollectUserInformation operation with permission of SRF-initiated disconnect. In the Initiating SSF relay case and the Direct SCF-SRF case, on the receipt of this event, the SCSM transits to the State 2, Preparing SSF Instructions. This event maps into the event (e11);

- (e4.3) Continue_SCF_Processing – This is an internal event that takes place when the SCSM finishes the User Interaction and requests the disconnection of bearer connection between the Initiating SSF and SRF by means of SCF initiated disconnect. In this case, the SCSM sends the DisconnectForwardConnection operation to the Initiating SSF and transits to the State 2, Preparing SSF Instructions. This event maps into the event (e11);

- (e4.3′) Continue_SCF_Processing – This is an internal event that takes place when the SCSM finishes the User Interaction and requests the disconnection of bearer connection between the Initiating SSF and SRF by means of SRF-initiated disconnect, while an SpecializedResourceReport operation is requested to be returned to the SCF in case an announcement is completed. In this case, the SCSM sends the PlayAnnouncement (containing a request for returning a SpecializedResourceReport operation as an indication of completion of the operation) or PromptAndCollectUserInformation operation with permission of SRF-initiated disconnect to the SRF. In the case of Assisting SSF, the SRF-initiated disconnect cannot be used. In this case, the SCSM transits back to the same state;

- (e4.3") Continue_SCF Processing – This is an internal event that takes place when the SCSM fihishes the User Interaction and requests the disconnection of bearer connection between the Initiating SSF and SRF by means of SRF-initiated disconnect, while no SpecializedResourceReport operation is requested to be returned to the SCF in case the announcement is completed. In this case, the SCSM sends the PlayAnnouncement operation (not containing a request for returning a SpecializedResourceReport operation as an indication of completion of the operation) with permission of SRF-initiated disconnect to the SRF. In the case of Assisting SSF, the SRF-initiated disconnect cannot be used. In this case, the SCSM transits to the State 2, Preparing SSF Instructions. This event maps into the event (e11);

- (E4.5) Failure_from_SRF – This is an external event caused by the reception of return error for PlayAnnouncement or PromptAndCollectUserInformation operation. In this case, the SCSM does not change its state.

- (e4.6) Refresh_Timer_Expired (OPTIONAL: see 3.1.2.5) – This is an internal event that takes place on the expiration of $T_{SCF\text{-}SSF}$. In this case, the SCSM transmits the ResetTimer operation to the Initiating/Assisting SSF, and transits back to the same state; and

- (e4.7) Cancellation_Required – This is an internal event that takes place when the SCSM cancels the previous **PlayAnnouncement** or **PromptAndCollectUserInformation** operation. In this case, the SCSM sends the Cancel operation to the Assisting SSF (SSF relay case) or the SRF (Direct SCF-SRF case), and transits back to the same state.

It should be noted that the bearer connection between the SSF and the SRF is disconnected when the SCSM exits from this state.

- (e4.8) Non_Call_Processing_Instructions – This is an internal event caused by the service logic when there is a need to send this operation to the SSF. It causes one or more of the following operations to be issued to the SSF:

  – **ApplyCharging**;
  – **FurnishChargingInformation;**
  – **RequestNotificationChargingEvent**; and
  – **SendChargingInformation**.

In this case, the SCSM remains in the same state.

3.1.2.5.5 SDF-related states

The interaction with the SDF is possible from any state of the SCF. In the following subclauses, the SDF-related states are specified. The model describes the relationship of one SCF with one SDF. If an SCF needs to access another SDF a new FSM should be instantiated.

In what follows, the states and events are enumerated independent of the rest of the SCSM; the discussion is accompanied by Figure 22B.

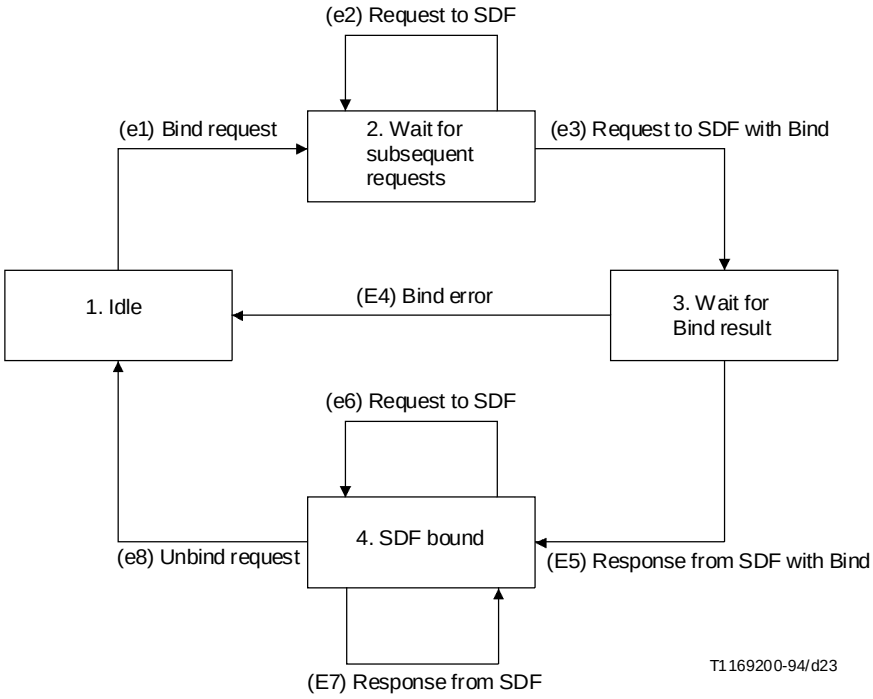


FIGURE 22B/Q.1218

The SDF-related states

3.1.2.5.5.1 State 1 – "Idle"

This state is a sub-state of any state before a request to the SDF is issued. The following event is considered in this state:

– (e1) Bind_Request – This is an internal event, caused by the need of the service logic to create an association with an SDF in order to begin accessing data. This event causes a transition to the State 2, **Wait for Subsequent Requests.**

3.1.2.5.5.2 State 2 – "Wait for Subsequent Requests"

In this state, subsequent operations to be sent with the **Bind** operation (in the same message) to the SDF are expected. The following two events are considered in this state:

- (e2) Request_to_SDF – This is an internal event, caused by the reception of an operation. The operation is buffered until the reception of a delimiter (or a timer expiration). The SCSM remains in the same state; and

- (e3) Request_to_SDF_with_Bind – This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing the argument of the **Bind** operation and other operations' arguments, if any, is sent to the SDF. This event causes a transition out of this state to the State 3, **Wait for Bind Result.**

3.1.2.5.5.3 State 3 – "Wait for Bind Result"

In this state, the SCF is waiting for the response from the SDF. Two events are considered in this state:

- (E4) Bind Error – This is an external event, caused by the reception of an error to the **Bind** operation previously issued to the SDF. This event causes a transition out of this state to State 1, **Idle**.

- (E5) Response_from_SDF_with_Bind – This is an external event, caused by the reception of a **Bind** result combined with the responses to other operations previously issued to the SDF (if any). This event causes a transition to State 4, **SDF Bound**.

3.1.2.5.5.4 State 4 – "SDF Bound"

In this state, the SCF has established an authenticated access to the SDF, and is waiting for requests to the SDF from the service logic or is waiting for responses to the operations previously issued to the SDF. Three events are considered in this state:

- (e6) Request_to_SDF – This is an internal event, caused by the need of the service logic to access data in the SDF. The SCSM remains in the same state;

- (E7) Response_from_SDF – This is an external event, caused by the reception of responses to the operations previously issued to the SDF. The SCSM remains in the same state; and

- (e8) Unbind_request – This is an internal event, caused by the need of the service logic to terminate the authenticated access to the SDF. This event causes a transition state to State 1, **Idle.**

In addition to the above model, an SDL description of SCSM is shown in Annex B.

3.1.3 SRF application entity procedures

3.1.3.1 General

This subclause provides the definition of the SRF Application Entity (AE) procedures related to the SRF-SCF interface. The procedures are based on the use of Signalling System No. 7 (SS No. 7); other signalling systems can be used.

Other capabilities may be supported in an implementation-dependent manner in the IP, SSP or SN.

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes Transaction Capabilities Application Part (TCAP) and one or more ASEs called TC-users. The following subclauses define the TC-user ASE and SACF and MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.771.

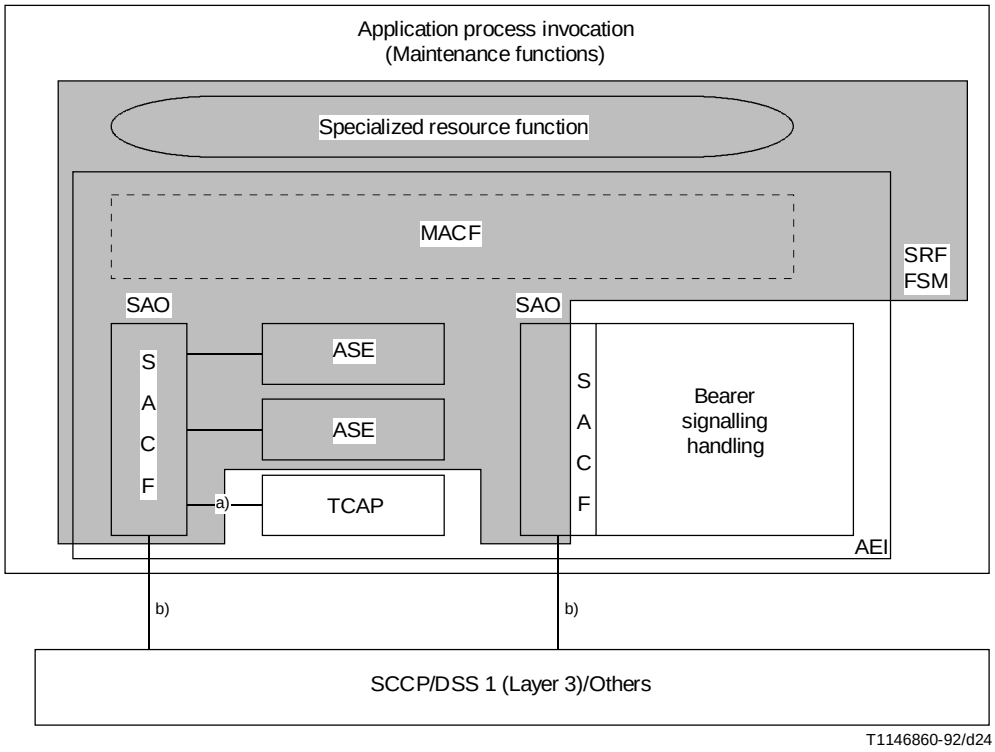
The procedure may equally be used with other message-based signalling systems supporting the application layer structures defined.

In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using TCAP services, the statements and rules contained in the detailed subclauses 3.3 and 3.4 shall be followed.

3.1.3.2 Model and interfaces

The functional model of the AE-SRF is shown in Figure 23; the ASEs interface to TCAP (to communicate with the SCF) as well as interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 23.

The interfaces shown in Figure 23 use the TC-user ASE primitives specified in Recommendation Q.771 [interface (1)] and N-Primitives specified in Recommendation Q.711 [interface (2)]. The operations and parameters of intelligent network application protocol (INAP) are defined in clause 2.



AEI Application Entity Invocation
SRF Specialized Resource Functions
FSM Finite State Machine
MACF Multiple Association Control Function
SACF Single Association Control Function
SAO Single Association Object

a) TC-primitives or Q.932-primitives.
b) N-primitives.

NOTE – The SRF-FSM includes several finite state machines.

FIGURE 23/Q.1218

Functional model of SRF-AE

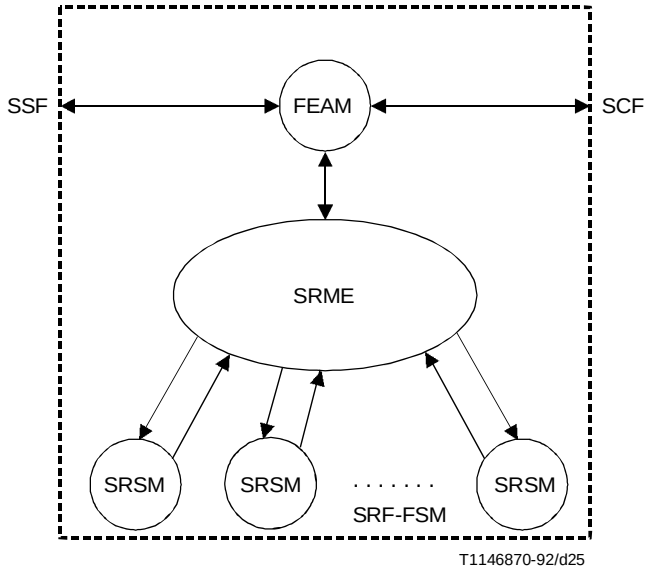
3.1.3.3 Relationship between the SRF-FSM and maintenance functions/bearer connection handling

The primitive interface between the SRF-FSM and the maintenance functions is an internal interface and is not subject for standardization in IN CS-1.

The relationship between the bearer connection handling and the SRF-FSM may be described as follows for the case of a call initiated by the SSF: when a call attempt is initiated by the SSF, an instance of an SRF-FSM is created.

The SRF-FSM handles the interaction with the SCF-FSM and the SSF-FSM.

The management functions related to the execution of operations received from the SCF are executed by the SRF Management Entity (SRME). The SRME interfaces the different SRF Call State Models (SRSM) and the Functional Entity Access Manager (FEAM). Figure 24 shows the SRF-FSM structure.



FEAM Functional Entity Access Manager
SRME SRF Management Entity
SRSM SRF Call State Model

FIGURE 24/Q.1218

SRF-FSM structure

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 explains the need for a single entity that performs the tasks of creation, invocation, and maintenance of the SRSM objects. This entity is called the SRF management entity (SRME). In addition to the above tasks, the SRME maintains the dialogues with the SCF and SSF on behalf of all instances of the SCSM. In particular, the SRME:

1) interprets the input messages from other FEs and translates them into corresponding SRSM events; and

2) translates the SRSM outputs into corresponding messages to other FEs.

NOTE – Such a request from the SCF is executed by the SCSM when it is in its State 4.

Finally, the Functional Entity Access Manager (FEAM) relieves the SRME of low-level interface functions. The FEAM functions include:

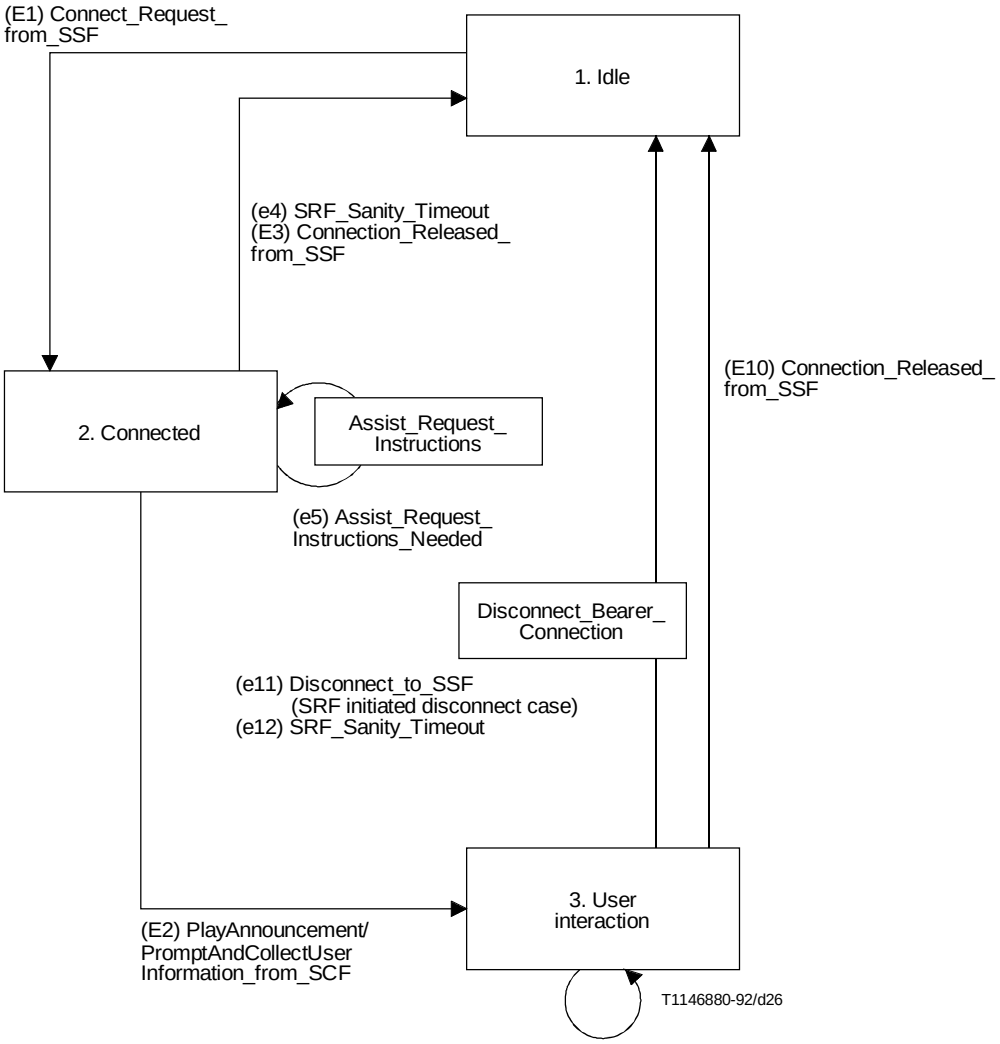
1) establishing and maintaining the interfaces to the SSF and SCF;

2) passing (and queueing when necessary) the messages received from the SSF and SCF to the SRME; and

3) formatting, queueing (when necessary), and sending messages received from the SRME to the SSF and SCF.

3.1.3.4 The SRSM

The SRSM is presented in Figure 25. In what follows, each state is described in a separate subclause together with the events that cause a transition out of this state. Finally, the outputs are presented within smaller rectangles than the states are; unlike the states and events, the outputs are not enumerated.



(E5) PlayAnnouncement/PromptAndCollectUserInformation_from_SCF
(E6) Cancel_from_SCF
(e7) SRF_Report_to_SCF
(e8) PlayAnnouncement/PromptAndCollectUserInformation_Cancelled_to_SCF
(e9) Cancel_Error_to_SCF

FIGURE 25/Q.1218

The SRSM

Each state is discussed in the following subclauses. General rules applicable to more than one state are addressed here.

One component or a sequence of components received in one or more TCAP messages may include a single operation or multiple operations, and it is processed as follows:

– The SRSM processes the operations in the order in which they are received.

– The SRSM examines subsequent operations in the sequence. When a Cancel (for PlayAnnouncement or PromptAndCollectUserInformation) operation is encountered in the sequence in state user interaction, it executes it immediately. In all other cases, the SRSM queues the operations and awaits an event (such an event would be the completion of the operation being executed, or reception of an external event).

– If there is an error in processing one of the operations in the sequence, the SRF-FSM processes the error (see below) and discards all remaining operations in the sequence.

– If an operation is not understood or is out of context (i.e. it violates the SACF rules defined by the SRSM) as described above, the SRF-FSM processes the error according to the rules given in 3.4.2 (using TC U-REJECT or the operation error UnexpectedComponentSequence).

In any state, if there is an error in a received operation, the maintenance functions are informed. Generally, the SRSM remains in the same state in which it received the erroneous operations, however different error treatment is possible in specific cases as described in 3.2; depending on the class of the operation, the error could be reported by the SRF to the SCF using the appropriate component (see Recommendation Q.774).

In any state, if the dialogue with the SCF (direct SCF-SRF case) is terminated, then the SRSM returns to idle state after ensuring that all resources allocated to the dialogues have been de-allocated. The SRF shall remain connected to the SSF as long as it has PlayAnnouncement operations active or buffered. The resources allocated to the call will be de-allocated when all announcements are completed or when the SSF disconnects the bearer connection (i.e. call party release).

In any state (except idle), if the SSF disconnects the bearer connection to the SRF before the SRF completes the user interaction, then the SRSM clears the call and ensures that all SRF resources allocated to the call have been de-allocated. Then it transits to the idle state.

The SRSM has an application timer, T_{SRF}, whose purpose is to prevent excessive call suspension time. This timer is set when the SRF sends Setup Response bearer message to the SSF (SSF relay case) or the AssistRequestInstructions operation (Direct SCF-SRF case). This timer is stopped when a request is received from the SCF. The SRF may reset T_{SRF} on transmission of the SpecializedResourceReport operation or the return result for the PromptAndCollectUserInformation operation when there is no queued user interaction operation. On the expiration of T_{SRF}, the SRSM transits to the idle state ensuring that all SRF resources allocated to the call have been de-allocated.

3.1.3.4.1 State 1 – “Idle”

The idle state represents the condition prior to, or at the completion of, an instance of user interaction. This state is entered as a result of events E3, e4, E10, e11 and e12. It is exited as a result of event E1.

– (E1) Connect_Request_from_SSF – This event corresponds to a bearer signalling connection request message from the SSF. The details of the bearer signalling state machine related to establishing the connection are not of interest to the FSM. The SRSM goes to state “Connected”;

– (E3) Connection_Released_from_SSF – This event takes place when the SRSM receives a release message from the SSF in connected state. The SRSM goes to state “Idle”;

– (e4) SRF_Sanity_Timeout – This event occurs when the SRSM has been in connected state for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/-PromptAndCollectUserInformation operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state “Idle”;

– (E10) Connection_Released_from_SSF – This event takes place when the SRSM receives a release message from the SSF in user interaction state. The SRSM goes to state “Idle”;

– (e11) Disconnect_to_SSF – This event occurs when the SCF has enabled SRF initiated disconnect by the last PlayAnnouncement/PromptAndCollectUserInformation from SCF (E2) or (E5) with the parameter disconnectFromIPForbidden. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the last SpecializedResourceReport operation to the SCF (e7). The SRSM goes to state "Idle"; and

– (e12) SRF_Sanity_Timeout – This event occurs when the SRSM has been in User interaction state for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/-PromptAndCollectUserInformation operation to execute. The SRF initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

3.1.3.4.2 State 2 – "Connected"

This state represents the condition of the SRSM when a bearer channel has been established between a user and the SRF but the initial PlayAnnouncement/PromptAndCollectUserInformation has not yet been received (e.g. when EstablishTemporaryConnection procedures are used). The method used to provide this bearer channel is not of interest in the FSM.

– (E1) Connect_Request_from_SSF – This event corresponds to a bearer signalling connection request message from the SSF in the Idle state. The details of the bearer signalling state machine related to establishing the connection are not of interest in the SRF-FSM. The SRSM goes to state "Connected".

– (E2) PlayAnnouncement/PromptAndCollectUserInformation_from_SCF – This event takes place when the first PlayAnnouncement or PromptAndCollectUserInformation operation(s) from the SCF is received. The SRSM goes to state "User interaction".

– (E3) Connection_Released_from_SSF – This event takes place when the SRF receives a release message from the SSF. The SRSM goes to state "Idle".

– (e4) SRF_Sanity_Timeout – This event occurs when the SRSM has been connected for a network-operator-defined period of time (timer T_{SRF}) without having a PlayAnnouncement/-PromptAndCollectUserInformation operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

– (e5) Assist_Request_Instructions_Needed – This event occurs when the AssistRequestInstructions operation is sent from the SRSM to the SCF in the absence of a PlayAnnouncement/-PromptAndCollectUserInformation event (E2) initiated by the presence of a PlayAnnouncement/-PromptAndCollectUserInformation operation concatenated with the setup request from SSF (E1) (Direct SCF-SRF case). No state change occurs as a result of this event.

3.1.3.4.3 State 3 – "User interaction"

The User interaction state indicates that communication is occurring between the user and the SRF via the bearer channel established at the Connected state. This state is entered as a result of event E2. It is exited as a result of events E10, e11 and e12. Events E5, E6, e7, e8 and e9 do not cause a state change. Event E5 also represents additional PlayAnnouncement/PromptAndCollectUserInformation operations which are buffered as discussed in the procedures.

– (E2) and (E5) PlayAnnouncement/PromptAndCollectUserInformation_from_SCF – This event takes place when an initial or subsequent PlayAnnouncement or PromptAndCollectUserInformation operation(s) from the SCF is received. The SRSM goes to state "User interaction" on the first (E2). The SRSM remains in state "User interaction" for subsequent (E5)s.

– (E6) Cancel_from_SCF (for PlayAnnouncement/PromptAndCollectUserInformation) – This event takes place when the corresponding PlayAnnouncement or PromptAndCollectUserInformation operation is received from the SCF. The indicated interaction is terminated if it is presently running, otherwise it is deleted from the buffer. The SRSM remains in state "User interaction".

– (e7) SRF_Report_to_SCF – This event takes place when a SpecializedResourceReport or a return result for PromptAndCollectUserInformation operation is sent to the SCF. The SRSM remains in state "User interaction".

– (e8) PlayAnnouncement/PromptAndCollectUserInformation_Cancelled_to_SCF – This event takes place when the PlayAnnouncement/PromptAndCollectUserInformation error caused by the Cancel (for PlayAnnouncement or PromptAndCollectUserInformation) operation is sent to the SCF. This event represents the successful cancellation of an active or buffered PlayAnnouncement/-PromptAndCollectUserInformation operation. The SRSM remains in state "User interaction".

– (e9) Cancel_Error_to_SCF – This event takes place when the cancel error (for PlayAnnouncement or PromptAndCollectUserInformation) is sent to the SCF. This event represents the unsuccessful cancellation of a PlayAnnouncement/PromptAndCollectUserInformation operation. The SRSM remains in state "User interaction".

– (E10) Connection_Released_from_SSF – This event takes place when the SRSM receives a release message from the SSF. The SRSM goes to state "Idle".

– (e11) Disconnect_to_SSF – This event occurs when the SCF has enabled SRF initiated disconnect with the last PlayAnnouncement/PromptAndCollectUserInformation from SCF (E2) or (E5). The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the last SpecializedResourceReport or a return result for PromptAndCollectUserInformation operation to the SCF. The SRSM goes to state "Idle".

– (e12) SRF_Sanity_Timeout – This event occurs when the SRSM has been in user interaction state for a network-operator-defined-period of time (timer T_{SRF}) without having a PlayAnnouncement/-PromptAndCollectUserInformation operation to execute. The SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system. The SRSM goes to state "Idle".

In addition to these explicitly marked transitions, failure of a user-SRF bearer connection will cause the SRSM to transit to Idle from any state. These transitions are not shown on Figure 25 for the purpose of visual clarity.

3.1.3.5 Example SRF control procedures

This subclause provides a detailed description of the SRF procedures. Arrow diagrams are used for the description of the connect, interaction with the end user, and disconnect stages.

The SRF control procedures are based on various physical allocation patterns of SRF. The various control procedures are described in this subclause in accordance with the example physical scenarios of protocol architecture in 0.2.

The service assist and handoff procedures based on the physical scenarios are also described in this subclause as examples.

Note that, through this subclause, bearer connection control signalling messages are used for explanatory purpose, and are not subject for standardization in this Recommendation. The terms used for bearer connection control signalling messages only represent the functional meaning.

3.1.3.5.1 SRF connect procedures

3.1.3.5.1.1 SRF connect physical procedures

Several procedures are required for different physical scenarios. The cases to be covered are described below and illustrated in Figure 26:

i) the IP is integrated into the SSP, or directly attached to the SSP, that is interacting with the SCP but the SCP's operations to the IP are relayed via the SSP which performs any needed protocol conversion;

ii) the IP is directly attached to the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved;

iii) the IP is integrated into another SSP, or directly attached to another SSP, than the one that is interacting with the SCP but the SCP's operations to the IP are relayed via the second SSP (called the "Assist" method), and on completion of the user interaction, control is returned to the first SSP;

iv) the IP is directly attached to a node other than the SSP that is interacting with the SCP but the SCP's operations to the IP are sent directly to the IP without SSP relaying involved (called the "Assist" method, but with a variation on the physical connectivity of the entities involved), and on completion of the user interaction, control is returned to the first SSP; and

v) the IP is attached to another SSP and on completion of the user interaction, control of the call is retained at that SSP (called the "Handoff" approach).

In each of the above cases, the operations between the SCP and the SSP may be SS No. 7 TCAP-based; the messaging between the SSP and the IP when the SSP does relaying may be DSS 1 using the facility IE (in this case, the SSP would have to do protocol conversion from SS No. 7 TCAP to DSS-1 facility IE for the operations and responses it relayed between the SCP and the IP); the direct messaging between the SCP and the IP may be SS No. 7 TCAP based; and bearer control signalling may be any system.

Each of the scenarios will now be examined using arrow diagrams.

Case i) is illustrated in Figure 27. Note that for the integrated IP/SSP, the internal activities of the node can still be modelled in this way, but the details of how this is achieved are left to the implementor. This approach makes it unnecessary for the SCP to distinguish between integrated and external but directly connected IPs. See also a note on the possibility of concatenating the first user interaction operation with the ConnectToResource operation discussed in the subclause on user interaction below. The establishment of the SCF-SRF relationship in this case is implicit.

Case ii) requires that the IP indicate to the SCP that it is ready to receive operations (see Figure 28). The establishment of the SCF-SRF relationship is explicit. Note that it is necessary to convey a correlation ID to ensure that the transaction established between the SCP and the IP can be correlated to the bearer connection set-up as a result of the SCP's preceding operation to the SSP.

Case iii) requires that a transaction be opened with the assisting SSP so that it may relay operations from the SCP to the IP (integrated or external). Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP. After the AssistRequestInstructions are received by the SCP, the procedures are the same as in case i). Figure 29 illustrates the preamble involved.

Case iv) does not require the establishment of a second transaction from the assisting exchange, hence it need not be an SSP. This then becomes a preamble to the procedure shown in Figure 28 as shown in Figure 30.

Case v) merely requires the sending of an operation to the first SSP to route the call to the handed-off SSP, and then Figure 27 applies at handed-off SSP. This is shown in Figure 31. Note that the activity at handed-off SSP represents a new interaction with the SCP and "AssistRequestInstructions" is used. Once the bearer control signalling has reached the assisting SSP, it triggers on the identity of the called facility, and initiates an interaction with the SCP that has requested the assistance. It would also be possible to trigger on other IEs such as the incoming address. The bearer control signalling must contain information to identify the SCP requesting the assistance, and a correlation ID. This information may be hidden in the address information in such a way that non-message based signalling systems may also be used to establish the bearer connection to the assisting SSP.

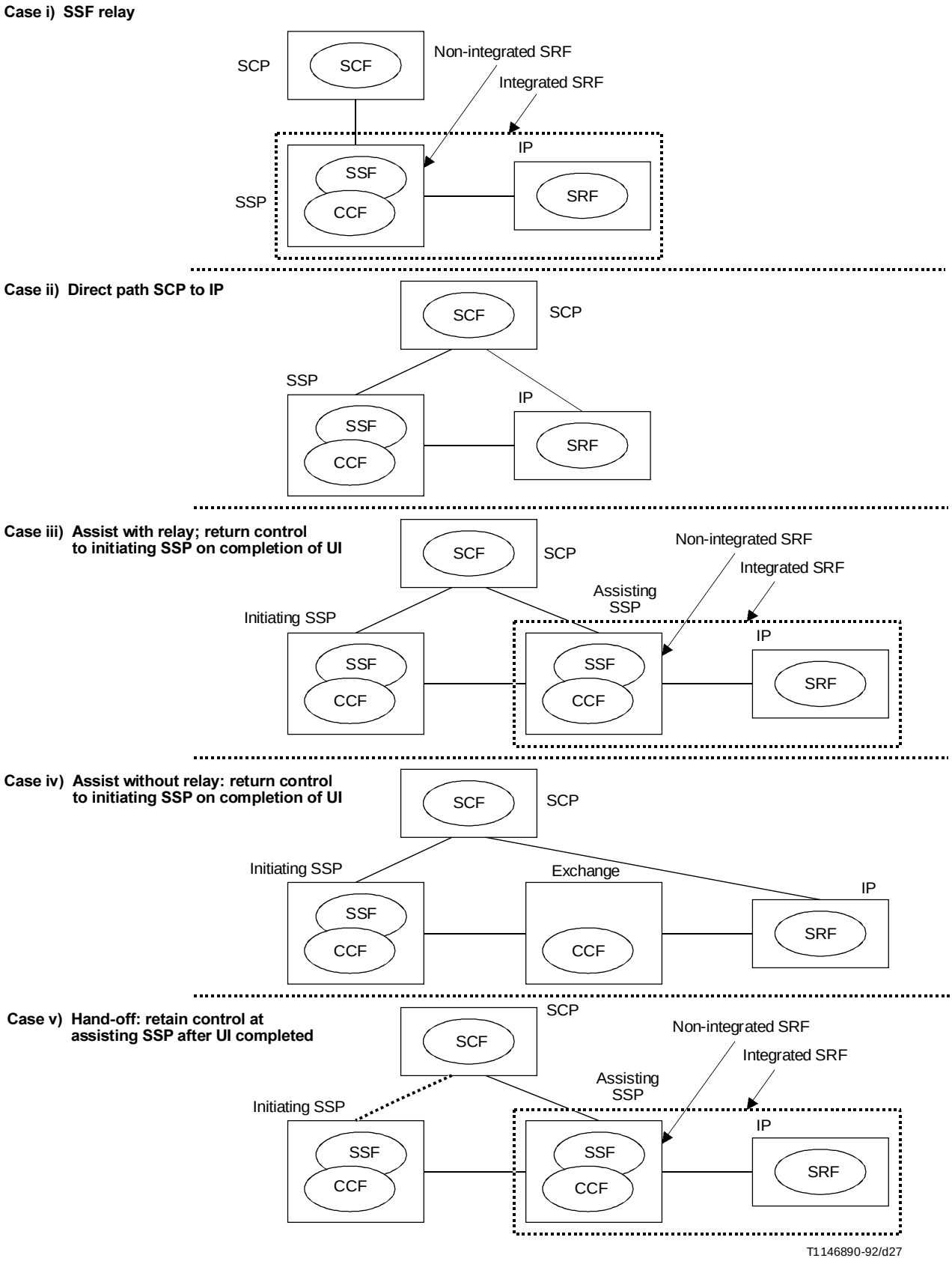


FIGURE 26/Q.1218

Physical scenarios

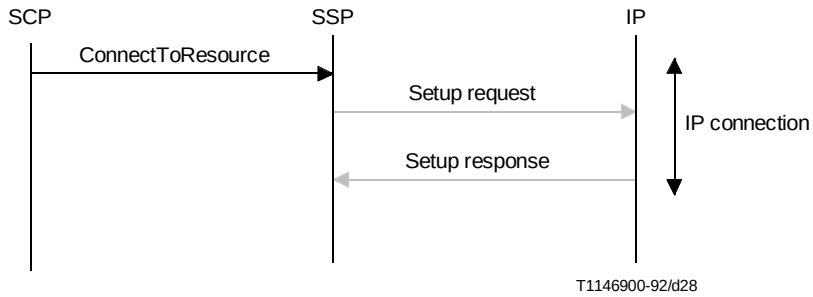


FIGURE 27/Q.1218

Connection to integrated or external IP with SSP relay of IP operations

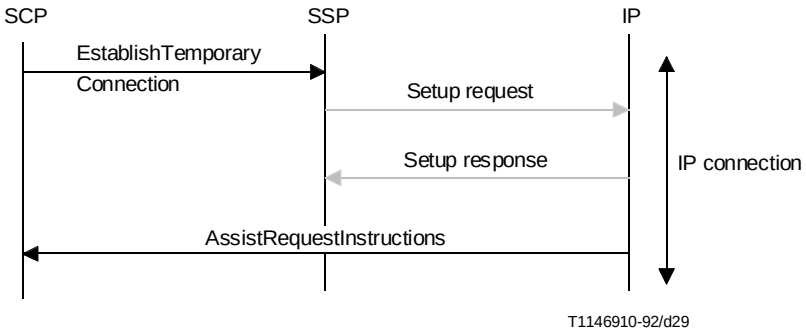


FIGURE 28/Q.1218

Connection to IP with direct link to SCP – IP initiates interaction with SCP

NOTE – Black lines indicate INAP operations, gray lines indicate DSS 1 operations.

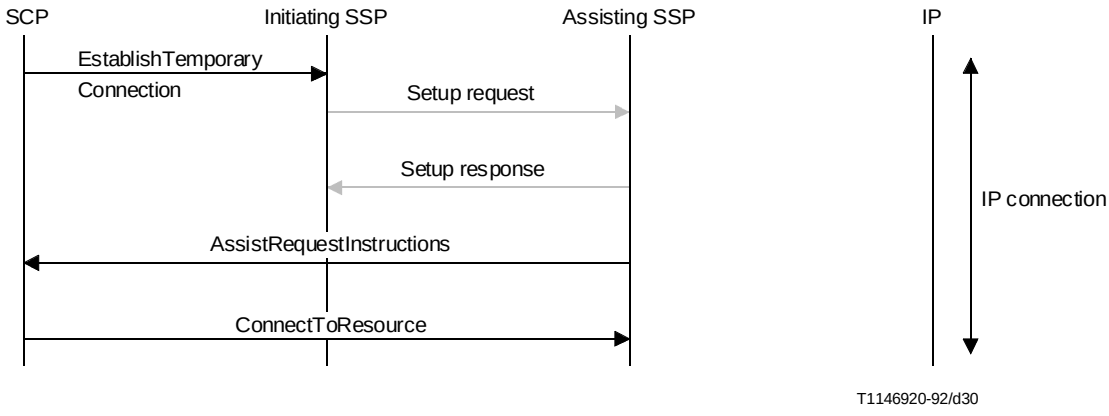


FIGURE 29/Q.1218

Preamble for assist case with integrated IP
or external IP and SSP relay of SCP-IP messages

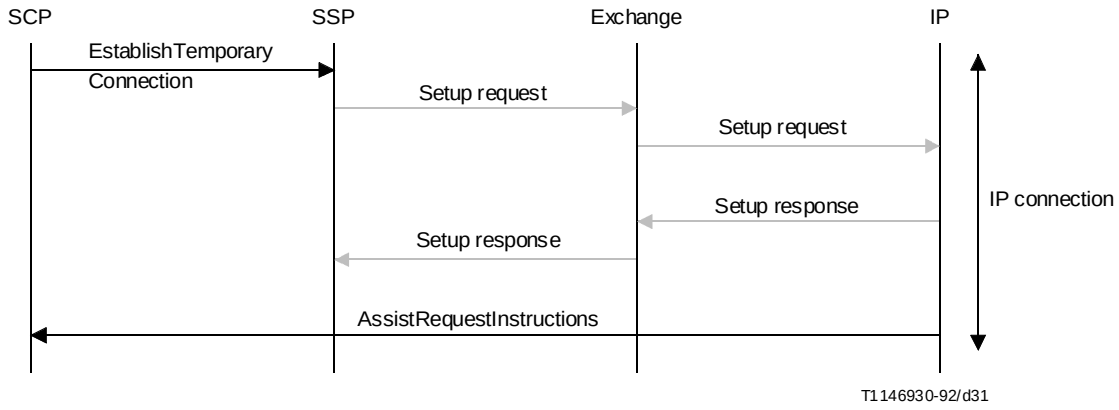


FIGURE 30/Q.1218

Preamble for assist case with external IP and direct SCP-IP messaging

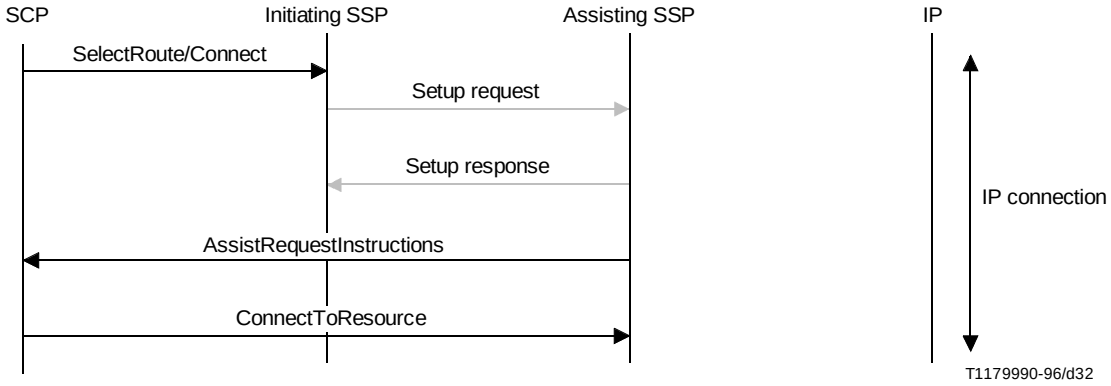


FIGURE 31/Q.1218

Preamble for hand-off case

3.1.3.5.2 SRF end user interaction procedures

The end user interaction procedures allow:

- the sending of one or multiple messages to the end user by using the PlayAnnouncement operations;
- a dialogue with the end user by using one or a sequence of PromptAndCollectUserInformation operations;
- a combination of the above; and
- cancellation of a PlayAnnouncement or PromptAndCollectUserInformation operations by using a generic cancel operation.

3.1.3.5.2.1 Play Announcement/Prompt & Collect User Information (PA/P&C)

There are only two physical scenarios for user interaction:

i) the SSP relays the operations from the SCP to the IP and the responses from the IP to the SCP (SSF relay case); and

ii) the operations from the SCP to the IP and the responses from the IP are sent directly between the SCP and the IP without involving the SSP (direct SCF-SRF case).

Case i) is illustrated in Figure 32 below.

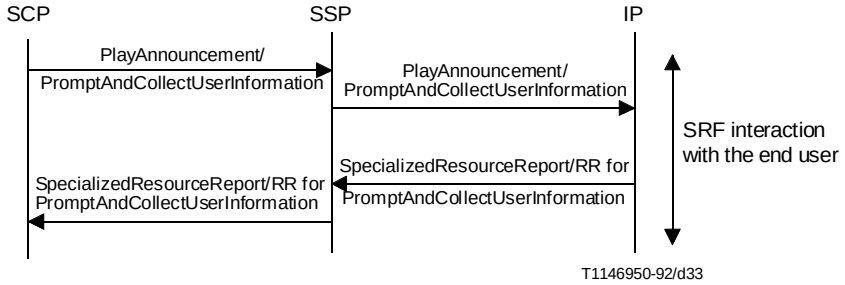


FIGURE 32/Q.1218

SSP relay of user interaction operations and responses

Case ii) is illustrated in Figure 33.

It is also necessary to consider the capability of SS No. 7 TCAP to concatenate several Invoke PDUs in one message. This capability allows, for the scenario in Figure 27, the ConnectToResource and the first PlayAnnouncement/PromptAndCollectUserInformation to be carried in one message. This has some advantages in this physical scenario, such as reduced numbers of messages, and possibly better end-user perceived performance.

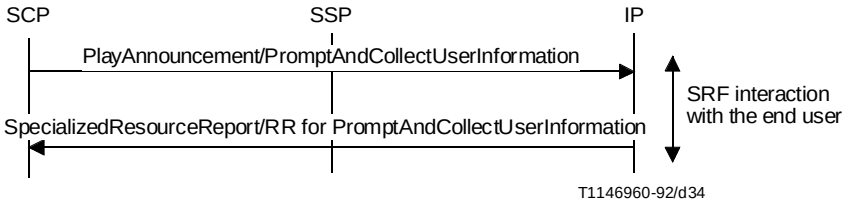


FIGURE 33/Q.1218

Direct SCF-SRF of user interaction operations and responses

3.1.3.5.3 SRF disconnection procedures

The disconnection procedures are controlled by the SCF and the procedure used is selected based on the needs of the service being executed. The bearer disconnection procedure selected by the SCF is to either allow the SRF to disconnect on completion of user interaction, or to have the SCF explicitly order the SSF to disconnect.

SRF disconnect does not cause disconnection by the SSF-CCF back to the end user terminal unless the transaction with the SCF has been terminated, indicating the user interaction completed the call. The SSF-CCF recognizes that a connection to an SRF is involved because the operations from the SCF for this purpose are distinct from the operations that would be used to route the call towards a destination. There is no impact on bearer signalling state machines as a result of this since incoming and outgoing bearer signalling events are not simply transferred to each other, but rather are absorbed in call processing, and regenerated as needed by call processing. Therefore, to achieve the desired functionality, call processing need simply choose not to regenerate the disconnect in the backward direction. Figure 34 illustrates this concept.

As for the SRF connection procedures, the SRF disconnection is affected by the physical network configuration.

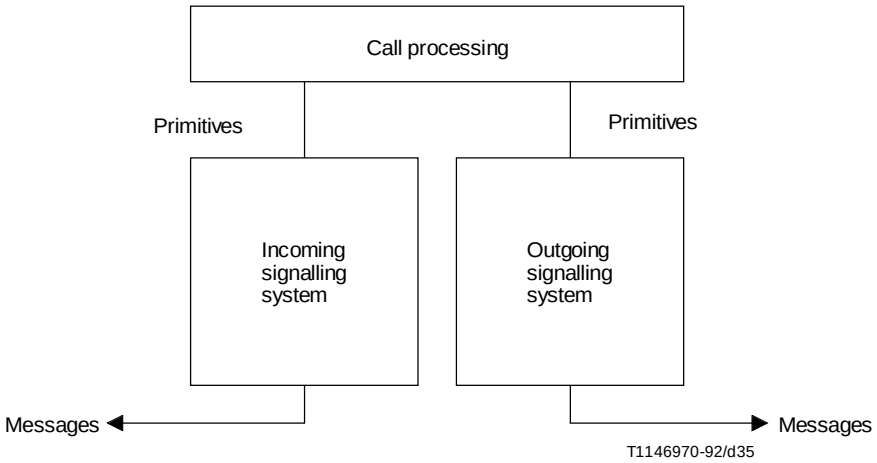


FIGURE 34/Q.1218

Relationship of incoming and outgoing signalling systems to call processing

In order to simplify the interface between the SCF and the SRF, a number of assumptions are made. The assumptions, and the resulting rules, result in unambiguous procedures from both the SCF and the SRF points of view. The rules, presented below, refer to the SRF originated disconnect, or "SRF Initiated Disconnect", and to the SCF originated disconnect, or "SCF Initiated Disconnect". While other scenarios are possible, they are not included because they either duplicate the functionality presented below or they otherwise do not add value from a service perspective.

1) If a series of PlayAnnouncement/PromptAndCollectUserInformation operations are to be executed by the same SRF, then SRF disconnect is inhibited for all but the last and may be inhibited on the last PlayAnnouncement/PromptAndCollectUserInformation. When a subsequent PlayAnnouncement/-PromptAndCollectUserInformation is received, it is buffered until the completion of any preceding PlayAnnouncement/PromptAndCollectUserInformation.

2) A generic cancel operation terminates the indicated PlayAnnouncement/-PromptAndCollectUserInformation if it is being executed by the SRF, but does not disconnect the SRF. If the cancel operation is for a buffered PlayAnnouncement/PromptAndCollectUserInformation, that PlayAnnouncement/PromptAndCollectUserInformation is discarded, but the current and any buffered PlayAnnouncement/PromptAndCollectUserInformations are executed. An SRF interacts with one user only and therefore cancelling a PlayAnnouncement/PromptAndCollectUserInformation only affects the user to which the SRF is connected.

3) The SCF must either explicitly order "Disconnect" or enable SRF initiated disconnect at the end of the PlayAnnouncement/PromptAndCollectUserInformation. An SRF left connected without a PlayAnnouncement/PromptAndCollectUserInformation to execute may autonomously disconnect if it has not received any PlayAnnouncement/PromptAndCollectUserInformation operations within a defined time limit. This could occur, for example, after an EstablishTemporaryConnection which is not followed within a reasonable time period with a PlayAnnouncement/PromptAndCollectUserInformation operation. This sanity timing value will depend on the nature of the interaction the SRF supports and should be selected by the network operator accordingly.

4) When SRF initiated disconnect is enabled in a PlayAnnouncement/PromptAndCollectUserInformation, then the SRF must disconnect on completion of the user interaction.

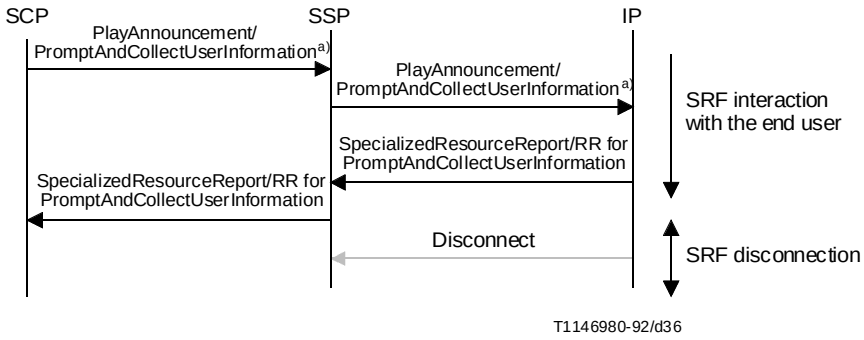
5) When SRF initiated disconnect is not enabled, the SCF must ask the SRF to inform it of the completion of the user interaction using the SpecializedResourceReport operation for "announcement complete" or using the return result for the PromptAndCollectUserInformation operation.

6) If the user disconnects, the SRF is disconnected and the SSF releases resources and handles the transaction between the SSF and the SCF as specified in Recommendation Q.1214 and in this Recommendation. The SRF discards any buffered operations and returns its resources to idle. The relationship with the SCF is terminated.

7) When the SCF explicitly orders the SSF to disconnect by "DisconnectForwardConnection" operation, the SSF releases the bearer connection to the SRF, and returns to the "waiting for instructions" state. No operation reporting SRF disconnect from the SSF to the SCF is required.

3.1.3.5.3.1 SRF initiated disconnect

The SRF disconnect procedure is illustrated in Figure 35. The SRF disconnect is enabled by the SCF within a PlayAnnouncement/PromptAndCollectUserInformation operation. When the SRF receives a PlayAnnouncement/-PromptAndCollectUserInformation enabling disconnection, it completes the dialogue as instructed by the PlayAnnouncement/PromptAndCollectUserInformation, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The SSF-CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The SSF returns to the "waiting for instructions" state and executes any buffered operations. In the handoff case, the SSP shown in Figure 35 is the "handed-off" SSP.



a) Disconnect from SRF is not forbidden.

FIGURE 35/Q.1218

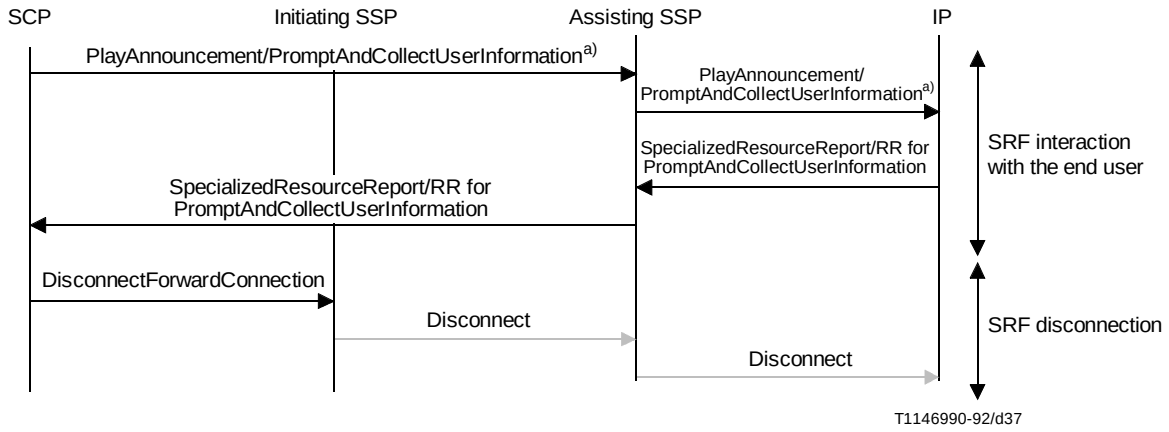
SCF disconnect for local, embedded and hand-off scenarios

For the assisting SSF case, the SRF initiated disconnect procedures are not used because the assisting SSF remains in the "waiting for instructions" state and does not propagate the disconnection of the bearer connection to the Initiating SSF. The SCF initiated disconnect procedures described in the following subclause are used for the assisting SSF case.

For the direct SCF-SRF case, the procedures also work in the same manner. The SRF disconnect is enabled by the SCF within a PlayAnnouncement/PromptAndCollectUserInformation operation. When the SRF receives a PlayAnnouncement/PromptAndCollectUserInformation enabling disconnection, it completes the dialogue as instructed by the PlayAnnouncement/PromptAndCollectUserInformation, and then initiates the SRF initiated disconnection using the applicable bearer control signalling. The Initiating SSF-CCF knows that it is an SRF disconnecting and does not continue clearing the call toward the end user. The Initiating SSF returns to the "waiting for instructions" state and executes any buffered operations.

3.1.3.5.3.2 SCF initiated disconnect

The SCF initiated disconnect procedure is illustrated in Figure 36. Bearer messages are shown in gray. The figure shows only the assisting SSF case, and the direct SCF-SRF case is not shown. To initiate the SCF initiated disconnection of the SRF, the SCF must request and receive a reply to the last PlayAnnouncement/PromptAndCollectUserInformation operation requested. The SpecializedResourceReport operation contains an "announcement complete" and return result for PromptAndCollectUserInformation contains "collected information."



a) Disconnect from SRF forbidden.

FIGURE 36/Q.1218

SCF initiated disconnect for assist scenario

The SCF initiated disconnect uses an operation called DisconnectForwardConnection. Once the DisconnectForwardConnection is received by the SSF, it will initiate a "release of bearer channel connection" between the physical entities containing the SSF and SRF, using applicable bearer control signalling. Since the SCF (which initiates the disconnect), the SSF (which instructs bearer signalling to disconnect) and the SRF (which receives disconnect notification via bearer signalling) are aware that disconnect is occurring, they are synchronized. Therefore, a "pre-arranged" end may be used to close the transaction. This does not preclude the use of explicit end messages for this purpose.

For assisting SSF case, the initiating SSP, on receipt of the DisconnectForwardConnection from the SCP, disconnects forward to the assisting SSP, and this disconnection is propagated to the IP. The initiating SSP, knowing that the forward connection was initiated as the result of an EstablishTemporaryConnection, does not disconnect back to the user but returns to the "waiting for instructions" state.

3.1.3.5.4 Examples illustrating complete user interaction sequences

The following figures and their accompanying tables provide examples of complete sequences of user interaction operations covering the three stages:

- connect the SRF and the end user (bearer connection) and establish the SCF-SRF relationship;
- interact with the end user;
- disconnect the SRF and the end user (bearer connection) and terminate the SCF-SRF relationship.

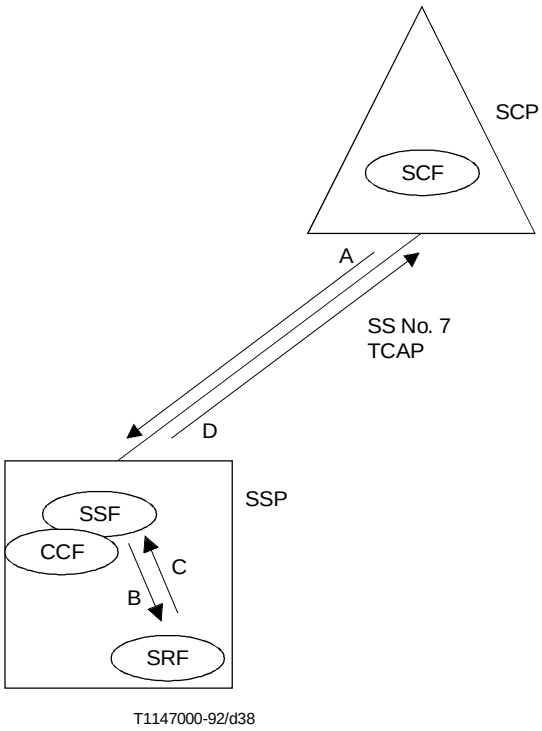


FIGURE 37/Q.1218

SSP with integrated SRF

In Figure 37, the SSP with an integrated (or embedded) SRF, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA/P&C	ConnectToResource; PlayAnnouncement/ PromptAndCollectUserInformation Setup; PlayAnnouncement/ PromptAndCollectUserInformation	A B
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation SpecializedResourceReport/RR for PromptAndCollectUserInformation	A then B C then D
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation Disconnect	C then D C (intra-SSP bearer control)
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation DisconnectForwardConnection Disconnect	C then D A B (intra-SSP bearer control)

A simple extension to this integrated case is the configuration where the SRF is located in an intelligent peripheral locally attached to the SSP. The SCP-IP operations are relayed via the SSF in the SSP. This is depicted in Figure 38.

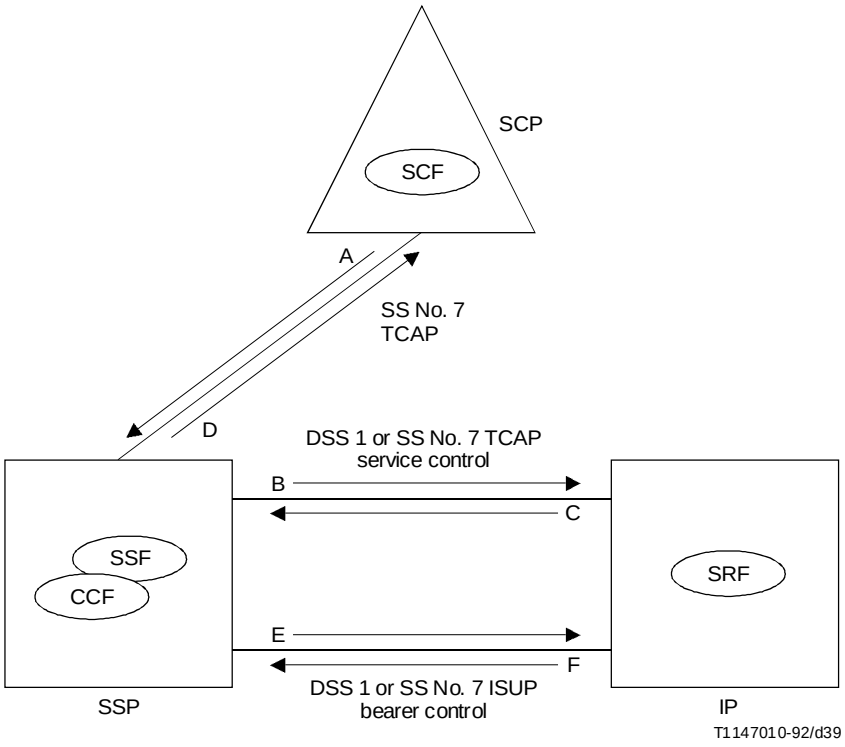


FIGURE 38/Q.1218

SSP relays messages between SCP and IP

The procedural scenarios for this relay SSF with an IP (see Figure 38) can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource and first PA/P&C	ConnectToResource; PlayAnnouncement/ PromptAndCollectUserInformation *If DSS 1 used:* Setup; PlayAnnouncement/ PromptAndCollectUserInformation *If SS No. 7 used:* IAM PlayAnnouncement/ PromptAndCollectUserInformation	A E and B (Facility IE) E B
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation SpecializedResourceReport/RR for PromptAndCollectUserInformation	A then B C then D
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C then D F F
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C then D A E E

In some cases, the IP may have an SS No. 7 or other interface to the controlling SCP. This case is shown in Figure 39. Note that the SCP must correlate two transactions to coordinate the activities.

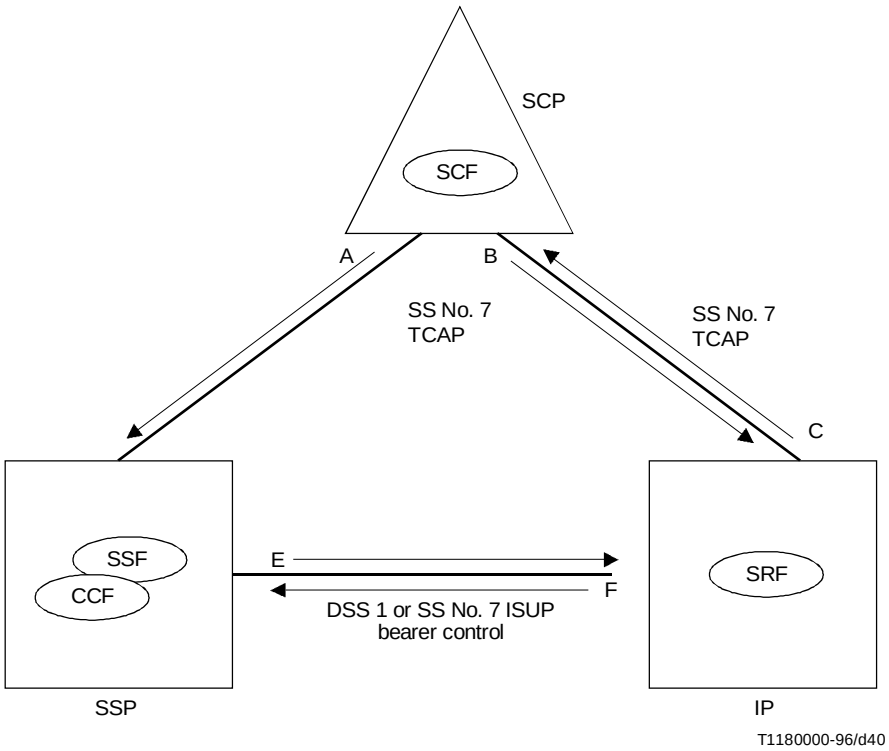


FIGURE 39/Q.1218

Direct SCP-IP information transfer

In Figure 39, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Connect to resource	EstablishTemporaryConnection *If DSS 1 used:* Setup AssistRequestInstructions *If SS No. 7 used:* IAM AssistRequestInstructions	A E C E C
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation SpecializedResourceReport/RR for PromptAndCollectUserInformation	B C
SRF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C F F
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	C A E E

The assisting SSF scenario involves straightforward procedural extensions to the basic cases shown above. One mapping of the assisting SSF case is shown in Figure 40. In this case, SRF initiated disconnect cannot be used. Other physical mappings can be derived as described in the text following the figure and its accompanying table.

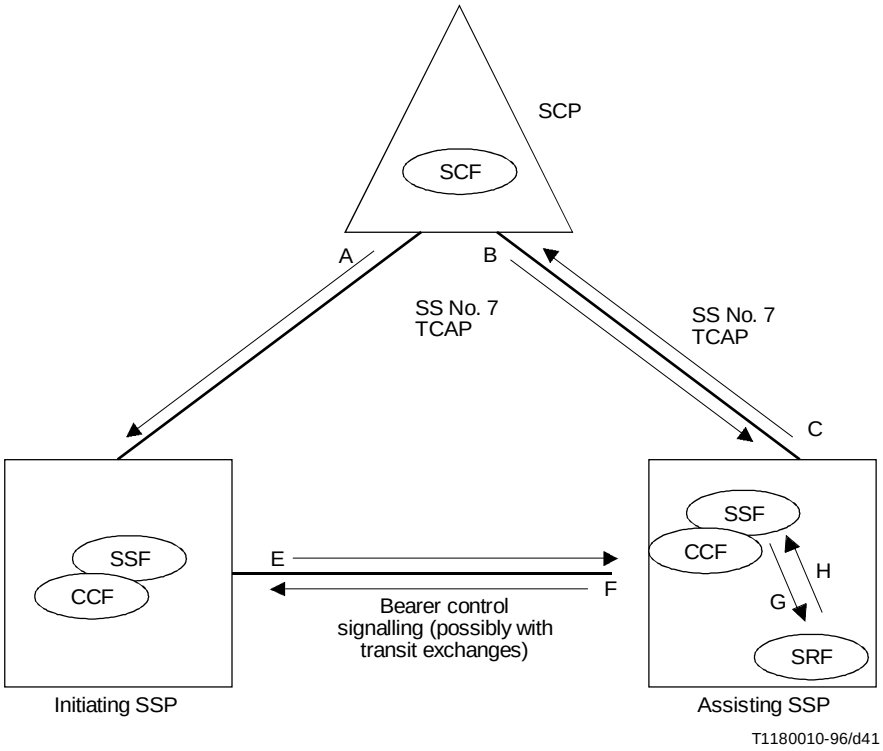


FIGURE 40/Q.1218

SSP assist (relay SSP)

Note that the integrated SRF and SSF relay case requires a transaction between the SCP and the assisting SSP (see Figure 40) but the SCP direct case does not since the transaction is directly between the SCP and the IP connected to the remote exchange. In the latter case, any transit exchanges, including the one the IP (SRF) is connected to, are transparent to the procedures.

Note also that the SCP must again correlate two transactions.

In Figure 40, the procedural scenarios can be mapped as follows:

Procedure name	Operations	Protocol flows
Assist preamble	EstablishTemporaryConnection *If DSS 1 used:* Setup AssistRequestInstructions ConnectToResource Setup ResetTimer *If SS No. 7 used:* IAM AssistRequestInstructions ConnectToResource Setup ResetTimer	A E C B G A E C B G A
User interaction	PlayAnnouncement/ PromptAndCollectUserInformation SpecializedResourceReport/RR for PromptAndCollectUserInformation	B then G H then C
SCF initiated disconnect	SpecializedResourceReport/RR for PromptAndCollectUserInformation DisconnectForwardConnection *If DSS 1 used:* Disconnect *If SS No. 7 used:* Release	H then C A E and G (intra-SSP bearer ctrl) E and G (intra-SSP bearer ctrl)

Note that the assisting SSP case shown in Figure 40 can be generalized to cover both the case where the SRF is embedded in assisting SSP (as shown), and the case where the SRF is locally connected to assisting SSP. In this latter case, the SRF communication (protocol flows B, C, G and H) would conform to the physical scenario shown in Figure 38.

The service handoff scenario can similarly be viewed as a sequence consisting of an IN service to route a call from one SSP to another, followed by any one of the previously described physical user interaction scenarios. For describing this scenario Figure 40 can be used also.

3.1.3.5.4.1 Message sequences for service assist

The following subclause provides additional details on the message sequences for the service assist procedure in Figure 40:

1) The SCP, during the processing of a request for instruction, determines that resources remote from the initiating SSP are required and that call processing will continue from the initiating SSP after the remote resources have been used (e.g. the call will be completed to a destination address after information is collected from the calling party). An EstablishTemporaryConnection operation containing the address of the assisting SSP (for routing the call), the ScfID and the CorrelationID (both used for the assisting SSP to establish communication back to the SCP) is sent to the initiating SSP. The EstablishTemporaryConnection is used instead of a regular Connect operation because of the nature of the connection to the assisting SSP. The initiating SSP must be aware that the SCP will ask it to continue in the processing of the call at some point in the future.

NOTE 1 – The ScfID and CorrelationID may be included in the routing address of the assisting SSP.

Protocol Flow A

2) The initiating SSP routes the call to the assisting SSP. The ScfID and CorrelationID are sent to the assisting SSP. Existing in-band signalling and SS No. 7 information elements (e.g. routing number) could be used to transport this information. The transport mechanism used to send this information between SSPs is independent of the service assist control procedures between the SCF and SSF.

Protocol Flow E

3) The assisting SSP uses an AssistRequestInstructions operation to establish communication with the SCP. The CorrelationID is sent in the AssistRequestInstructions to allow the SCP to correlate two transactions.

Protocol Flow C

4) The SCP sends instructions to the assisting SSP based on service logic control.

Protocol Flow B

5) The SCP may need to generate reset timer events to the initiating SSP so that it does not time out the call.

Protocol Flow A

NOTE 2 – The usage of ResetTimer operation is optional.

6) When resource functions have been completed, a DisconnectForwardConnection operation is sent to the initiating SSP. This indicates, that the temporary connection to the assisting SSP has to be disconnected.

Protocol Flow A

7) The initiating SSP sends a message via bearer control signalling to the assisting SSP to close the "assist" transaction.

Protocol Flow E

8) The call control returns to the initiating SSP.

3.1.3.5.4.2 Message sequences for handoff

The following subclause outlines message sequences for the handoff procedure using the protocol flows shown in Figure 40:

1) The SCP, during the processing of a request for instruction, determines that resources remote from the initiating SSP are required and that call processing need not continue from the initiating SSP after the remote resources have been used (e.g. a terminating announcement will be played). A Connect operation containing the address of the assisting SSP (for routing the call), the ScfID and the CorrelationID (both used for the assisting SSP to establish communication back to the SCP) is sent to the initiating SSP.

NOTE – The ScfID and CorrelationID may be included in the routing address of the assisting SSP.

Protocol Flow A

2) The initiating SSP routes the call to the assisting SSP. The ScfID and CorrelationID are sent to the assisting SSP. Existing in-band signalling and SS No. 7 information elements (e.g. routing number) could be used to transport this information. The transport mechanism used to send this information between SSPs is independent of the service assist control procedures between the SCF and SSF.

Protocol Flow E

3) The assisting SSP uses an AssistRequestInstructions operation to establish communication with the SCP. The CorrelationID is sent in the AssistRequestInstructions to allow the SCP to correlate two transactions. The AssistRequestInstructions is used instead of a regular request instruction (InitialDP or DP-specific operation) because the SCP must associate the AssistRequestInstructions from the assisting SSP-IP with an already active dialogue the SCP has with another SSP.

Protocol Flow C

4) The SCP sends instructions to the assisting SSP based on service logic control.

Protocol Flow B

5) The call control remains at the assisting SSP.

The same service assist and handoff procedures can be reused for a direct link to an IP in this and future capability sets.

3.1.4 SDF application entity procedures

3.1.4.1 General

This subclause provides the definition of the SDF Application Entity (AE) procedures related to the SDF-SCF interface. The procedures are based on the use of Signalling System No. 7 (SS No. 7).

Other capabilities may be supported in an implementation-dependent manner in the SCP, SDP, SSP, AD or SN.

The AE, following the architecture defined in Recommendations Q.700, Q.771 and Q.1400, includes Transaction Capabilities Application Part (TCAP) and one or more ASEs called TC-users, which are based on the Directory (X.500-Series Recommendations). The following subclauses define the TC-user ASE and SACF & MACF rules, which interface with TCAP using the primitives specified in Recommendation Q.771.

The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed subclauses 3.3 and 2.2.2.2 shall be followed.

3.1.4.2 Model and interfaces

The functional model of the AE-SDF is shown in Figure 41; the ASEs interface to TCAP to communicate with the SCF, and interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 41.

The interfaces shown in Figure 41 use the TC-user ASE primitives specified in Recommendation Q.771 [interface 1)] and N-Primitives specified in Recommendation Q.711 [interface 2)]. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clause 2.

An instance of SDF-FSM may be created if IN call handling is received from the SCF.

The SDF-FSM handles the interaction with the SCF-FSM.

3.1.4.3 Relationship between the SDF-FSM and the maintenance function

The primitive interface between the SDF-FSM and the maintenance functions is an internal interface and is not subject for standardization in IN CS-1.

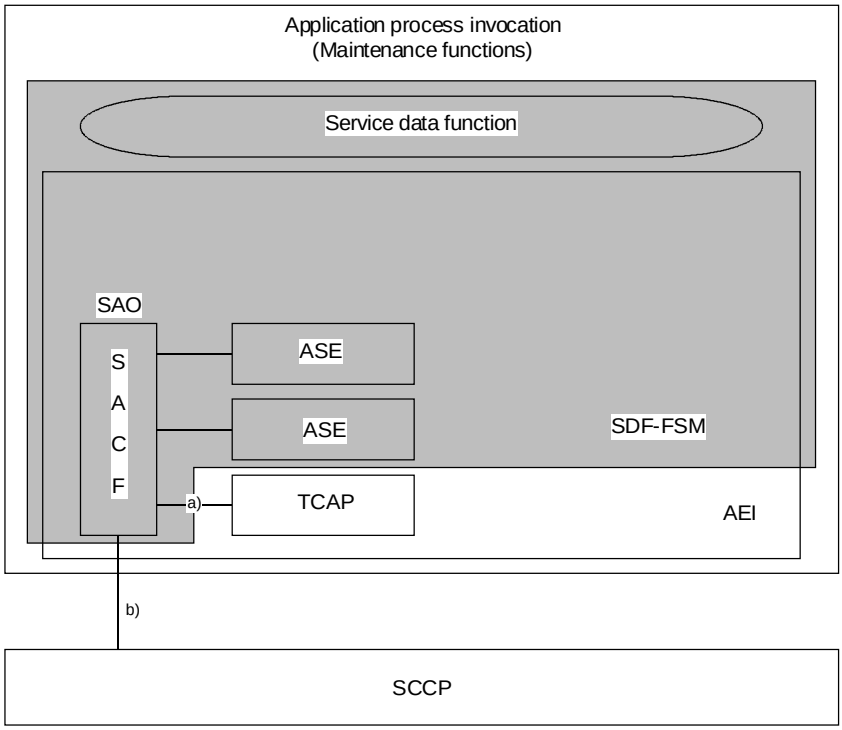
3.1.4.4 SDF state transition model

As far as IN CS-1 is concerned, the SDF's job is to (synchronously) respond to every request from the SCF after the Bind procedure. The respective Finite State Machine (FSM) is depicted in Figure 42.

Each state is discussed in one of the following subclauses.

General rules applicable to more than one state are as follows:

In any state, if the dialogue with the SCF is terminated, then the SDF-FSM returns to **Idle** state after ensuring that any resources allocated to the call have been de-allocated.



AEI Application entity invocation
SDF Service data function
FSM Finite state machine
SACF Single association control function
SAO Single association object

a) TC-primitives or Q.932-primitives.
b) N-primitives.

NOTE – The SDF FSM includes several finite state machines.

FIGURE 41/Q.1218

Functional model of SDF-AE

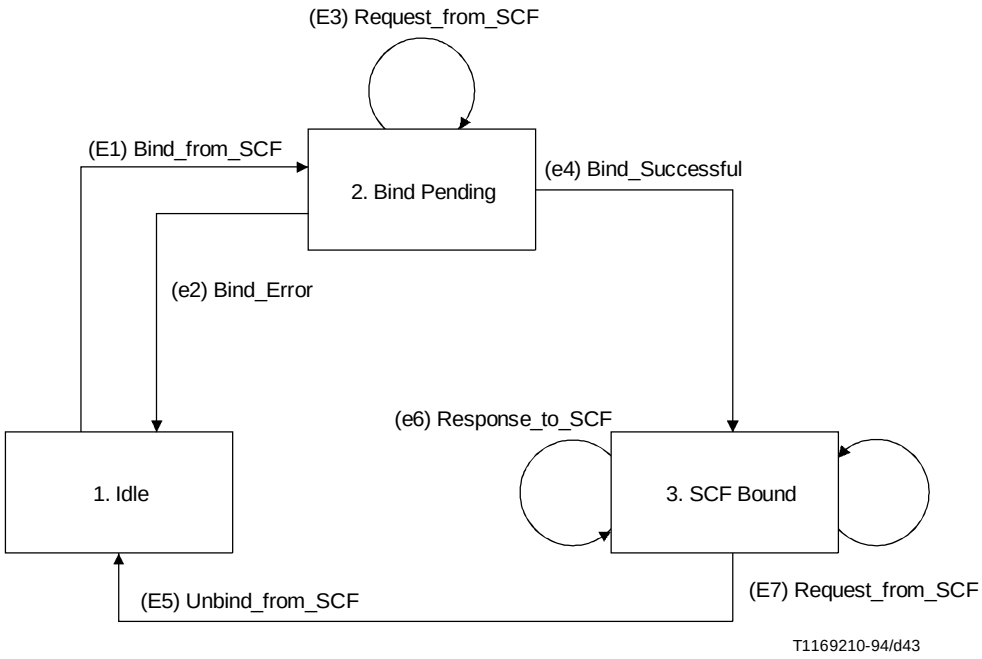


FIGURE 42/Q.1218

The SDF-FSM

3.1.4.4.1 State 1 – "Idle"

The only event accepted in this state is:

– (E1) Bind_from_SCF – This is an external event caused by the reception of Bind operation from the SCF. This event causes a transition out of this state to State 2, **Bind Pending**.

3.1.4.4.2 State 2 – "Bind Pending"

In this state, a Bind request has been received from the SCF. The SDF is performing the SCF access control procedures behind the Bind operation (e.g. access authentication). There may also be a case such that the Bind operation is a dummy one. Then, the access authentication is not required. Three events are considered in this state:

– (e2) Bind_Error – This is an internal event, caused by the failure of the Bind operation previously issued to the SDF. This event causes a transition out of this state to State 1, **Idle** and a Bind error is returned to the invoking SCF;

– (E3) Request_from_SCF – This is an external event, caused by the reception of operations before the result from the Bind operation is determined. The SDSM remains in the same state; and

– (e4) Bind_Successful – This is an internal event, caused by the successful completion of the Bind operation previously issued to the SDF. This event causes a transition out of this state to State 3, **SCF Bound**.

3.1.4.4.3 State 3 – "SCF Bound"

In this state, the access of the SCF to the SDF was authorized and operations coming from the SCF are accepted. Besides waiting for requests from the SCF, the SDF can send in that state responses to previously issued operations. Three events are considered in this state:

– (E5) Unbind_from_SCF – This is an external event, caused by the reception of the Unbind operation from the SCF. The SCF-SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1, **Idle**;

– (e6) Response_to_SCF – This is an internal event, caused by the completion of the operations previously issued by the SCF. Responses are sent to the SCF. The SDSM remains in the same state; and

– (E7) Request_from_SCF – This is an external event, caused by the reception of a request from the SCF to the SDF. The SDSM remains in the same state.

In addition to this finite state machine, an SDL description of SDSM is shown in the Annex B.

3.2 Error procedures

This subclause defines the generic error procedures for the IN CS-1 INAP. The error procedure descriptions have been divided in two subclauses, 3.2.1 listing the errors related to INAP operations and 3.2.2 listing the errors related to error conditions in the different FEs which are not directly related to the INAP operations.

3.2.1 Operation related error procedures

The following subclauses define the generic error handling for the operation related errors. The errors are defined as operation errors in 2.1.2. The TCAP services which are used for reporting operation errors are described in 3.4.

Errors which have a specific procedure for an operation are described in 3.3 with the detailed procedure of the related operation.

All errors, which can be detected by the ASN.1 decoder, already may be detected during the decoding of the TCAP message and indicated by the TC error indication "MistypedParameter" in the TC-U-Reject.

3.2.1.1 Attribute error

3.2.1.1.1 General description

3.2.1.1.1.1 Error description

This error is sent by the SDF to the SCF to report an attribute related problem. The conditions under which an attribute error is to be issued are defined in 12.4/X.511 (1993).

3.2.1.1.1.2 Argument description

The attribute error parameter and problem codes are specified in 12.4/X.511 (1993).

3.2.1.1.2 Operations SCF-> SDF

– Search;

– Add Entry;

– Modify Entry.

Procedure at Invoking Entity (SCF)

A) Sending Operation

Precondition:	SCSM state 4	SDF Bound or
	SCSM state 2	Wait for subsequent requests
Postcondition:	SCSM state 4	SDF Bound or
	SCSM state 2	Wait for subsequent requests

B) Receiving Error

Precondition:	SCSM state 4	SDF Bound
Postcondition:	SCSM state 4	SDF Bound

Error Procedure is dependent on the Service Logic. If the SCF is able to change the request, it can do another SDF query, otherwise the service processing should be terminated.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition:	SDF-FSM state 3	SCF Bound
Postcondition:	SDF-FSM state 3	SCF Bound

B) Returning Error

Precondition:	SDF-FSM state 3	SCF Bound
Postcondition:	SDF-FSM state 3	SCF Bound

The SDF could not perform the operation due to an attribute problem and therefore sends an Attribute error to the SCF. After returning the error, no further error treatment is performed.

3.2.1.2 Cancelled

3.2.1.2.1 General description

3.2.1.2.1.1 Error description

The Error "Cancelled" 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.

3.2.1.2.2 Operations SCF->SRF

– PlayAnnouncement

– PromptAndCollectUserInformation

Procedures at Invoking Entity (SCF)

A) Sending Cancel

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

The SCF sends a Cancel after a Play Announcement or PromptAndCollectUserInformation has been sent. The SCF remains in the same state.

B) Receiving Cancelled Error

Precondition:	SCSM state any	Service Logic dependent.
Postcondition:	SCSM state any	Service Logic dependent.

After sending a Cancel operation the Service Logic may continue (e.g. sending more PlayAnnouncement or PromptAndCollectUserInformation or a DisconnectForwardConnection). The Cancelled Error can therefore be received in any state. The treatment is Service Logic dependent.

Procedures at Responding Entity (SRF)

A) Receiving Cancel

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

The indicated PlayAnnouncement or PromptAndCollectUserInformation is terminated if it is presently executing or deleted from the buffer. If the indicated PlayAnnouncement or PromptAndCollectUserInformation is already executed this causes a failure ("CancelFailed").

B) Sending Cancel Error

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

After returning the "Cancelled" Error the SRF stays in the same state. The execution of the indicated PlayAnnouncement or PromptAndCollectUserInformation is aborted, i.e. the SRF remains connected and the next PlayAnnouncement or PromptAndCollectUserInformation is executed if available.

3.2.1.3 CancelFailed

3.2.1.3.1 General description

3.2.1.3.1.1 Error description

This Error is returned by Cancel if the cancelling of an Operation, as requested by the SCF, was not successful. Possible failure reasons are:

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.

3.2.1.3.1.2 Argument description

```
PARAMETER SEQUENCE {
        problem         [0] ENUMERATED {
                                unknown operation (0),
                                tooLate (1),
                                operationNotCancellable (2) },
        operation       [1] InvokeID
}
```

-- *The operation failed to be cancelled.*

3.2.1.3.2 Operations SCF → SSF

– CancelStatusReportRequest;

– Cancel.

3.2.1.3.3 Operations SCF → SRF

– Cancel.

Procedures at Invoking Entity (SCF)

A) Sending Cancel

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

The SCF sends a Cancel after a Play Announcement or PromptAndCollectUserInformation has been sent. The SCF remains in the same state.

B) Receiving CancelFailed Error

Precondition:	SCSM state any	Service Logic dependent.
Postcondition:	SCSM state any	Service Logic dependent.

After sending a Cancel operation the Service Logic may continue (e.g. sending another PlayAnnouncement or PromptAndCollectUserInformation or a DisconnectForwardConnection). The CancelFailed Error can therefore be received in any state. The treatment is Service Logic dependent.

Procedures at Responding Entity (SRF)

A) Receiving Cancel. However, the indicated PlayAnnouncement or PromptAndCollectUserInformation is not known, or already executed. This causes a failure, CancelFailed.

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.
or	SRSM state 1	Idle.

B) Sending CancelFailed Error

Precondition:	SRSM state 3	User Interaction.
or	SRSM state 1	Idle.
Postcondition:	SRSM state 3	User Interaction.
or	SRSM state 1	Idle.

After returning the CancelFailed the SRF stays in the same state.

3.2.1.4 ETCFailed

3.2.1.4.1 General description

3.2.1.4.1.1 Error description

ETCFailed is an error from SSF to SCF, indicating the fact that the establishment of a temporary connection to an assisting SSF or SRF was not successful (e.g. receiving a "Backwards Release" after sending an IAM).

3.2.1.4.2 Operations SCF → SSF

– EstablishTemporaryConnection.

Procedures at Invoking Entity (SCF)

A) SCF sends EstablishTemporaryConnection to SSF

Precondition:	SCSM state 3.1	Determine Mode.
Postcondition:	SCSM state 3.2	Waiting for AssistRequestInstructions.

B) SCF receives ETCFailed Error from SSF

Precondition:	SCSM state 3.2	Waiting for AssistRequestInstructions.
Postcondition:	SCSM state 2.1	Preparing SSF Instructions.

Error handling depends on the Service Logic, e.g. selecting another SRF or continue the processing of the call.

Procedures at Responding Entity (SSF)

A SSF receives EstablishTemporaryConnection from a SCF but the establishment of the connection fails, results in returning an ETCFailed Error to the SCF.

Precondition:	SSF-FSM state c	Waiting for Instructions.
Postcondition:	SSF-FSM state c	Waiting for Instructions.

No further Error treatment.

3.2.1.5 ImproperCallerResponse

3.2.1.5.1 General description

3.2.1.5.1.1 Error description

The format of the user input has been checked by the SRF and does not correspond to the required format as it was defined in the initiating PromptAndCollectUserInformation Operation.

3.2.1.5.2 Operations SCF → SRF

– PromptAndCollectUserInformation.

Procedures at Invoking Entity (SCF)

A) SCF sends PromptAndCollectUserInformation to SRF

Precondition:	SCSM state 3.1	Determine Mode; PromptAndCollectUserInformation will accompany the ConnectToResource.

	or	SCSM state 3.2	Waiting for AssistRequestInstructions; after EstablishTemporaryConnection.
	or	SCSM state 4.1	Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUser-Informations are active.
Postcondition:		SCSM state 4.1	Waiting for Response from the SRF.

B) SCF receives ImproperCallerResponse Error from SRF

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

Error treatment depends on Service Logic. A SCF can initiate new User Interaction or force a Disconnect (to SSF).

Procedures at Responding Entity (SRF)

A) SRF receives PromptAndCollectUserInformation

Precondition:	SRSM state 2	Connected.
or	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

B) Response from caller is not correct, SRF returns ImproperCallerResponse to SCF

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

SRF waits for a new Operation from SCF. This may be a new PromptAndCollectUserInformation or PlayAnnouncement.

3.2.1.6 MissingCustomerRecord

3.2.1.6.1 General description

3.2.1.6.1.1 Error description

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.

3.2.1.6.2 Operations SSF → SCF

- AnalysedInformation;
- AssistRequestInstructions;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- TAnswer;
- TBusy;
- TDisconnect;
- TermAttemptAuthorized;
- TMidCall;
- TNoAnswer.

Procedures at Invoking Entity (SSF)

A) Sending Operation

Precondition:	SSF-FSM state b	Trigger processing.
or	SSF-FSM state b'	Waiting for Instructions; in case of assist/hand-off.
Postcondition:	SSF-FSM state c	Waiting for Instructions.
or	SSF-FSM state b'	Waiting for Instructions; in case of assist/hand-off.

B) SSF receives Error "MissingCustomerRecord"

Precondition:	SSF-FSM state c	Waiting for Instructions.
or	SSF-FSM state b'	Waiting for Instructions; in case of assist/hand-off.
Postcondition:	SSF-FSM state a	Idle.
or	SSF-FSM state a'	Idle; in case of assist/hand-off.

The CCF routes the call if necessary (e.g. default routing to a terminating announcement).

Procedures at Responding Entity (SCF)

Precondition:	1) SCSM	Appropriate state.
	2) SCSM	Operation received, appropriate event occurred.
Postcondition:	1) SCSM state 1	Idle; in case of all operations listed above, except AssistRequestInstructions.
	2) SCSM state 2.1	Preparing SSF instructions; for AssistRequest-Instructions.

The SCSM detects that the required Service Logic Program does not exist. The Service Logic Program Instance may not exist anymore (e.g. in case of the operation AssistRequestInstructions), or the Service Logic Program may have never existed at all (i.e. the customer record in the SCF does not exist, e.g. in case of TDPs a Service Logic Program is attempted to be invoked). The Error parameter MissingCustomerRecord is used to inform the invoking entity of this situation. The maintenance functions are informed (however, it is optional for the TDP operation case).

3.2.1.6.3 Operations SRF → SCF

– AssistRequestInstructions.

Procedures at Invoking Entity (SRF)

A) Sending Operation

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 2	Connected.

B) SRF receives Error "MissingCustomerRecord"

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 1	Idle.

SRF initiated Disconnect.

Procedures at Responding Entity (SCF)

The SCSM detects that the required Service Logic Program does not exist (anymore). The establishment of a connection between the SSF and the SRF took too long or the correlationID was invalid. In both cases the requested Service Logic Program cannot be found. The Error parameter MissingCustomerRecord is used to inform the invoking entity of this situation. The maintenance functions are informed.

3.2.1.7 MissingParameter

3.2.1.7.1 General description

3.2.1.7.1.1 Error description

There is an Error in the received Operation argument. The responding entity cannot start to process the requested Operation because the argument is incorrect: a mandatory parameter (the application shall always return this error in case it is not detected by the ASN.1 decoder) or an expected optional parameter which is essential for the application is not included in the Operation argument.

3.2.1.7.2 Operations SCF → SSF

Non Call Associated

- ActivateServiceFiltering.

Call Associated/Non Call Processing

- ApplyCharging;
- CallInformationRequest;
- Cancel;
- CancelStatusReportRequest;
- FurnishChargingInformation;
- RequestCurrentStatusReport;
- RequestEveryStatusChangeReport;
- RequestFirstStatusMatchReport;
- RequestNotificationChargingEvent;
- RequestReportBCSMEvent;
- ResetTimer;
- SendChargingInformation.

Call Associated/Call Processing

- AnalyseInformation;
- CollectInformation;
- Connect;
- ConnectToResource;
- EstablishTemporaryConnection;
- HoldCallInNetwork;
- InitiateCallAttempt;
- SelectFacility;
- SelectRoute.

Procedures at Invoking Entity (SCF)

A) Sending Operation

Precondition:	SCSM	Any state in which the above Call associated operations can be transferred.
	SCME	Any state in which the above Non call associated operations can be transferred.
Postcondition:	SCSM	Any state as result of the transfer of any of the above operations.
	SCME	Any state as result of the transfer of any of the above Non call associated operations.

B) SCF receives Error "MissingParameter"

Precondition:	SCSM	Any state as result of the transfer of any of the above Call associated operations.
	SCME	Any state as result of the transfer of any of the above Non call associated operations.
Postcondition:	SCSM	Transition to the initial state (i.e. before sending the erroneous operation).
	SCME	Transition to the initial state (i.e. before sending the erroneous operation).

The Service Logic and maintenance functions are informed. Further treatment of the call is dependent on Service Logic.

Procedures at Responding Entity (SSF)

Precondition:	1)	SSF-FSM	Appropriate state.
	2)	SSF-FSM	Call associated operation received, appropriate event occurred.
	3)	SSME	Appropriate state.
	4)	SSME	Non call associated operation received, appropriate event.
Postcondition:	1)	SSF-FSM	Transition to the same state.
	2)	SSME	Transition to the initial state (i.e. before receiving the erroneous operation).

The SSF-FSM detects the error in the received operation. The Error parameter is returned to inform the SCF of this situation.

3.2.1.7.3 Operations SSF → SCF

- AnalysedInformation;
- ApplyChargingReport;
- AssistRequestInstructions;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- TAnswer;
- TBusy;
- TDisconnect;
- TermAttemptAuthorized;
- TMidCall;
- TNoAnswer.

Procedures at Invoking Entity (SSF)

A) Sending Operation

Precondition:	SSF-FSM	Any state in which the above operations can be transferred.
Postcondition:	SSF-FSM	Any state as result of the transfer of any of the above operations.

B) SSF receives Error "MissingParameter"

Precondition:	SSF-FSM	Any state as result of the transfer of any of the above operations.
Postcondition:	SSF-FSM state a	Idle.

After receiving this Error, the SSF-FSM returns to the state Idle. The CCF routes the call if necessary (default routing to a terminating announcement). If the call is already established (i.e. mid-call trigger or ApplyChargingReport), the CCF may maintain the call or disconnect it. The choice between these two options is network operator specific. In case of an assisting SSF, the temporary connection is released by the assisting SSF.

Procedures at Responding Entity (SCF)

Precondition:	1)	SCSM	Appropriate state.
	2)	SCSM	Operation received, appropriate event occurred.
Postcondition:	1)	SCSM state 1	Idle; in case of any operation listed above, except AssistRequestInstructions.
or	2)	SCSM state 2.1	Preparing SSF Instructions; in case of AssistRequestInstructions.

The SCSM detects the erroneous situation. The Error parameter is used to inform the SSF of this situation. The Service Logic and maintenance functions are informed.

3.2.1.7.4 Operations SCF → SRF

- PlayAnnouncement;
- PromptAndCollectUserInformation;
- Cancel.

Procedures at Invoking Entity (SCF)

A) Sending Operation

Precondition:	SCSM state 3.1	Determine Mode; PromptAndCollectUserInformation or PlayAnnouncement will accompany the ConnectTo-Resource.
or	SCSM state 3.2	Waiting for AssistRequestInstructions; after EstablishTemporaryConnection.
or	SCSM state 4.1	Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUserInfor-mations are outstanding.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

B) Receiving Error

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

Error treatment depends on Service logic. SCF can initiate new User Interaction or force Disconnect (to SSF).

Procedures at Responding Entity (SRF)

Precondition:	SRSM state 2	Connected.
or	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

The SRSM detects that a required parameter is not present in the Operation argument. The Error parameter MissingParameter is used to inform the SCF of this situation. The SCF should take the appropriate actions to treat this error.

3.2.1.7.5 Operations SRF → SCF

- AssistRequestInstructions.

Procedures at Invoking Entity (SRF)

A) Sending Operation

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 2	Connected.

B) Receiving Error

Precondition:	SRSM state 2	Connected.
Postcondition:	SRSM state 1	Idle.

Procedures at Responding Entity (SCF)

Precondition:	SCSM state 3.2	Waiting for AssistRequestInstructions.
Postcondition:	SCSM state 2.1	Preparing SSF instructions.

The SCSM detects the error in the received operation. The Error parameter is used to inform the SRF of this situation. The Service Logic and maintenance functions are informed. The SCF might try another SRF, route the call or release the call (Service Logic dependent).

3.2.1.8 Name error

3.2.1.8.1 General description

3.2.1.8.1.1 Error description

This error is sent by the SDF to the SCF to report a problem related to the name of the object. The conditions under which an attribute error is to be issued are defined in 12.5/X.511 (1993).

3.2.1.8.1.2 Argument description

The name error parameter and problem codes are specified in 12.5/X.511 (1993).

3.2.1.8.2 Operations SCF → SDF

- Search;
- Add Entry;
- Remove Entry;
- Modify Entry.

Procedure at Invoking Entity (SCF)

A) Sending Operation

Precondition:	SCSM state 4	SDF Bound or
	SCSM state 2	Wait for subsequent requests.
Postcondition:	SCSM state 4	SDF Bound or
	SCSM state 2	Wait for subsequent requests.

B) Receiving Error

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is dependent on the Service Logic. If the SCF is able to change the request, it can do another SDF query, otherwise the service processing should be terminated.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition:	SDF-FSM state 3	SCF Bound.
Postcondition:	SDF-FSM state 3	SCF Bound.

B) Returning Error

Precondition:	SDF-FSM state 3	SCF Bound.
Postcondition:	SDF-FSM state 3	SCF Bound.

The SDF could not perform the operation due to a name problem and therefore sends a Name error to the SCF. After returning the error, no further error treatment is performed.

3.2.1.9 ParameterOutOfRange

3.2.1.9.1 General description

3.2.1.9.1.1 Error description

The responding entity cannot start the processing of the requested Operation because an Error in a parameter of the Operation argument is detected: a parameter value is out of range. This error is applied for the following two cases (when the error is determined by the application):

1) For the parameter which type is defined with the range of its size, such as INTEGER(x..y), SEQUENCE SIZE(x..y) OF Type. This error is applied when the parameter value is z or the parameter size is z where $z < x$ or $z > y$.

2) For the parameter which type is defined as list of ENUMERATED value, the ParameterOutOfRange error is applied when the parameter value is not equal to any of the ENUMERATED values in the list.

3.2.1.9.2 Operations SCF → SSF

Non Call Associated

– ActivateServiceFiltering.

Call Associated/Non Call Processing

– ApplyCharging;
– CallInformationRequest;
– RequestCurrentStatusReport;
– RequestEveryStatusChangeReport;
– RequestFirstStatusMatchReport;
– SendChargingInformation.

Call Associated/Call Processing

– AnalyzeInformation;
– CollectInformation
– Connect;
– InitiateCallAttempt;
– RequestNotificationChargingEvent;
– RequestReportBCSMEvent;
– ResetTimer;
– SelectFacility;
– SelectRoute.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.9.3 Operations SSF → SCF

– AnalyzedInformation;
– ApplyChargingReport;
– CollectedInformation;
– InitialDP;
– OAnswer;
– OCalledPartyBusy;
– Disconnect;
– OMidCall;
– ONoAnswer;
– OriginationAttemptAuthorized;
– RouteSelectFailure;
– TAnswer;
– TCalledPartyBusy
– TDisconnect;
– TermAttemptAuthorized
– TMidCall;
– TNoAnswer.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.9.4 Operations SCF → SRF

- PlayAnnouncement;
- PromptAndCollectUserInformation.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.10 RequestedInfoError

3.2.1.10.1 General description

3.2.1.10.1.1 Error description

The RequestedInfoError is an immediate response to the CallInformationRequest operation, indicating that the requested information is not known to the SSF or is not available. RequestedInfoError is used when a specific SSF-CCF cannot offer the information specified with RequestedInformationType but there exists other SSF-CCF that can offer the information.

3.2.1.10.1.2 Argument description

PARAMETER ENUMERATED {
unknownRequestedInfo (1),
requestedInfoNotAvailable (2)
-- *other values FFS*
}

3.2.1.10.2 Operations SCF → SSF

- CallInformationRequest.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.11 Service error

3.2.1.11.1 General description

3.2.1.11.1.1 Error description

This error is sent by the SDF to the SCF 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 12.8/X.511 (1993).

3.2.1.11.1.2 Argument description

The Service error parameter and problem codes are specified in 12.8/X.511 (1993).

3.2.1.11.2 Operations SCF → SDF

- Bind;
- Search;
- Add Entry;
- Remove Entry;
- Modify Entry.

Procedure at Invoking Entity (SCF)

A) Sending Operation

Precondition:		SCSM state 1	Idle (in case of Bind).
	or	SCSM state 2	Wait for subsequent requests (in case of operations except Bind).
	or	SCSM state 4	SDF Bound (in case of operations except Bind).

	Postcondition:	SCSM state 2	Wait for subsequent requests (in case of all the possible operations).
	or	SCSM state 4	SDF Bound (in case of operations except Bind).

B) Receiving Error

Precondition:	SCSM state 3	Wait for Bind result (in case of Bind).
or	SCSM state 4	SDF Bound (in case of operations except Bind).
Postcondition:	SCSM state 1	Idle (in case of Bind).
or	SCSM state 4	SDF Bound (in case of operations except Bind).

Error Procedure is dependent on the Service Logic. If there is an alternative SDF it may be possible to do another query, otherwise the service processing should be terminated.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition:	SDF-FSM state 1	Idle (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).
Postcondition:	SDF-FSM state 2	Bind Pending (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).

B) Returning Error

Precondition:	SDF-FSM state 2	Bind Pending (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).
Postcondition:	SDF-FSM state 1	Idle (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).

The SDF could not perform the operation due to a service related problem and sends a Service error to the SCF. After returning the error, no further error treatment is performed.

3.2.1.12 Security error

3.2.1.12.1 General description

3.2.1.12.1.1 Error description

This error is sent by the SDF to the SCF 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 12.7/X.511 (1993).

3.2.1.12.1.2 Argument description

The security error parameter and problem codes are specified in 12.7/X.511 (1993).

3.2.1.12.2 Operations SCF → SDF

- Bind;
- Search;
- Add Entry;
- Remove Entry;
- Modify Entry.

Procedure at Invoking Entity (SCF)

A) Sending Operation

Precondition:	SCSM state 1	Idle (in case of Bind).
or	SCSM state 2	Wait for subsequent requests (in case of operations except Bind).
or	SCSM state 4	SDF Bound (in case of operations except Bind).
Postcondition:	SCSM state 2	Wait for subsequent requests (in case of all the possible operations).
or	SCSM state 4	SDF Bound (in case of operations except Bind).

B) Receiving Error

Precondition:	SCSM state 3	Wait for Bind result (in case of Bind).
or	SCSM state 4	SDF Bound (in case of operations except Bind).
Postcondition:	SCSM state 1	Idle (in case of Bind).
or	SCSM state 4	SDF Bound (in case of operations except Bind).

Error Procedure is independent of the Service Logic. The service processing should be terminated.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition:	SDF-FSM state 1	Idle (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).
Postcondition:	SDF-FSM state 2	Bind Pending (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).

B) Returning Error

Precondition:	SDF-FSM state 2	Bind Pending (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).
Postcondition:	SDF-FSM state 1	Idle (in case of Bind).
or	SDF-FSM state 3	SCF Bound (in case of operations except Bind).

The SDF could not perform the operation for security reasons and therefore sends a Security error to the SCF. After returning the error, no further error treatment is performed.

3.2.1.13 SystemFailure

3.2.1.13.1 General description

3.2.1.13.1.1 Error description

This 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 (clarification is FFS).

3.2.1.13.1.2 Argument description

PARAMETER

UnavailableNetworkResource

UnavailableNetworkResource ::= ENUMERATED {
unavailableResources (0),
componentFailure (1),
basicCallProcessingException (2),
resourceStatusFailure (3),
endUserFailure (4)}

3.2.1.13.2 Operations SCF → SSF

Non Call Associated

- ActivateServiceFiltering.

Call Associated/Non Call Processing

- ApplyCharging;
- CallInformationRequest;
- CancelStatusReportRequest;
- RequestCurrentStatusReport;
  RequestEveryStatusChangeReport;
- RequestFirstStatusMatchReport;
- RequestNotificationChargingEvent;
- RequestReportBCSMEvent;
- SendChargingInformation.

Call Associated/Call Processing

- AnalyseInformation;
- CollectInformation;
- Connect;
- ConnectToResource;
- DisconnectForwardConnection;
- EstablishTemporaryConnection;
- HoldCallInNetwork;
- InitiateCallAttempt;
- SelectFacility;
- SelectRoute.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.13.3 Operations SSF → SCF

- AnalysedInformation;
- ApplyChargingReport;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- TAnswer;
- TBusy;
- TDisconnect;
- TermAttemptAuthorized;
- TMidCall;
- TNoAnswer.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.13.4 Operations SCF → SRF

- Cancel;
- PlayAnnouncement;
- PromptAndCollectUserInformation.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.13.5 Operations SRF → SCF

- AssistRequestInstructions.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.14 Task refused

3.2.1.14.1 General introduction

3.2.1.14.1.1 Error description

This 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. (The precise definition is FFS.)

3.2.1.14.1.2 Argument description

```
PARAMETER ENUMERATED {
    generic (0),
    unobtainable (1),
    congestion (2)
    -- other values FFS
    }
```

3.2.1.14.2 Operations SCF → SSF

Non Call Associated

- ActivateServiceFiltering.

Call Associated/Non Call Processing

- ApplyCharging;
- CallInformationRequest;
- Cancel;
- CancelStatusReportRequest;
- FurnishChargingInformation;
- RequestCurrentStatusReport;
- RequestEveryStatusChangeReport;
- RequestFirstStatusMatchReport;
- RequestNotificationChargingEvent;
- RequestReportBCSMEvent;
- ResetTimer;
- SendChargingInformation.

Call Associated/Call Processing

- AnalyseInformation;
- CollectInformation;
- Connect;
- ConnectToResource;
- DisconnectForwardConnection;
- EstablishTemporaryConnection;
- HoldCallInNetwork;
- InitiateCallAttempt;
- SelectFacility;
- SelectRoute;

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.14.3 Operations SSF → SCF

- AnalysedInformation;
- ApplyChargingReport;
- AssistRequestInstructions;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- TAnswer;
- TBusy;
- TDisconnect;
- TermAttemptAuthorized;
- TMidCall;
- TNoAnswer.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.14.4 Operations SCF → SRF

- Cancel;
- PlayAnnouncement;
- PromptAndCollectUserInformation.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.14.5 Operations SRF → SCF

- AssistRequestInstructions.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.15 UnavailableResource

3.2.1.15.1 General description

3.2.1.15.1.1 Error description

The SRF is not able to perform its function (i.e. play a certain announcement and/or collect specific user information), and cannot be replaced. A reattempt is not possible.

3.2.1.15.2 Operations SCF → SRF

- PlayAnnouncement;
- PromptAndCollectUserInformation.

Procedures at Invoking Entity (SCF)

A) SCF sends PlayAnnouncement or PromptAndCollectUserInformation to SRF

Precondition:		SCSM state 3.1	Determine Mode; PlayAnnouncement or PromptAnd-CollectUserInformation will accompany the ConnectToResource.
	or	SCSM state 3.2	Waiting for AssistRequestInstructions; after Establish-TemporaryConnection.
	or	SCSM state 4.1	Waiting for Response from the SRF; if more PlayAnnouncements or PromptAndCollectUser-Informations are outstanding.
Postcondition:		SCSM state 4.1	Waiting for Response from the SRF.

B) SCF receives UnavailableResource Error from SRF

Precondition:	SCSM state 4.1	Waiting for Response from the SRF.
Postcondition:	SCSM state 4.1	Waiting for Response from the SRF.

If the chosen resource cannot perform its function, the further treatment is service dependent.

Examples:
- request SSF to connect to alternative SRF;
- service processing without PlayAnnouncement or PromptAndCollectUserInformation (if possible);
- terminate service processing.

Procedures at Responding Entity (SRF)

A) SRF receiving PlayAnnouncement or PromptAndCollectUserInformation

Precondition:	SRSM state 2	Connected; if initial PlayAnnouncement or PromptAndCollectUserInformation.
or	SRSM state 3	User Interaction; if not initial PlayAnnouncement or PromptAndCollectUserInformation.

B) SRF is not able to perform its function (and cannot be replaced). SRF sends UnavailableResource.

Precondition:	SRSM state 3	User Interaction.
Postcondition:	SRSM state 3	User Interaction.

3.2.1.16 UnexpectedComponentSequence

3.2.1.16.1 General description

3.2.1.16.1.1 Error description

The responding entity cannot start the processing of the requested operation because a SACF or MACF rule is violated, or the operation could not be processed in the current state of the FSM.

3.2.1.16.2 Operations SCF → SSF

Non Call Associated

- ActivateServiceFiltering.

Call Associated/Non Call Processing

- ApplyCharging;
- CallInformationRequest;
- FurnishChargingInformation;
- RequestCurrentStatusReport;
- RequestEveryStatusChangeReport;
- RequestFirstStatusMatchReport;
- RequestNotificationChargingEvent;
- RequestReportBCSMEvent;
- ResetTimer;
- SendChargingInformation.

Call Associated/Call Processing

- AnalyseInformation;
- CollectInformation;
- Connect;
- ConnectToResource;
- DisconnectForwardConnection;
- EstablishTemporaryConnection;
- HoldCallInNetwork;
- InitiateCallAttempt;
- SelectFacility;
- SelectRoute.

In this case the SSF detects the erroneous situation, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF the Service Logic and maintenance functions are informed and the Service Logic decides about error treatment.

3.2.1.16.3 Operations SSF → SCF

- AnalysedInformation;
- ApplyChargingReport;
- AssistRequestInstructions;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- Answer;
- TBusy;
- TDisconnect;
- TermAttemptAuthorized;
- TMidCall;
- TNoAnswer.

In case of assisting SSF an error occurs in case an AssistRequestInstructions is sent while a relationship between SCF and assisting SSF has already been established, the SCF returns the error parameter. Service Logic and maintenance are informed. On receiving the error, the assisting SSF moves to Idle and the temporary connection is released.

In case the operation is sent by an "initiating" SSF in the context of an existing relationship, the SCF returns the error parameter. Service Logic and maintenance are informed. On receiving the error, the SSF moves to Idle.

3.2.1.16.4 Operations SCF → SRF (only applicable for direct SCF-SRF case)

- PlayAnnouncement;
- PromptAndCollectUserInformation.

In this case the SRF detects the erroneous situation, sends the UnexpectedComponentSequence error and remains in the same state. In the SCF, the Service Logic and maintenance functions are informed and the Service Logic decides about error treatment. Possible error treatment is to send the DisconnectForwardConnection operation to the SSF.

3.2.1.16.5 Operations SRF → SCF

- AssistRequestInstructions.

In this case, an error occurs if the SRF has already an established relationship with the SCF and sends an AssistRequestInstructions. The SCF detects the erroneous situation, informs Service Logic and maintenance functions and returns the error parameter. On receiving the parameter, the SRF moves to Idle and releases the temporary connection.

3.2.1.17 UnexpectedDataValue

3.2.1.17.1 General description

3.2.1.17.1.1 Error description

The responding entity cannot complete the processing of the requested Operation because a parameter has an unexpected data value.

Note that this error does not overlap with "ParameterOutOfRange"

Example: **startTime DateAndTime ::=** -- *value indicating January 32 1993, 12:15:01*

The responding entity does not expect this value and responds with "UnexpectedDataValue".

3.2.1.17.2 Operations SCF → SSF

Non Call Associated

- ActivateServiceFiltering.

Call Associated/Non Call Processing

- ApplyCharging;
- CallInformationRequest;
- Cancel;
- CancelStatusReportRequest;
- FurnishChargingInformation;
- RequestCurrentStatusReport;
- RequestEveryStatusChangeReport;
- RequestFirstStatusMatchReport;
- RequestNotificationChargingEvent;
- RequestReportBCSMEvent;
- ResetTimer;
- SendChargingInformation.

Call Associated/Call Processing

- AnalyseInformation;
- CollectInformation;
- Connect;
- ConnectToResource;
- EstablishTemporaryConnection;
- HoldCallInNetwork;
- InitiateCallAttempt;
- SelectFacility;
- SelectRoute.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.17.3 Operations SSF → SCF

- AnalysedInformation;
- ApplyChargingReport;
- AssistRequestInstructions;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- TAnswer;
- TBusy;
- TDisconnect;
- TermAttemptAuthorized;
- TMidCall;
- TNoAnswer.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.17.4 Operations SCF → SRF

- Cancel;
- PlayAnnouncement;
- PromptAndCollectUserInformation.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.17.5 Operations SRF → SCF

- AssistRequestInstructions.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.18 UnexpectedParameter

3.2.1.18.1 General description

3.2.1.18.1.1 Error description

There is an error in the received Operation argument. A valid but unexpected parameter was present in the Operation argument. The presence of this parameter is not consistent with the presence of the other parameters. The responding entity cannot start to process the Operation.

3.2.1.18.2 Operations SCF → SSF

Non Call Associated

- ActivateServiceFiltering.

Call Associated/Non Call Processing

- ApplyCharging;
- CallInformationRequest;
- FurnishChargingInformation;
- RequestCurrentStatusReport;
- RequestEveryStatusChangeReport;
- RequestFirstStatusMatchReport;
- RequestNotificationChargingEvent;
- RequestReportBCSMEvent;
- ResetTimer;
- SendChargingInformation.

Call Associated/Call Processing

- AnalyseInformation;
- CollectInformation;
- Connect;
- ConnectToResource;
- EstablishTemporaryConnection;
- HoldCallInNetwork;
- InitiateCallAttempt;
- SelectFacility;
- SelectRoute.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.18.3 Operations SSF → SCF

- AnalysedInformation;
- ApplyChargingReport;
- AssistRequestInstructions;
- CollectedInformation;
- InitialDP;
- OAnswer;
- OCalledPartyBusy;
- ODisconnect;
- OMidCall;
- ONoAnswer;
- OriginationAttemptAuthorized;
- RouteSelectFailure;
- TAnswer;
- TBusy;

– TDisconnect;
– TermAttemptAuthorized;
– TMidCall;
– TNoAnswer.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.18.4 Operations SCF → SRF

– PlayAnnouncement;
– PromptAndCollectUserInformation.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.18.5 Operations SRF → SCF

– AssistRequestInstructions.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.19 UnknownLegID

3.2.1.19.1 General description

3.2.1.19.1.1 Error description

This error is used to indicate to the SCF that a specific leg, indicated by the LegID parameter value in the operation, is unknown to the SSF.

3.2.1.19.2 Operations SCF → SSF

Call Associated/Non Call Processing

– SendChargingInformation.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.20 UnknownResource

3.2.1.20.1 General description

3.2.1.20.1.1 Error description

This 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.

3.2.1.20.2 Operations SCF → SSF

– Call Associated/Non Call Processing;
– RequestCurrentStatusReport;
– RequestEveryStatusChangeReport;
– RequestFirstStatusMatchReport.

Refer to 3.2.1.7 MissingParameter for the appropriate error procedures.

3.2.1.21 Update error

3.2.1.21.1 General description

3.2.1.21.1.1 Error description

This error is sent by the SDF to the SCF 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 12.9/X.511 (1993).

3.2.1.21.1.2 Argument description

The update error parameter and problem codes are specified in 12.9/X.511 (1993).

3.2.1.21.2 Operations SCF → SDF

- Add Entry;
- Remove Entry;
- Modify Entry.

Procedure at Invoking Entity (SCF)

A) Sending Operation

Precondition:	SCSM state 4	SDF Bound or
	SCSM state 2	Wait for subsequent requests.
Postcondition:	SCSM state 4	SDF Bound or
	SCSM state 2	Wait for subsequent requests.

B) Receiving Error

Precondition:	SCSM state 4	SDF Bound.
Postcondition:	SCSM state 4	SDF Bound.

Error Procedure is dependent on the Service Logic.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition:	SDF-FSM state 3	SCF Bound.
Postcondition:	SDF-FSM state 3	SCF Bound.

B) Returning Error

Precondition:	SDF-FSM state 3	SCF Bound.
Postcondition:	SDF-FSM state 3	SCF Bound.

The SDF could not perform the operation due to a problem related to the addition, deletion or modification of information and sends an Update error to the SCF. After returning the error, no further error treatment is performed.

3.2.2 Entity related error procedures

The following subclauses define the error handling for the entity related errors. Since the error situations are not originated by the reception of an operation, the invoking entity is denoted here as the entity at which the error situation is detected. The responding entity is the entity which receives the error report.

The TCAP services used for reporting errors are described in 3.4.

3.2.2.1 Expiration of T_{SSF}

3.2.2.1.1 General description

3.2.2.1.1.1 Error description

A timeout occurred in the SSF on the response from the SCF.

3.2.2.1.2 Procedures SSF → SCF

Procedure at the Invoking Entity (SSF)

Timeout occurs in SSF on T_{SSF}.

Precondition:	SSF-FSM state c	Waiting for Instructions.
or	SSF-FSM state d	Waiting for end of User Interaction.
or	SSF-FSM state e	Waiting for end of Temporary connection.
Postcondition:	SSF-FSM state a	Idle.

The SSF-FSM aborts the dialogue and moves to the Idle state, the CCF routes the call if necessary (e.g. default routing to a terminating announcement). The abort is reported to the maintenance functions.

Procedure at the Responding Entity (SCF)

SCF receives a dialogue abort.

Precondition:	Any state.	
Postcondition:	SCSM state 1	Idle; if the abort is related to a SSF dialogue.
or	SCSM state 2	Preparing SSF instructions; if the abort is related to an assisting SSF dialogue.

The SCF releases all allocated resources and reports the abort to the maintenance functions, if the abort is received on a SSF dialogue. The SCF releases all resources related to the dialogue, reports the abort to the maintenance functions and returns to state preparing SSF instructions, if the abort is received on an assisting SSF dialogue.

3.2.2.2 Expiration of T_{SRF}

3.2.2.2.1 General description

3.2.2.2.1.1 Error description

A timeout occurred in the SRF on the response from the SCF. This procedure concerns only the direct SCF-SRF case.

3.2.2.2.2 Procedures SRF → SCF

Procedure at the Invoking Entity (SRF)

Timeout occurs in SRF on T_{SRF}.

Precondition:	SRSM state 2	Connected.
or	SRSM state 3	User Interaction.
Postcondition:	SRSM state 1	Idle.

The SRF aborts the dialogue and moves to the Idle state, all allocated resources are de-allocated. The abort is reported to the maintenance functions.

Procedure at the Responding Entity (SCF)

SCF receives a dialogue abort.

Precondition:	SCSM state 4	User Interaction.
Postcondition:	SCSM state 2	Preparing SSF instructions.

The SCF releases all resources related to the dialogue, reports the abort to the maintenance functions and returns to state preparing SSF instructions.

3.3 Detailed Operation procedures

The intended use of operations and parameters from clause 2 not described in the following detailed procedure descriptions is for further study. For more information please refer to 6.4/Q.1214.

3.3.1 ActivateServiceFiltering procedure

3.3.1.1 General description

When receiving this operation, the SSF handles calls to destinations in a specified manner without request for instructions to the SCF. In the case of service filtering, the SSF executes a specific service filtering algorithm. For the transfer of service filtering results refer to the operation 'ServiceFilteringResponse'.

3.3.1.1.1 Parameters

a) filteredCallTreatment:

This parameter specifies how filtered calls are treated. It includes information about the announcement to be played, the charging approach, the number of counters used and the release cause to be applied to filtered calls.

i) sFBillingChargingCharacteristics:

This parameter determines the charging to be applied for service filtering. Its content is network specific.

NOTE – Actual format and encoding is for further study.

ii) informationToSend:

This parameter indicates an announcement, a tone or display information to be sent to the calling party. At the end of information sending, the call shall be released.

- inbandInfo:

  This parameter specifies the inband information to be sent.

  – messageID:

    This parameter indicates the message(s) to be sent, it can be one of the following:

    1) elementaryMessageID:

       This parameter indicates a single announcement.

    2) text:

       This parameter indicates a text to be sent. The text shall be transformed to inband information (speech). This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

    3) elementaryMessageIDs:

       This parameter specifies a sequence of announcements.

    4) variableMessage:

       This parameter specifies an announcement with one or more variable parts.

  – numberOfRepetitions:

    This parameter indicates the maximum number of times the message shall be sent to the end-user.

  – duration:

    This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

  – interval:

    This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

- tone:

  This parameter specifies a tone to be sent to the end-user.

  – toneID:

    This parameter indicates the tone to be sent.

  – duration:

    This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

- displayInformation:

  This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

iii) maximumNumberOfCounters:

This parameter provides the number of counters to be allocated as well as the number of destinations included in the service filtering, i.e. 'maximumNumberOfCounters' subsequent destination addresses beginning with the destination address provided in 'filteringCriteria' are used for service filtering. One counter is assigned to each of these destination addresses.

The number of counters may only be >1 if the ’filteringCriteria’ are of the type ’addressAndService’.

iv) releaseCause:

This parameter provides the cause value used for call release after the ’informationToSend’ (for example announcement) has been sent to the calling party. If ’releaseCause’ is not present, the default value is the same as the ISUP value decimal 31 (normal unspecified).

b) filteringCharacteristics:
This parameter indicates the severity of the filtering and the point in time when the ’ServiceFilteringResponse’ shall be sent. It determines whether the ’interval’ or the ’numberOfCalls’ are used.

i) interval:

After expiration of the interval timer the next call to arrive causes following actions:

- sending of an ’InitialDP’ or a DP-specific operation;
- sending of a ’ServiceFilteringResponse’;
- starting again the interval timer.

When filtering is started the first interval is started.

An interval of 0 indicates that all calls matching the filtering criteria will result in sending of an ’InitialDP’ or a DP-specific operation and no filtering will be applied (i.e. no ’ServiceFilteringResponse’ will be sent).

An interval of –1 indicates that none of the calls matching the filtering criteria will either result in sending of an 'InitialDP' or a DP-specific operation or a 'ServiceFilteringResponse' operation.

Other values indicate duration in seconds.

ii) numberOfCalls:

The *n* th call causes an 'InitialDP' or a DP-specific operation and a 'ServiceFilteringResponse' operation sent to the SCF. This threshold value is met if the sum of all counters assigned to one service filtering entity is equal to 'numberOfCalls'.

A number of calls of 0 indicates that none of the calls matching the filtering criteria will result in sending of an 'InitialDP' or a DP-specific operation and a 'ServiceFilteringResponse' operation.

c) filteringTimeOut:

This parameter indicates the duration of the filtering. When the time expires, a 'ServiceFilteringResponse' is sent to the SCF and service filtering is stopped. Two approaches are supported (duration or stopTime):

i) duration:

If the duration time expires, then service filtering is stopped and the final report is sent to the SCF.

A duration of 0 indicates that service filtering is to be removed.

A duration of –1 indicates an infinite duration.

A duration of –2 indicates a network specific duration.

Other values indicate duration in seconds.

ii) stopTime:

When the 'stopTime' is met then service filtering is stopped and the final report is sent to the SCF. If 'stopTime' was already met, i.e. the value of the stopTime is less than the value of the actual time but the difference does not exceed the value equivalent to 50 years, then service filtering is immediately stopped and the actual counter values are reported to the SCF. This occurs in cases where the SCF wishes to explicitly stop a running service filtering.

d) filteringCriteria:

This parameter specifies which calls are filtered based on 'serviceKey', 'callingAddressValue', 'calledAddressValue' or 'locationNumber'. It is a choice of 'servicekey' or 'addressAndService'.

i) serviceKey:

This parameter identifies unambiguously the requested IN service for which filtering should be applied.

ii) addressAndService:

This parameter identifies the IN service and dialled number for which filtering should be applied. The geographical area may also be identified ('callingAddressValue' and/or 'locationNumber').

iii) calledAddressValue:

This parameter contains the dialled number towards which filtering shall be applied. The complete called party number shall be specified.

iv) serviceKey:

This parameter identifies unambiguously the requested IN service for which filtering should be applied.

v) callingAddressValue:

This parameter contains the calling party number which identifies the calling party or geographical origin of the call for which filtering shall be applied.

vi) locationNumber:

This parameter identifies the geographical area from which the call to be filtered originates. It is used when 'callingAddressValue' does not contain any information about the geographical location of the calling party.

e) startTime:

This parameter defines when filtering is started. If 'startTime' is not provided or was already met, the SSF starts filtering immediately.

3.3.1.2 Invoking Entity (SCF)

3.3.1.2.1 Normal procedure

SCF Precondition:

– SLPI detects that service filtering has to be initiated at the SSF.

SCF Postconditions:

1) SLPI starts an application timer to monitor the expected end of service filtering.

2) The SCME is in the state "Waiting For ServiceFilteringResponse".

Sending the 'ActivateServiceFiltering' operation causes a transition of the SCME from the state "Service Filtering Idle" to the state "Waiting For SSF Service Filtering Response". The SCME remains in this state until the application timer in the SLPI expires. The SCME is informed by the SLPI about timer expiration. Then it moves to the state "Service Filtering Idle".

If no errors occurred after receiving an 'ActivateServiceFiltering' at the SSF an empty Return Result is sent to the SCF. That causes no state transition in the SCME.

To change the parameters of an existing service filtering entity, the SCF has to send an 'ActivateServiceFiltering' operation with the same 'filtering Criteria'. The second parameter set replaces the first one.

3.3.1.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.1.3 Responding Entity (SSF)

3.3.1.3.1 Normal procedure

SSF Precondition:

- None.

SSF Postcondition:

- The SSME-FSM is in the state "Non Call Associated Treatment".

If there is no already existing SSME-FSM for the "filteringCriteria" provided, then a new SSME-FSM is created. This SSME-FSM enters the state "Non-Call Associated Treatment" and initializes the service filtering for the specified IN calls. The parameters "filteredCallTreatment", "filteringCharacteristics", "filteringCriteria", "filteringTimeOut" and "startTime" are set as provided in the operation. A number of counters will be allocated and reset. In the case of the "startTime" that has not been met yet, the service filtering will be started at the specified point in time.

If the operation "ActivateServiceFiltering" addresses an already existing service filtering entity, the parameters "filteredCallTreatment", "filteringCharacteristics", "filteringTimeOut" and "startTime" are modified as provided in the operation. In the case that the addressed service filtering entity is active, the SSF reports the counter values to the SCF via the operation "ServiceFilteringResponse". The service filtering process is stopped if an already expired "stopTime" or "duration" equal to ZERO or a new not yet met "startTime" is provided. The SSF then proceeds as described for "ServiceFilteringResponse". In the case of the "startTime" that has not been met yet, the service filtering will be continued at the specified point in time.

If the service filtering proceeds, then the SSME-FSM remains in the state "Non-Call Associated Treatment". Otherwise the SSME-FSM moves to state "Idle Management".

When a call matches several active "filteringCriteria" it should be subject to filtering on the most specific criteria, i.e. the criteria with the longest "callingAddressValue" or "locationNumber", or alternatively the criteria with the largest number of parameters specified.

When performing service filtering with the "filteringCriteria"-"addressAndService", the first parameters checked will always be the "serviceKey" and "calledAddressValue".

If an "ActivateServiceFiltering" operation is passed to the SSF with the "filteringCriteria" "addressAndService" with both "callingAddressValue" and "locationNumber" present, the following is applicable:

- When the SSF receives a call that matches "serviceKey" and "calledAddressValue" (in the active "filteringCriteria"), it investigates whether or not the "locationNumber" is present in the initial address message. If it is present and matches the active "filteringCriteria", the call is filtered. If the SSF finds that the "locationNumber" is absent, then it will check the "callingAddressValue" and perform filtering depending on that parameter.

If no errors occurred after receiving an "ActivateServiceFiltering" on the SSF, an empty Return Result is sent to the SCF. That causes no state transition in the SSME-FSM.

Following application timers are used:

- detect moment to start service filtering (start time);
- duration time for service filtering;
- interval time for service filtering (for timer controlled approach).

3.3.1.3.2 Error handling

If the SSF detects an error with any of the defined error values, then this error is reported to the SCF.

The event is recorded in the SSF and an error condition indicated.

In case a new SSME-FSM should be created, the relationship is ended and all concerned resources (e.g. counters) are released. The SSME-FSM remains in the state "Idle Management".

In case there is already an existing SSME-FSM, the service filtering data remains unchanged. The SSME-FSM remains in the state "Non-Call Associated Treatment".

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.2 ActivityTest procedure

3.3.2.1 General description

This operation is used to check for the continued existence of a relationship between the SCF and SSF. If the relationship is still in existence, then the SSF will respond. If no reply is received within a given time period, then the SCF will assume that the SSF has failed in some way and will take the appropriate action.

3.3.2.1.1 Parameters

None.

3.3.2.2 Invoking Entity (SCF)

3.3.2.2.1 Normal procedure

SCF Preconditions:

1) A relationship exists between the SCF and the SSF.

2) The activity test timer (Tati) expires, after which the ’ActivityTest’ operation is sent to the SSF.

SCF Postcondition:

– If a Return Result 'ActivityTest' is received, the SCME resets the activity test timer and takes no further action.

3.3.2.2.2 Error handling

If a time-out on the 'ActivityTest' operation or a P-Abort is received from TCAP, this is an indication that the relationship with the SSF was somehow lost. If a time-out is received, SCF aborts the dialogue.

The SLPI that was the user of this dialogue will be informed, the corresponding SCSM-FSM will move to the state "Idle".

3.3.2.3 Responding Entity (SSF)

3.3.2.3.1 Normal procedure

SSF Precondition:

– A relationship exists between the SCF and the SSF.

SSF Postconditions:

1) The SSME-FSM stays in, or moves to the state "Non Call Associated Treatment".

2) If the Dialogue ID is active and if there is a SSF-FSM using the dialogue, the SSME sends a Return Result 'ActivityTest' to the SCF. If there are no other management activities, the SSME-FSM returns to the state "Idle Management"; or

   If the Dialogue ID is not active, the TCAP in the SSF will issue a P-Abort, the SSME will in that case never receive the 'ActivityTest' req.ind and thus will not be able to reply.

3.3.2.3.2 Error handling

Operation related error handling is not applicable, due to class 3 operation.

3.3.3 AddEntry procedure

3.3.3.1 General description

The X.500 'AddEntry' operation is used to request the SDF to add a leaf entry (either an object entry or an alias entry) in the DIT. For a full description of the AddEntry operation, see 11.1/X.511.

3.3.3.1.1 Parameters

See 11.1.1/X.511 and 11.1.2/X.511.

3.3.3.2 Invoking Entity (SCF)

3.3.3.2.1 Normal procedure

SCF Precondition:

- SCSM: "SDF Bound" or "Wait for Subsequent Requests".

SCF Postcondition:

- SCSM: "SDF Bound".

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the service logic to add an entry in the SDF exists, an internal event [(e2) Request_to_SDF] occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response [(E5) Response_from_SDF_with_Bind or (E4) Bind_Error] to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the AddEntry operation was discarded. In the State "SDF Bound", the response of the AddEntry operation [(E7) Response_from_SDF] causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the AddEntry operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the service logic to add an entry in the SDF exists, an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response [(E7) Response_from_SDF] to the AddEntry operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the AddEntry operation or an error.

3.3.3.2.2 Error handling

Generic error handling for the operation related errors is described in 11.1.4/X.511 and 11.1.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.3.3 Responding Entity (SDF)

3.3.3.3.1 Normal procedure

SDF Precondition:

- SDSM: "SCF Bound"or "Bind Pending".

SDF Postcondition:

- SDSM "SCF Bound".

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of an 'AddEntry' operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of an 'AddEntry' operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before adding the new entry item, the SDF takes the following actions:

- verify that the superior object to which the entry should be added exists in the SDF;
- verify that the entry does not already exist in the SDF;
- verify that the access rights to add the entry and each of its components (attributes and values) are sufficient;
- verify that the entry conforms to the Directory schema.

After the specified actions indicated above are successfully executed, the entry is added into the SDF database. A null result is returned to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

3.3.3.3.2 Error handling

Generic error handling for the operation related errors is described in 11.1.4/X.511 and 11.1.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.4 AnalysedInformation Procedure

3.3.4.1 General description

This operation is sent by the SSF to the SCF after detecting a valid trigger condition at the Analysed Info DP, or to report an event requested by RequestReportBCSMEvent.

3.3.4.1.1 Parameters

– dPSpecificCommonParameters:

– dialledDigits:

See Recommendation Q.1290.

– callingPartyBusinessGroupID:

See Recommendation Q.1290. The SCF can use this IE to select SLPs based on the group and for authorization purposes. The network operators can specify that this IE should be used if their particular network has the information available.

– callingPartySubaddress:

See Recommendation Q.931.

– callingFacilityGroup:

See Recommendation Q.1290.

– callingFacilityGroupMember:

See Recommendation Q.1290.

– originalCalledPartyID:

See Recommendation Q.762. Original Called Number signalling information.

– prefix:

See Recommendation Q.1290.

– redirectingPartyID:

This parameter (if available) is the directory number of the last redirecting party.

– redirectionInformation:

See Recommendation Q.763 Redirection Information signalling information.

– routeList:

Route List represents the list of routes which would have been used in order to route the call. The network operators can specify that this IE should be used if their particular network has the information available.

– travellingClassMark:

See Recommendation Q.1290.

– featureCode:

See Recommendation Q.1290.

– accessCode:

See Recommendation Q.1290.

– carrier:

See Recommendation Q.1290.

3.3.4.2 Invoking Entity (SSF)

3.3.4.2.1 Normal procedure

SSF Preconditions (TDP):

1) Call origination attempt has been initiated.

2) Called Party Number is available and nature of address determined.

3) Call gapping or service filtering are not in effect.

4) DP criteria are met.

5) For a TDP-R, there is no existing control relationship influencing the call segment.

SSF Preconditions (EDP):

1) For an EDP-R, there is an existing control relationship and the EDP AnalysedInformation is armed.

2) For an EDP-N, there is a monitoring or control relationship.

SSF Postconditions (TDP):

1) For a TDP-R, basic call processing has been suspended at Analysed_Info DP, and a control relationship has been established.

2) For a TDP-N, basic call processing proceeds at Select_Route PIC, and no control relationship is established.

SSF Postconditions (EDP):

3) For an EDP-R, basic call processing has been suspended at Analysed_Info DP, and the existing control relationship continues.

4) For an EDP-N, basic call processing proceeds at Select_Route PIC, and the existing non-control relationship continues unless no further EDPs are armed and no "CallInformationReport or "ApplyChargingReport" are requested.

The SSF has sufficient information available associated with the originating call portion. This information has been analysed and the results are as described below:

1) From a non-ISDN line or ISDN interface, the Analysis Results consists of one or more of the following:

- CalledPartyID – The number used to identify the called party in the forward direction (i.e. it is used to populate the bearer signalling protocol's called party number information element).

- TypeOfCall – Indicates one of: interexchange carrier, international carrier, local exchange operator, interexchange carrier operator or international carrier operator.

- Carrier (for calls requiring an interexchange carrier) – Indicates the type of carrier requested.

- CarrierIdentificationCode (for calls requiring an interexchange carrier) – Indicates the code for the specific carrier to be used.

- CarrierSelection (for calls requiring an interexchange carrier) – Indicates whether the caller dialled the selected carrier and whether the caller pre-subscribed to the selected carrier.

- RouteIndex – An index to a route list (if the call does not terminate on this SSF).

- Collected Information – Access Codes and Prefixes, Collected Address Information/Digits.

2) From a conventional or SS No. 7-supported trunk interface, this consists of one or more of the following:

- ChargeNumber.
- CalledPartyID.
- TypeOfCall.
- CarrierIdentificationCode (for interexchange carrier calls).
- Carrier (for interexchange carrier calls).
- CarrierSelection (for interexchange carrier calls).
- RouteIndex.
- Collected Information – Collected Address Information, Prefixes, etc. The Collected Information for a call from an SS No. 7-supported trunk is based on information provided in the IAM.

3.3.4.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.4.3 Responding Entity (SCF)

3.3.4.3.1 Normal procedure

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.4.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.5 AnalyseInformation procedure

3.3.5.1 General description

This operation requests the SSF to perform the originating basic call processing actions to analyse destination information that is either collected from a calling party, or provided by the SCF. This includes actions to validate the destination information according to a specified dialling plan, and if valid, to determine call set-up information.

3.3.5.1.1 Parameters

– destinationRoutingAddress:

  Represents a list of called party numbers (primary and alternates).

– alertingPattern:

  See Recommendation Q.1290. It only applies if the network signalling support this parameter or if SSF is the terminating local exchange for the subscriber.

– iSDNAccessRelatedInformation:

  Carries the same information as the protocol element ISUP Access Transport parameter in Recommendation Q.762.

– originalCalledPartyID:

  See Recommendation Q.762 Original Called Number signalling information.

– callingPartyNumber:

  See Recommendation Q.762.

– callingPartysCategory:

  See Recommendation Q.762

– calledPartyNumber:

  See Recommendation Q.762. Either the calledPartyNumber or the destinationRoutingAddress shall be provided by the SCF in the AnalyseInformation operation.

– chargeNumber:

  See Recommendation Q.1290.

– travellingClassMark:

  See Recommendation Q.1290.

– carrier:

  See Recommendation Q.1290. In this message, the carrier selection field is null (00000000) and Carrier ID indicates the carrier to use for the call.

3.3.5.2 Invoking Entity (SCF)

3.3.5.2.1 Normal procedure

SCF Preconditions:

1) Call origination attempt has been initiated.
2) Authority/ability to place outgoing call has been verified.
3) Destination information is available in the SSF or provided by the SCF.
4) Basic call processing has been suspended at one of the following DPs:
   - Origination_Attempt_Authorized;
   - Collected_Info;
   - Analysed_Info;
   - Route_Select_Failure;
   - O_Called_Party_Busy;
   - O_No_Answer;
   - O_Disconnect (called party disconnect only).

5) A control relationship has been established and the SLPI is processing the incoming request.

SCF Postconditions:

1) SLPI execution is terminated if no monitoring is requested.

2) SLPI execution is suspended pending the monitored event occurring, if monitoring is requested.

The AnalyseInformation message requests the SSF-CCF to resume call origination processing taking into account the address, routing and billing information provided in the message parameters. Processing resumes at the ANALYSE_INFORMATION PIC.

3.3.5.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.5.3 Responding Entity (SSF)

3.3.5.3.1 Normal procedure

SSF Preconditions:

– A TDP or EDP request has been invoked.

SSF Postconditions:

– The SSF performs the call processing actions to analyse destination information.

The following message parameter configurations are supported and shall result in the SSF-CCF call processing actions specified:

1) No Carrier Specified:

- Switch-based carrier selection shall be performed.
- The called party address provided in the CalledPartyNumber parameter shall be subjected to trigger analysis. It should be noted that if the same called party address is returned, the same trigger may be detected and processed again at the Analysed Information TDP.

2) Carrier Specified:

- The SSF-CCF shall resume call processing at the ANALYSE INFORMATION PIC using the designated carrier (network operator specific).

The following additional requirements also apply to the AnalyseInformation message.

- The SSF-CCF shall accept called party address information contained in the CalledPartyNumber parameter that conforms to the public network numbering plan.
- If a route requires a TCM, the value in the TCM parameter shall be used. If a TCM is required and the TCM parameter is not supplied in the AnalyseInformation, the SSF-CCF shall derive the TCM in accordance with requirements that apply assuming no IN involvement with the call. (Network operator specific option).
- If the SCF does not designate a carrier and carrier selection is performed by the SSF-CCF, this indicator shall be set in the normal fashion as if IN involvement with the call had not occurred (network operator specific option).

AnalyseInformation with CallingPartysCategory and ChargeNumber:

- If the CallingPartysCategory parameter is included, that value shall be used as the DTMF digits or the ISDN-UP Calling Party's Category parameter in any subsequent signalling including signalling to an operator system via operator services signalling (network operator specific).
- When DTMF digits are needed for subsequent signalling, the decimal equivalent of the CallingPartysCategory value shall be used (e.g. "01000000" is equivalent to 11 digits of "64") (network operator specific).

- If the CallingPartysCategory parameter is not included, a proper failure indication shall be used in any subsequent DTMF or ISDN-UP signalling (network operator specific).

- If the ChargeNumber parameter is included, that value shall be used as the Calling party number on non-SS No. 7 trunks or the network specific ISDN-UP Charge Number parameter in any subsequent signalling including signalling to an operator system via Operator Services Signalling (network operator specific).

- If the ChargeNumber parameter is not included, the value "0 digits" shall be used for the ChargeNumber in any subsequent non-SS No. 7 trunk or network specific ISDN-UP signalling (network operator specific).

AnalyseInformation and O_Called_Party_Busy, O_No_Answer:

- When the SSF-CCF receives the AnalyseInformation message in response to an O_Called_Party_Busy message the SSF-CCF shall do the following:

  1) The SSF-CCF shall release any resources that were used to process the call between the ANALYSE_INFORMATION and ROUTING & ALERTING PICs.

  2) The SSF-CCF shall resume call processing at the ANALYSE_INFORMATION PIC, and process the message as described in this procedure description.

  3) If the originating access is DSS-1, then the SSF-CCF shall not send another CALL PROCeeding message. (INAP-DSS-1 interworking is for further study.)

- When the AnalyseInformation message is received in response to an O_No_Answer TDP-Request Message, the SSF-CCF shall do the following:

  1) The SSF-CCF, if it is providing audible ringing tone to the calling party, shall remove this tone.

  2) The SSF-CCF shall release any resources that were used to process the call between the ANALYSE_INFORMATION and ROUTING & ALERTING PICs.

  3) The SSF-CCF shall resume call processing at the ANALYSE_INFORMATION PIC, and process the message as described in this procedure description.

  4) If the originating access is DSS-1, then the SSF-CCF shall not send another CALL PROCeeding message (INAP-DSS-1 interworking is for further study.)

- When the Analyse Information message is received in response to an O_No_Answer EDP-Request message, the SSF-CCF shall do the following:

  1) The SSF-CCF, if it is providing audible ringing tone to the calling party, shall remove this tone.

  2) The SSF-CCF shall release any resources that were used to process the call between the ANALYSE_INFORMATION and ROUTING & ALERTING PICs.

  3) The SSF-CCF shall resume call processing at the ANALYSE_INFORMATION PIC, and process the message as described in this Recommendation.

  4) If the originating access is DSS-1, then the SSF-CCF shall not send another CALL PROCeeding message. (INAP-DSS-1 interworking is for further study.)

AnalyseInformation with OriginalCalledPartyID:

- When the SSF-CCF receives an AnalyseInformation message that contains the OriginalCalledPartyID parameter, the SSF-CCF shall map the received OriginalCalledPartyID parameter to, for example (ISUP interworking is for further study):

  1) The OriginalCalledNumber information element in the Facility Information Element if the SSF-CCF does not yet have a value for the OriginalCalledNumber AND if the SSF-CCF routes the call to an ISDN line (for Basic rate and Primary rate ISDN signalling).

2) The OriginalCalledNumber parameter in the IAM if the SSF-CCF does not already have a value for the OriginalCalledNumber AND if the SSF-CCF routes the call to an SS No. 7 trunk (for ISDN-UP signalling).

Other AnalyseInformation Requirements.

- If the SSF-CCF has associated a Business Group ID with the call and the call is routed over an SS No. 7 trunk, the SSF-CCF shall include this information in the Business Group parameter of the network specific ISDN-UP IAM (network operator specific).
- If the CalledPartyNumber or CallingPartyNumber parameter is included in the AnalyseInformation operation, that value shall be used subsequently for the call and the existing value replaced.

3.3.5.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.6 ApplyCharging procedure

3.3.6.1 General description

This operation is used for interacting from the SCF with the SSF charging mechanisms. The "ApplyChargingReport" operation provides the feedback from the SSF to the SCF.

As several connection configurations may be established during a call, a possibility exists for the "ApplyCharging" to be invoked at the beginning of each connection configuration, for each party.

The charging scenarios supported by this operation are 4.1 and 4.2 (refer to Appendix II/Q.1214, "Charging scenarios").

3.3.6.1.1 Parameters

– aChBillingChargingCharacteristics:

This parameter specifies the charging related information to be provided by the SSF and the conditions on which this information has to be reported back to the SCF via the "ApplyChargingReport" operation. Its contents is network operator specific.

– partyToCharge:

This parameter indicates the party in the call to which the "ApplyCharging" operation should be applied. If it is not present, then it is applied to the A-party.

3.3.6.2 Invoking Entity (SCF)

3.3.6.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) The SLPI has determined that an "ApplyCharging" operation has to be sent.

SCF Postconditions:

1) No FSM state transition.
2) The SLPI is expecting "ApplyChargingReport" operations from the SSF.

The SCSM-FSM is in state "Preparing SSF Instructions" or is in state "Queuing FSM". This operation is invoked by the SCF if a SLPI results in the request of interacting with the charging mechanisms within the SSF to get back information about the charging.

3.3.6.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.6.3 Responding Entity (SSF)

3.3.6.3.1 Normal procedure

SSF Preconditions:

– The SSF-FSM is in one of the following states:

- "Waiting for Instructions" (state c);
- "Waiting for End of User Interaction" (state d);
- "Waiting for End of Temporary Connection" (state e);
- "Monitoring" (state f);

or the assisting/hand-off SSF-FSM is in state:

- "Waiting for Instructions" (state b).

SSF Postconditions:

– No FSM state transition.

On receipt of this operation, the SSF sets the charging data using the information elements included in the operation and acts accordingly. In addition, the SSF will start the monitoring of the end of the connection configuration and other charging events, if requested.

3.3.6.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.7 ApplyChargingReport procedure

3.3.7.1 General description

This operation is used by the SSF to report charging related information to the SCF as requested by the SCF using the 'ApplyCharging' operation.

During a connection configuration the 'ApplyChargingReport' operation may be invoked on multiple occasions. For each call party and each connection configuration, the 'ApplyChargingReport' operation may be used several times. Note that at least one 'ApplyChargingReport' operation is to be sent at the end of the connection configuration charging process.

The charging scenarios supported by this operation are 4.1 and 4.2 (refer to Appendix II/Q.1214, 'Charging scenarios').

3.3.7.1.1 Parameters

– CallResult:

This parameter provides the SCF with the charging related information previously requested using the 'ApplyCharging' operation. The 'CallResult' will include the 'partyToCharge' parameter as received in the related 'ApplyCharging' operation to correlate the result to the request. The remaining content of 'CallResult' is network operator specific. Examples of these contents may be: bulk counter values, costs, tariff change and time of change, time stamps, durations, etc.

3.3.7.2 Invoking Entity (SSF)

3.3.7.2.1 Normal procedure

SSF Preconditions:

1) A control relationship exists between the SCF and the SSF.

2) A charging event has been detected that was requested by the SCF via an 'ApplyCharging' operation.

SSF Postconditions:

– If the connection configuration does not change, then no FSM state transition shall occur. If the connection configuration changes, then the FSM shall move to:

  - "Idle" state if there is no other EDP armed and no report requests are pending; or
  - otherwise, shall remain in the same state.

This operation is invoked if a charging event has been detected that was requested by the SCF. The 'ApplyChargingReport' operation only deals with charging events within the SSF itself. Examples of charging events may be: threshold value reached, timer expiration, tariff change, end of connection configuration, etc.

3.3.7.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.7.3 Responding Entity (SCF)

3.3.7.3.1 Normal procedure

SCF preconditions:

– An 'ApplyCharging' operation has been sent at the request of an SLPI and the SLPI is expecting an 'ApplyChargingReport' from the SSF.

SCF postconditions:

– No FSM state transition if further reports, including "EventReportBCSM" and "CallInformationReport", are expected, or Transition to the state "Idle" if the report is the last one and no "EventReportBCSM" or "CallInformationReport" is expected.

On receipt of this operation the SLPI which is expecting this operation will continue.

3.3.7.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 the TCAP services used for reporting operation errors are described in 3.4.

3.3.8 AssistRequestInstructions procedure

3.3.8.1 General description

This operation is sent to the SCF by an SSF, which is acting as the assisting SSF in an assist or handoff procedure, or by a SRF. The operation is sent when the assisting SSF or SRF receives an indication from an initiating SSF containing information indicating an assist or handoff procedure.

3.3.8.1.1 Parameters

– correlationID:

  This parameter is used by the SCF to associate the 'AssistRequestInstructions' from the assisting SSF or by a SRF with the request from the initiating SSF. The value of the 'correlationID' may be extracted from the digits received from the initiating SSF or be all of the digits.

– iPAvailable:

  See Recommendation Q.1290. This parameter is applicable to this operation only in the physical scenarios corresponding to assist with relay or handoff. The use of this parameter is network operator dependent.

– iPSSPCapabilities:

  See Recommendation Q.1290. This parameter is applicable to this operation only in the physical scenarios corresponding to assist with relay or handoff. The use of this parameter is network operator dependent.

3.3.8.2 Invoking Entity (SSF-SRF)

3.3.8.2.1 Normal procedure

SSF Precondition:

– An assist indication is detected by the assisting or Handoff SSF.

SSF Postcondition:

– The assisting or Handoff SSF waits for instructions.

On receipt of an assist indication from the initiating SSF, the SSF or SRF shall assure that the required resources are available to invoke an 'AssistRequestInstructions' operation in the SSF-SRF and indicate to the initiating SSF that the call is accepted (refer to Recommendation Q.71). The 'AssistRequestInstructions' operation is invoked by the SSF or SRF after the call, which initiated the assist indication, is accepted. The assisting SSF-FSM transitions to state "Waiting For Instructions".

3.3.8.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.8.3 Responding Entity (SCF)

3.3.8.3.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the initiating SSF in case of assist procedure.

2) The SCF waits for 'AssistRequestInstructions'.

SCF Postcondition:

– An SSF or SRF instruction is being prepared.

On receipt of this operation in the SCSM state "Waiting for Assist Request Instructions", the SCP has to perform the following actions.

If the 'AssistRequestInstructions' operation was received from an assisting SSF, and the resource is available, the SCSM prepares the 'ConnectToResource' and 'PlayAnnouncement' or 'PromptAndCollectUserInformation' to be sent to the assisting SSF.

The SCF determines SSF-SRF by means of 'correlationID', 'destinationNumber' or network knowledge.

If the 'AssistRequestInstructions' operation was received from a SRF, and the resource is available, the SCSM prepares the 'PlayAnnouncement' or 'PromptAndCollectUserInformation' to be sent to the SRF.

On receipt of this operation from the Handoff SSF, the SCSM associated with the Handoff SSF transits from the "Idle" state to the "Preparing SSF Instructions" state.

3.3.8.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.9 Bind procedure

3.3.9.1 General description

The X.500 'Bind' operation is used by the SDF to create an authenticated association between an SCF and an SDF on behalf of the end user. It carries the authentication information of the end user if any. For a full description of the Bind operation, see 8.1/X.511.

3.3.9.1.1 Parameters

See 8.1.2/X.511 and 8.1.3/X.511.

3.3.9.2 Invoking Entity (SCF)

3.3.9.2.1 Normal procedure

SCF Preconditions:

– SCSM: "Idle".

SCF Postconditions:

1) SCSM: "SDF Bound" in case of success.

2) SCSM: "Idle" in case of failure.

When the SCSM is in the state "Idle" and a need of the service logic to interrogate an SDF exists, an internal event occurs. This event, called (e1) Bind_Request, causes a transition to the state "Wait for Subsequent Request" and other operations are awaited. Until the application process has not indicated by a delimitor that the Bind should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimitor causes a transition to the state "Wait for Bind result" through the internal transition (e3) Request_to_SDF_with_Bind. The operation is sent to the SDF. The SCSM waits for the response from the SDF. The reception of the response [(E5) Response_from_SDF_with _Bind] to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" if the result of the Bind operation is positive. Otherwise the reception of an error [(E4) Bind_Error] moves back the SCSM to the state "Idle".

3.3.9.2.2 Error handling

Generic error handling for the operation related errors is described in 8.1.4/ X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.9.3 Responding Entity (SDF)

3.3.9.3.1 Normal procedure

SDF Preconditions:

– SDSM: "Idle".

SDF Postconditions:

1) SDSM: "SCF Bound" (success).

2) SDSM: "Idle" (failure).

The SDF is initially in the state "Idle". After accepting the external event (E1) Bind_from_SCF caused by the reception of a 'Bind' operation from the SCF, a transition to state "Bind Pending" occurs. The SDF performs the Bind operation according to the contents of the Bind argument. Once the SDF has completed the 'Bind' operation, the result or error indication is returned to the SCF. The SDF returns to the state "Idle" if the Bind fails or to the state "SCF Bound" if the Bind is successful. Should the Bind request succeed, the result returned may consist of credentials of the DirectoryBindResult. These credentials allow the user to establish the identity of the Directory. They allow information identifying the DSA (that is directly providing the Directory service) to be conveyed to the DUA. The credentials are of the same form (i.e. name, password) as those supplied by the user.

3.3.9.3.2 Error handling

Generic error handling for the operation related errors is described in 8.1.4/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.10 CallGap procedure

3.3.10.1 General description

This operation is used to request the SSF to reduce the rate at which specific service requests are sent to the SCF.

3.3.10.1.1 Parameters

a) gapCriteria:

This parameter identifies the criteria for a call to be subject to call gapping.

i) calledAddressValue:

This parameter indicates that call gapping will be applied when the leading digits of the dialled number of a call attempt match those specified in "gapCriteria".

ii) gapOnService:

This parameter indicates that call gapping will be applied when the "servicekey" of a call attempt match those specified in "gapCriteria".

iii) calledAddressAndService:

This parameter indicates that call gapping will be applied when the "serviceKey" and the leading digits of the dialled number of a call attempt match those specified in "gapCriteria".

iv) callingAddressAndService:

This parameter indicates that call gapping will be applied when the "serviceKey" and the leading digits of the calling party number or the location number of a call attempt match those specified in "gapCriteria".

b) gapIndicators:

This parameter indicates the gapping characteristics.

i) duration:

Duration specifies the total time interval during which call gapping for the specified gap criteria will be active.

A duration of 0 indicates that gapping is to be removed.

A duration of –1 indicates an infinite duration.

A duration of –2 indicates a network specific duration.

Other values indicate duration in seconds.

ii) gapInterval:

This parameter specifies the minimum time between calls being allowed through.

An interval of 0 indicates that calls meeting the gap criteria are not to be rejected.

An interval of –1 indicates that all calls meeting the gap criteria are to be rejected.

Other values indicate interval in milliseconds.

c) controlType:

This parameter indicates the reason for activating call gapping.

The "controlType" value "sCPOverloaded" indicates that an automatic congestion detection and control mechanism in the SCP has detected a congestion situation.

The "controlType" value "manuallyInitiated" indicates that the service and or network/service management centre has detected a congestion situation, or any other situation that requires manually initiated controls[2)].

---

2) The controlType "manuallyInitiated" will have priority over "sCPOverloaded" call gap.

It should be noted that also non-IN controlled traffic control mechanism can apply to an exchange with the SSF functionality. The non-IN controlled traffic control may also have some influence to the IN call. Therefore it is recommended to take measures to coordinate several traffic control mechanisms. The non-IN controlled traffic control and coordination of several traffic control mechanisms are out of the scope of INAP.

d) gapTreatment:

This parameter indicates how calls that were stopped by the call gapping mechanism shall be treated.

i) informationToSend:

This parameter indicates an announcement, a tone or display information to be sent to the calling party. At the end of information sending, the call shall be released.

- inbandInfo:

  This parameter specifies the inband information to be sent.

  – messageID:

    This parameter indicates the message(s) to be sent, it can be one of the following:

    1) elementaryMessageID:

       This parameter indicates a single announcement.

    2) text:

       This parameter indicates a text to be sent. The text shall be transformed to inband information (speech). This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

    3) elementaryMessageIDs:

       This parameter specifies a sequence of announcements.

    4) variableMessage:

       This parameter specifies an announcement with one or more variable parts.

  – numberOfRepetitions:

    This parameter indicates the maximum number of times the message shall be sent to the end-user.

  – duration:

    This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

  – interval:

    This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

- tone:

  This parameter specifies a tone to be sent to the end-user.

  – toneID:

    This parameter indicates the tone to be sent.

  – duration:

    This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

- displayInformation:

  This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

ii) releaseCause:

This parameter indicates that the call shall be released using the given release cause. See Recommendation Q.762.

iii) both:

This parameter indicates inband info, a tone or display information to be sent to the calling party. At the end of information sending, the call shall be released, using the given release cause.

3.3.10.2 Invoking Entity (SCF)

3.3.10.2.1 Normal procedure

SCF Preconditions:

- The SCF detects an overload, condition persists and call gapping has to be initiated at the SSF; or
- the SCF receives a manually initiated call gapping request.

SCF Postcondition:

- The SCME-FSM remains in the same state upon issuing the "CallGap" operation.

A congestion detection and control algorithm monitors the load of SCP resources. After detection of a congestion situation the parameters for the "CallGap" operation are provided.

If the congestion level changes, new 'CallGap' operations may be sent for active gap criteria but with new gap interval. If no congestion is detected, gapping may be removed.

A manual initiated call gap will take prevail over an automatic initiated call gap.

3.3.10.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.10.3 Responding Entity (SSF)

3.3.10.3.1 Normal procedure

SSF Preconditions:

- call gapping for gapCriteria is not active; or
  call gapping for gapCriteria is active.

SSF Postconditions:

1) the SSME-FSM is in the state "Non call associated treatment";
2) call gapping for gapCriteria is activated; or
   call gapping for gapCriteria is renewed; or
   call gapping for gapCriteria is removed.

If there is no already existing SSME-FSM for the gap criteria provided, a new SSME-FSM is created. This SSME-FSM enters the state 'Non-call associated treatment' and initializes call gapping for the specified IN calls. The parameters 'gapIndicators', 'controlType' and 'gapTreatment' for the indicated gap criteria will be set as provided by the 'CallGap' operation.

In general, the manuallyInitiated call gapping will prevail over automatically initiated ('sCPOverloaded'). More specifically, the following rules will be applied in the SSF to manage the priority of different control Types associated with the same 'gapCriteria':

- If an SSME-FSM already exists for the 'gapCriteria' provided, then:
  1) If the (new) 'controlType' equals an existing 'controlType', then the new parameters (i.e. 'gapIndicators' and 'gapTreatment') will overwrite the existing parameter values.
  2) If the (new) 'controlType' is different than the existing 'controlType', then the new parameters (i.e. 'controlType', 'gapIndicators', and 'gapTreatment') will be appended to the appropriate SSME-FSM (in addition to the existing parameters). The SSME-FSM remains in the state "Non-Call Associated Treatment".

If the SSF meets a TDP, it will check if call gapping was initiated either for the 'serviceKey' or for the 'calledAddressValue' assigned to this TDP. If not, an 'InitialDP' or a DP-specific operation can be sent. In case call gapping was initiated for 'calledAddressAndService' or 'callingAddressAndService' and the 'serviceKey' matches, a

check on the 'calledAddressValue' and 'callingAddressValue' – and optionally 'locationNumber' – for active call gapping is performed. If not, an 'InitialDP' or a DP-specific operation can be sent.

In case of gapping on 'callingAddressAndService' and the parameter 'locationNumber' is present, gapping will be performed on 'locationNumber' instead of 'callingAddressValue'.

If a call to a controlled number matches only one 'gapCriteria', then the corresponding control is applied. If both 'manuallyInitiated' and 'sCPOverload' controls are active, then only the manually initiated control will be applied.

If a call to a controlled called number matches several active 'gapCriteria', then only the 'gapCriteria' associated with the longest called party number should be used, and the corresponding control should be applied. For example, the codes 1234 and 12345 are under control. Then the call with 123456 is subject to the control on 12345. Furthermore, if both 'manuallyInitiated' and 'sCPOverloaded' 'controlTypes' are active for this 'gapCriteria', then the 'manuallyInitiated' control will be applied.

If call gapping shall be applied and there is no gap interval active, an "InitialDP" or a DP-specific operation can be sent including the "cGEncountered" parameter according to the specified controlType. A new gap interval will be initiated as indicated by "gapInterval".

If a gap interval is active, no "InitialDP" or a DP-specific operation is sent and the call is treated as indicated by "gapTreatment".

The call gap process is stopped if the indicated duration equals ZERO.

If call gapping proceeds then the SSME-FSM remains in the state "Non-call associated treatment". Otherwise, the SSME-FSM moves to state "idle management".

3.3.10.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.11 CallInformationReport procedure

3.3.11.1 General description

This operation is used to send specific call information for a single call to the SCF as requested by the SCF in previous 'CallInformationRequest' operation. The report is sent at the end of a call which is indicated by one of the events specified below.

3.3.11.1.1 Parameters

– requestedInformationList:

According to the requested information, the SSF sends the appropriate types and values to the SCF.

3.3.11.2 Invoking Entity (SSF)

3.3.11.2.1 Normal procedure

SSF Preconditions:

1) At least one party disconnects from a call or call set-up is not completed.

2) Requested call information has been collected.

3) 'CallInformationReport' is pending due to a previously received 'CallInformationRequest' operation.

4) A control relationship exists between the SCF and the SSF.

SSF Postcondition:

– The SSF-FSM shall move to the "Idle" state in the case where no other report requests are pending and no EDPs are armed, otherwise the SSF-FSM shall remain in the same state.

If the SSF-FSM executes a state transition caused by one of the following events:

- A party release;
- A party abandon;
- B party release;
- B party busy;
- SSF no answer timer expiration;
- route select failure indicated by the network;
- release call initiated by the SCF;

and 'CallInformationRequest' is pending, then one 'CallInformationReport' operation is sent to the SCF containing all information requested.

If a 'CallInformationReport' has been sent to the SCF then no 'CallInformationReport' is pending, i.e. a further 'CallInformationReport', for example in the case of follow-on, has to be explicitly requested by the SCF.

If an event causing the 'CallInformationReport' is also detected by an armed EDP-R, then immediately after 'CallInformationReport', the corresponding 'EventReportBCSM' has to be sent.

If an event causing the 'CallInformationReport' is also detected by an armed EDP-N, then immediately before 'CallInformationReport', the corresponding 'EventReportBCSM' has to be sent.

3.3.11.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.11.3 Responding Entity (SCF)

3.3.11.3.1 Normal procedure

SCF Preconditions:

1) An SLPI is expecting 'CallInformationReport'.
2) A control relationship exists between the SCF and the SSF.

SCF Postcondition:

- The SLPI may be further executed.

In any state (except 'Idle') the SCSM may receive 'CallInformationReport' from the SSF, when the 'CallInformationReport' is outstanding.

If 'CallInformationReport' is outstanding and the service logic program indicates that the processing has been completed, the SCSM remains in the same state until it receives the 'CallInformationReport' operation.

When the SCF receives the 'CallInformationReport' operation and the service logic processing has been completed, then the SCSM moves to the "Idle" state.

When the SCF receives the 'CallInformationReport' operation and the service logic processing has not been completed yet, then the SCSM remains in the same state (EventReportBCSM or ApplyChargingReport pending).

3.3.11.3.2 Error handling

If requested information is not available, a 'CallInformationReport' will be sent, indicating the requested information type, but with 'RequestedInformationValue' filled in with an appropriate default value as specified by the network operator.

Operation related error handling is not applicable, due to class 4 operation.

3.3.12 CallInformationRequest procedure

3.3.12.1 General description

This operation is used to request the SSF to record specific information about a single call and report it to the SCF using the ’CallInformationReport’ operation.

3.3.12.1.1 Parameters

– requestedInformationTypeList:

This parameter specifies a list of specific items of information which is requested.

The list may contain:

- callAttemptElapsedTime:

  This parameter indicates the duration between the end of INAP processing of operations initiating call set-up ('Connect', 'AnalyseInformation', 'CollectInformation', 'Continue' and 'SelectRoute') and the received answer indication from the called party side.

  In case of unsuccessful call set-up the network event indicating the unsuccessful call set-up stops the measurement of 'callAttemptElapsedTime'.

- callStopTime:

  This parameter indicates the time stamp when the connection is released.

- callConnectedElapsedTime:

  This parameter indicates the duration between the received answer indication from the called party side and the release of the connection.

- calledAddress

  This parameter indicates the incoming called party address that was received by the SSF (i.e. before translation by the SCF) and is as available on the UNI or NNI and interpreted as per the numbering plan.

- releaseCause:

  See Recommendation Q.762.

Any set of these values can be requested.

3.3.12.2 Invoking Entity (SCF)

3.3.12.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) The SLPI has determined that a 'CallInformationRequest' operation has to be sent by the SCF.

SCF Postcondition:

– The SLPI is expecting a 'CallInformationReport' from SSF.

When the service logic program requests call information, the SCF sends the 'CallInformationRequest' operation to the SSF to request the SSF to provide call related information.

The 'CallInformationRequest' operation specifies the information items to be provided by the SSF.

3.3.12.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.12.3 Responding Entity (SSF)

3.3.12.3.1 Normal procedure

SSF Preconditions:

1) Call origination attempt has been initiated.

2) A control relationship exists between SSF and SCF.

SSF Postconditions:

1) Requested call information is retained by the SSF.

2) The SSF is waiting for further instructions.

The SSF may receive the "CallInformationRequest" operation within an existing Call Associated (CA) dialogue only.

The "CallInformationRequest" operation is accepted by the SSF Finite State Machine (SSF-FSM) only in the state "Waiting for Instructions". The operation does not lead to any transition to another state.

The SSF allocates a record and stores the requested information if already available and prepares the recording of information items, that will become available later like for example "callStopTimeValue".

3.3.12.3.2 Error handling

In any other than the "Waiting for Instruction" state the "CallInformationRequest" operation will be handled as an error with the error code "UnexpectedComponentSequence".

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.13 Cancel procedure

3.3.13.1 General description

The SCF uses this class 2 operation to request the SRF-SSF to cancel a correlated previous operation.

The SRF operation to be deleted can be either a 'PlayAnnouncement' operation or a 'PromptAndCollectUserInformation' operation.

The cancellation of an operation is indicated via a respective error indication, "Cancelled", to the invoking entity of the cancelled "PlayAnnouncement" or "PromptAndCollectUserInformation" operation. The 'Cancel' operation can also be used to cancel all outstanding requests and enable the state machine (SSF) to go to idle. In this case the 'Cancel' operation does not specify any specific operation to be cancelled.

3.3.13.1.1 Parameters

– invokeID:

  This parameter specifies which operation is to be cancelled, i.e. PromptAndCollectUserInformation or PlayAnnouncement.

– allRequests:

  This parameter indicates that all active requests for EDP reports (generic or DP specific), "ApplyChargingReport" and "CallInformationReport" should be cancelled.

  NOTE – This cancellation is different from the invokeID based cancel mechanism described above.

3.3.13.2 Invoking Entity (SCF)

3.3.13.2.1 Normal procedure

The SCF may either invoke this operation to the SSF or to the SRF, different conditions will prevail in each case.

SCF Preconditions:

1) a control relationship exists between the SCF and the SSF-SRF;

2) an SLPI in the "Waiting for response from SRF" state has determined that a previously requested operation is to be cancelled; or

   an SLPI has determined that it is no longer interested in any reports or notifications from the SSF and that the control relationship should be ended.

SCF Postcondition:

– the SLPI remains in the "Waiting for Response from SRF" state; or

  in case all requests are cancelled, the control relationship with the concerned FE (SSF) is ended and the SCSM-FSM returns to "Idle" state.

3.3.13.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.13.3 Responding Entity (SRF)

3.3.13.3.1 Normal procedure

SRF Precondition:

– A PlayAnnouncement or PromptAndCollectUserInformation operation has been received and the SRF is in the "User Interaction" state.

SRF Postcondition:

– The execution of the PlayAnnouncement or PromptAndCollectUserInformation operation has been aborted and the SRF remains in the "User Interaction" state.

3.3.13.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.13.4 Responding Entity (SSF)

3.3.13.4.1 Normal procedure

SSF Precondition:

– The SSF-FSM is in the state "Waiting for Instructions" or "Monitoring".

SSF Postcondition:

1) all active requests for reports and notifications have been cancelled;

2) in case the SSF-FSM was in state "Monitoring" it shall return to idle; or
   in case the SSF-FSM was in state "Waiting for Instructions" it will remain in that state.

   A subsequent call-processing operation will move the SSF-FSM state to "Idle". The call, if in active state, is further treated by SSF autonomously as a normal (non-IN-) call.

3.3.13.4.2 Error handling

Sending of return error on cancel is not applicable in the cancel "allRequests" case. Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.14 CollectedInformation procedure

3.3.14.1 General description

This operation is sent by the SSF after detection of a TDP-R, TDP-N, EDP-R, or EDP-N in the originating BCSM, to indicate availability of complete initial package/dialling string from the originating party.

3.3.14.1.1 Parameters

– serviceAddressInformation (triggerType, serviceKey, miscCallInfo):

See Recommendation Q.1290. The currently defined events applicable to the Collected_Information DP are Off_Hook_Delay, Channel_Setup_PRI, and Shared_Interoffice_Trunk. The miscCallInfo indicates that a request for instructions has been issued to the SCF and the category for the trigger (line, group, office). The serviceKey is used in identifying the service logic to be invoked.

– bearerCapability:

This parameter indicates the type of bearer capability connection to the user. Refer to 6.4.4/Q.1214 for population rules for the bearer capability parameter.

– calledPartyNumber:

See Recommendation Q.762 Called Party Number signalling information. This parameter is used to identify the called party in the forward direction.

– callingPartyNumber:

See Recommendation Q.762 Calling Party Number signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartyNumber parameter.

– callingPartysCategory:

See Recommendation Q.762 Calling Party Category signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartysCategory parameter.

– iPSSPCapabilities:

See Recommendation Q.1290.

– iPAvailable:

See Recommendation Q.1290.

– iSDNAccessRelatedInformation:

This parameter contains (possibly multiple) information elements as per Recommendation Q.931. See Access Transport Parameter signalling information in Recommendations Q.762, Q.763 and Q.931. Refer to 6.4.4/Q.1214 for population rules for ISDN access related information.

– cGEncountered:

See Recommendation Q.1290.

– locationNumber:

This parameter is used to convey the geographical area address for mobility services, see Recommendation Q.762. It is used when the 'callingPartyNumber' does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

– serviceProfileIdentifier:

See Annex A/Q.932. See 6.4.4/Q.1214 for population rules for serviceProfileIdentifier.

– terminalType:

See Recommendation Q.1290. Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

– chargeNumber:

See Recommendation Q.1290. See 6.4.4/Q.1214 for population rules for chargeNumber.

– servingAreaID:

See Recommendation Q.1290.

– dialledDigits:

See Recommendation Q.1290. See 6.4.4/Q.1214 for population rules for dialledDigits.

– callingPartyBusinessGroupID:

See Recommendation Q.1290.

– callingPartySubaddress:

See Recommendation Q.931 for details.

– callingFacilityGroup:

See Recommendation Q.1290.

– callingFacilityGroupMember:

See Recommendation Q.1290.

– originalCalledPartyID:

See Recommendation Q.762 Original Called Number signalling information. Refer to 6.4.4/Q.1214 for population rules for the originalCalledPartyID parameter.

– prefix:

See Recommendation Q.1290.

– redirectingPartyID:

Contains the directory number of the last redirecting party.

– redirectionInformation:

See Recommendation Q.763 Redirection Information signalling information.

– travellingClassMark:

See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules for the travellingClassMark parameter.

– featureCode:

See Recommendation Q.1290.

– accessCode:

See Recommendation Q.1290.

– carrier:

See Recommendation Q.1290.

3.3.14.2 Invoking Entity (SSF)

3.3.14.2.1 Normal procedure

SSF Preconditions for TDP-R and TDP-N:

1) An event resulting in trigger criteria being met has occurred at the Collected_Information DP (e.g. Off_Hook_Delay, Channel_Setup_PRI, Shared_Interoffice_Trunk).
2) Call gapping and SS No. 7 overload are not in effect for the call, and the call is not to be filtered.

SSF Preconditions for EDP-R and EDP-N:

1) A control relationship has been established and the SCF requests Collected_Information as EDP-R or EDP-N.
2) Call processing reaches the Collected_Information DP.

SSF Postcondition for TDP-R:

– A control relationship has been established and the SSF waits for instructions from the SCF.

SSF post-condition for EDP-R:

– The control relationship continues and the SSF waits for instructions from the SCF.

SSF post-condition for TDP-N and EDP-N:

– Call processing continues at the Analysed_Information PIC.

The SSF/CCF collects enough information (e.g. a feature code, prefix, address information, etc.) from the originating access to process the call. This information is collected according to the dialling plan assigned to the originating access:

1) For a call from a non-ISDN line, the SSF-CCF sends an in-band prompt (e.g. dial tone) to prompt the user, attaches a digit receiver, collects digits and applies inter-digit timing. From a non-ISDN line, the "dialling plan in force" is the dialling plan assigned to the line. For example, if a Private Dialling Plan (PDP) (sometimes referred to as a Customized Dialling Plan) is assigned, then that PDP is considered to be the "dialling plan in force". If, on a given call, the caller dials an access code to escape to a Public Office Dialling Plan (PODP) (i.e. Recommendation E.164), then that PODP is considered to be the "dialling plan in force". In this case, call processing can not return to the PDP at this SSF-CCF. If a PDP is not assigned, then a PODP is assumed to be the "dialling plan in force".

2) For a call from an ISDN user following *en bloc* sending procedures (i.e. all the information necessary to process the call, which can include a feature code, feature activator, called party number, etc., is included in the SETUP message), no further action is taken on the originating access interface. For a call from an ISDN user not following *en bloc* sending procedures, the SSF-CCF prompts the user with a SETUP ACKnowledge message to send more information. Overlap sending procedures are then followed.

   For a call from an ISDN BRI or PRI, the "dialling plan in force" is determined by the type of number and numbering plan field of the called party number information element. If this field indicates "unknown" or if the keypad information element is used, then the dialling plan is determined as defined for non-ISDN lines above.

3) For conventional trunks, if required by the trunk type, an appropriate start signal is returned. A digit receiver is attached, digits are collected and inter-digit timing is applied.

   For a call from a conventional trunk, the number received in signalling is expected to be a non-private number that conforms to Recommendation E.164.

4) For SS No. 7 trunks, all the information may be available in the IAM or may come in subsequent messages (i.e. overlap sending). A check is made to ensure that the IAM contains all the information necessary to process the call.

   For a call from an SS No. 7 trunk, the number received in signalling is expected to be a non-private number that conforms to Recommendation E.164.

5) For private-facility trunks, if required by the private-facility trunk group, an appropriate start signal is returned. Depending on the private-facility trunk group, digits may be collected.

Following the trigger detection (due to the DP criteria assigned being met) related to an armed TDP-R or TDP-N in the BCSM at the Collected_Information DP, caused by a call origination attempt, the SSF checks if call gapping, SS No. 7 overload, or service filtering are not in effect for the related call segment. If these conditions are met, then the CollectedInformation operation is invoked by the SSF. The address of the SCF to which the CollectedInformation has to be sent is determined on the basis of trigger instance related data. The SSF provides parameters as per rules defined in 6.4.4/Q.1214. For TDP-R, a control relationship is established to the SCF and the SSF application timer T_{SSF} is set when the SSF sends the CollectedInformation operation for requesting instructions from the SCF. T_{SSF} is used to prevent excessive call suspension time. For EDP-R, T_{SSF} is also used. For TDP-N and EDP-N, no new control relationship is established. Hence, T_{SSF} is not set.

3.3.14.2.2 Error handling

If the destination SCF is not accessible, then the call is given final treatment (other treatments are for further study). On expiration of T_{SSF} before receiving any operation, the SSF aborts the interaction with the SCF and the call is given final treatment (e.g. routing to a final announcement). If the calling party abandons after the sending of the CollectedInformation operation, then the SSF aborts the control relationship after the first answer message from the SCF has been received. If the caller abandons the call and T_{SSF} is running, then the Transaction ID is held open until T_{SSF} is satisfied or expires. If the caller abandons the call and Timer TSSF is not running, then procedures to support the encountered situation (e.g. procedures to report the condition of Caller Abandon, procedures associated with the condition of T_{SSF} Timer Expired), shall be followed.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.14.3 Responding Entity (SCF)

3.3.14.3.1 Normal procedure

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

For TDP-R, on receipt of the CollectedInformation operation, the SCSM moves from "Idle" state to the state "Preparing SSF Instructions". A control relationship to the related SSF is created. A Service Logic Program Instance (SLPI) is invoked for process the CollectedInformation operation based on the triggerType parameter. By means of this control relationship, the SCF may influence the Basic Call Processing in accordance with the service logic invoked. The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the SLPI (i.e. the requested IN service itself).

For EDP-R, on receipt of the CollectedInformation operation, the SCSM moves from "Waiting for Notification" state to the state "Preparing SSF Instructions". A control relationship to the related SSF continues. A Service Logic Program Instance (SLPI) is invoked for process the CollectedInformation operation based on the triggerType parameter. By means of this control relationship, the SCF may influence the Basic Call Processing in accordance with the service logic invoked. The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the SLPI (i.e. the requested IN service itself).

For TDP-N, the SCSM remains in state "Idle" and takes appropriate action on the notification.

For EDP-N, if this is the last event notification and there are no CallInformationReport and ApplyChargingReport pending, the SCSM returns to state "Idle". Otherwise, it remains in state "Waiting for Notification or Report". Appropriate actions on the notification are taken.

3.3.14.3.2 Error handling

If the CollectedInformation operation is rejected, then the SCSM remains in the same state. The maintenance function is informed and no SLPI is invoked. Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.15 CollectInformation procedure

3.3.15.1 General description

The CollectInformation is a class 2 operation which is used by the SCF to request the call to return to the Collect_Information PIC, and then perform the basic originating call processing actions associated with this PIC (e.g. the checking of information in the CalledPartyNumber parameter with the supported dialling plan). This operation uses only the resources of the SSF-CCF to collect the information. The use of this operation is only appropriate for a call which had not yet left the set-up phase.

When the user provides calledPartyNumber, Collect_Information PIC processing includes collecting of destination information from a calling party. When the calledPartyNumber is included (as dialledDigits) in the Collect_Information operation, further collection is not performed (e.g. SSF-CCF checks the provided calledPartyNumber against the supported dialling plan).

3.3.15.1.1 Parameters

– alertingPattern:

  See Recommendation Q.1290. It only applies if the network signalling support this parameter or if SSF is the terminating local exchange for the subscriber.

– numberingPlan:

  See Recommendation Q.763 Numbering Plan for encoding.

– originalCalledPartyID:

  See Recommendation Q.762 Original Called Number signalling information.

– travellingClassMark:

  See Recommendation Q.1290. This parameter contains the travelling class mark information of the calling party. It uses the LocationNumber format which is based on the Q.763 Location Number format.

– callingPartyNumber:

  See Recommendation Q.762.

– dialledDigits:

  This parameter is applied against the supported dialling plan in the SSF-CCF and, if valid, is used in routing of the call. If provided, it replaces the calledPartyNumber for the call.

3.3.15.2 Invoking Entity (SCF)

3.3.15.2.1 Normal procedure

SCF Precondition:

– An SLPI has determined that more information from the calling party is required to enable processing to proceed.

SCF Postcondition:

– SLPI execution is suspended pending receipt of dialled digits.

This operation is invoked in the SCSM-FSM state "Preparing SSF Instructions" if the SLP requires additional information to progress the call. It causes a transition of the FSM to the state "Waiting for Notification or Report".

3.3.15.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.15.3 Responding Entity (SSF)

3.3.15.3.1 Normal procedure

SSF Precondition:

– A TDP Request has been invoked.

SSF Postconditions:

1) The SSF has executed a transition to the state "Monitoring".

2) The SSF performs the call processing actions to collect destination information from the calling party. This may include prompting the party with in-band or out-band signals.

3) Basic call processing is resumed at PIC 2.

The operation is only valid in the state "Waiting for Instruction" and after having received an operation 'RequestReportBCSMEvent' for DP2. The SSP has to perform the following actions:

– The SSF cancels T_{SSF}.

– When DP2 will be encountered, an 'EventReportBCSM' operation will be invoked, and the SSF-FSM will return to the state "Waiting for Instructions".

3.3.15.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4 .

3.3.16 Connect procedure

3.3.16.1 General description

This operation is used to request the SSF to perform the call processing actions to route a call to a specific destination. To do so, the SSF may use destination information from the calling party (e.g. dialled digits) and existing call set-up information (e.g. route index to a list of trunk groups) depending on the information provided by the SCF.

In general, all parameters which are provided in a Connect operation to the SSF shall replace the corresponding signalling parameter in the CCF, if the relevant parameter has already been received in the CCF, and shall be used for subsequent call processing. Parameters which are not provided by the Connect operation shall retain their value (if already assigned) in the CCF for subsequent call processing.

3.3.16.1.1 Parameters

– destinationRoutingAddress:

This parameter contains the list of called party numbers (see Recommendation Q.762) towards which the call is to be routed. The encoding of the parameter is defined in Recommendation Q.763. The 'destinationRoutingAddress' may include the 'correlationID' and 'scfID' if used in the context of a hand-off procedure, but only if 'correlationID' and 'scfID' are not specified separately.

– alertingPattern:

See Recommendation Q.1290. It only applies if the network signalling support this parameter or if SSF is the terminating local exchange for the subscriber.

– correlationID:

This parameter is used by the SCF to associate the 'AssistRequestInstructions' operation from the assisting SSF with the Request from the initiating SSF. The 'correlationID' is used in the context of a hand-off procedure and only if the correlation id is not embedded in the 'destinationRoutingAddress'. The network operators has to decide about the actual mapping of this parameter on the used signalling system.

– cutAndPaste:

This parameter is used by the SCF to instruct the SSF to delete (cut) a specified number of leading digits that it has received from the calling party and to paste the remaining dialled digits on to the end of the digits supplied by the SCF in the 'destinationRoutingAddress'.

– forwardingCondition:

Indicates the condition that must be met to complete the Connect operation.

– iSDNAccessRelatedInformation:

Carries the same information as the protocol element ISUP Access Transport parameter in Recommendation Q.762.

– originalCalledPartyID:

See Q.762 Original Called Number signalling information. The use of this parameter in the context of the 'Connect' operation is to be specified by the network operator.

– routeList:

This parameter is used to select the outgoing trunk group used for routing the call. A sequence of routes is provided to allow flexible routing for applications such as VPN without increasing the number of queries required for such applications.

– scfID:

See Recommendation Q.1290. The scfID is used in the context of a hand-off procedure and only if the SCF id is not embedded in the 'destinationRoutingAddress'. The network operator has to decide about the actual mapping of this parameter on the used signalling system.

– travellingClassMark:

The SCF uses the travellingClassMark parameter to provide essential route selection information to the SSF. The SSF uses this information to populate the outgoing ISUP-IAM message, the population and mapping of this parameter is network operator specific.

– carrier:

See Recommendation Q.1290. In this message, the carrier selection field is null (00000000) and Carrier ID indicates the carrier to use for the call.

– serviceInteractionIndicators:

This parameter contains indicators sent from the SCF to the SSF for control of the network based services at the originating exchange and the destination exchange.

– callingPartyNumber:

This parameter, if present, is used to identify the calling party for the call (see Q.762 Calling Party Number). It may be used for applications such as UPT, where only the SCF can verify the identity of the calling party.

– callingPartysCategory:

See Recommendation Q.762 Calling Party Category signalling information.

– redirectingPartyID:

This parameter, if present, indicates the last directory number the call was redirected from.

– redirectionInformation:

See Recommendation Q.763 Redirection Information signalling information.

3.3.16.2 Invoking Entity (SCF)

3.3.16.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) An SLPI has determined that a 'Connect' has to be sent by the SCF.

SCF Postcondition:

– SLPI execution may continue.

In the SCSM-FSM state "Preparing SSF Instructions", this operation is invoked by the SCF if the service logic results in the request to the SSF to route a call to a specific destination. If no event monitoring has been requested and no reports (CallInformationReport and ApplyChargingReport) have been requested in a previously sent operation, a SCSM-FSM transition to state "Idle" occurs. Otherwise, if event monitoring has been requested or any report (CallInformationReport and ApplyChargingReport) has been requested, the SCSM-FSM transitions to state "Waiting for Notification or Report". When the 'Connect' operation is used in the context of a hand-off procedure, the SCSM-FSM transitions to state "Idle". However, in this case, the SCF must maintain sufficient information in order to correlate the subsequent 'AssistRequestInstructions' operation (from the assisting SSF or SRF) to the existing SLPI.

3.3.16.2.2 Error handling

If reject or error messages are received, then the SCSM informs the SLPI and remains in the state "Preparing SSF Instructions".

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.16.3 Responding Entity (SSF)

3.3.16.3.1 Normal procedure

SSF Preconditions:

1) Call origination attempt has been initiated.

2) Basic call processing has been suspended at a DP.

3) The SSF waits for instructions.

SSF Postconditions:

1) The SSF performs the call processing actions to route the call to the specified destination.

2) In the O-BCSM, when only address information is included in the Connect operation, call processing resumes at PIC 3.

3) In the O-BCSM, when address information and routing information is included in the Connect operation, call processing resumes at PIC 4.

On receipt of this operation in the SSF-FSM state "Waiting for Instructions", the SSP performs the following actions:

– The SSF cancels T_{SSF}.

– If "cutAndPaste" is present, then the SSF deletes ("cut") from the dialled IN number the indicated number of digits and pastes the remaining dialled digits at the end of the "destinationRoutingAddress" parameter delivered by the SCF. The resulting directory number is used for routing to complete the related call.

– If "cutAndPaste" is not present, then the "destinationRoutingAddress" parameter delivered by the SCF is used for routing to complete the related call. Note that in the case of hand-off, this results in routing to an assisting SSP or IP.

– If the "callingPartyNumber" is supplied, this value may be used for all subsequent SSF processing.

– If no EDPs have been armed and neither a CallInformationReport nor an ApplyChargingReport has been requested, the FSM goes to state "Idle" (e9). Otherwise, the FSM goes to state "Monitoring" (e11).

No implicit activation or deactivation of DPs occurs.

Statistic counter(s) are not affected.

Connect completes when the INAP processing of the operation is complete and before the SSP starts the processing necessary to select a circuit.

Therefore, in order to detect route select failure after a "Connect" it is necessary to explicitly arm the "Route Select Failure" EDP before sending the "Connect" (although they may be in the same message).

3.3.16.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.17 ConnectToResource procedure

3.3.17.1 General description

This operation is used to connect a call from the SSF to a specialized resource. After successful connection to the SRF, the interaction with the caller can take place. The SSF relays all operations for the SRF and all responses from the SRF.

3.3.17.1.1 Parameters

- resourceAddress:

  This parameter identifies the physical location of the SRF.

  – iPRoutingAddress:

    This parameter indicates the routing address to set up a connection towards the SRF.

  – none:

    This parameter indicates that the call party is to be connected to a predefined SRF.

- serviceInteractionIndicators:

  This parameter contain indicators sent from the SCP to the SSP for control of the network based services at the originating exchange and the destination exchange.

3.3.17.2 Invoking Entity (SCF)

3.3.17.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) The SLPI has determined that additional information from the call party is needed.
3) The call party is not connected to any other party.
4) The SCSM-FSM is in the state "Routing to Resource", substate "Determine Mode".
5) The SLPI has determined that the SRF can be accessed from the SSF.

SCF Postconditions:

1) The SCSM sends out a 'PlayAnnouncement' or 'PromptAndCollectUserInformation' operation accompanying the 'ConnectToResource'
2) The SCSM-FSM moves to the state "User Interaction".

3.3.17.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.17.3 Responding Entity (SSF)

3.3.17.3.1 Normal procedure

SSF Preconditions:

1) Basic call processing has been suspended at a DP and a control relationship has been established.

2) The SSF-FSM is in the state "Waiting for Instructions".

SSF Postconditions:

1) The call is switched to the SRF.

2) A control relationship to the SRF is established.

3) The SSF-FSM moves to the state "Waiting for End of User Interaction". If necessary, T_{SSF} is set.

NOTES

1 Whether the T_{SSF} is used or not in this case is network operator dependent. But it must be synchronized with $T_{SCF-SSF}$ in the SCSM.

2 The successful connection to the SRF causes a state transition in the SRF-FSM from "Idle" to "Connected".

3.3.17.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.18 Continue procedure

3.3.18.1 General description

This operation is used to request the SSF to proceed with call processing at the DP at which it previously suspended call processing to await SCF instructions. The SSF continues call processing without substituting new data from the SCF.

3.3.18.1.1 Parameters

None.

3.3.18.2 Invoking Entity (SCF)

3.3.18.2.1 Normal procedure

SCF Precondition:

– SCSM is in the state "Preparing SSF instructions".

SCF Postcondition:

– SCSM is in the state "Waiting for Notification or Request", in case monitoring was required, or in the state "Idle", in case no monitoring was required.

The SCSM is in state "Preparing SSF instructions". The 'Continue' operation is invoked by a SLPI. This causes a SCSM transition to state "Idle" if no subsequent monitoring is required. However, if monitoring is required, like in the case of armed EDPs or outstanding report requests, the SCSM transitions to state "Waiting for Notification of Report".

3.3.18.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.18.3 Responding Entity (SSF)

3.3.18.3.1 Normal procedure

SSF Preconditions:

1) BCSM: Basic call processing has been suspended at any DP.

2) SSF-FSM is in the state "Waiting for Instructions".

SSF Postconditions:

1) BCSM: Basic call processing continues.

2) SSF-FSM is in the state "Monitoring", because at least one EDP was armed, or a 'CallInformationReport' or 'ApplyChargingReport' was requested; or

   SSF-FSM is in the state "Idle", because no EDPs were armed and neither the 'CallInformationReport' nor the 'ApplyChargingReport' was requested.

The SSF-FSM is in state "Waiting for instructions". The SSME-Control receives the 'Continue' operation and relays it to the appropriate SSF-FSM. The SSF-FSM transitions to state "Idle" in case no EDPs are armed and no outstanding report requests are present. The SSF-FSM transits to state "Monitoring" if at least one EDP is armed, or if there is at least one outstanding report request. Basic call processing is resumed.

3.3.18.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.19 DisconnectForwardConnection procedure

3.3.19.1 General description

This operation is used in the following two cases:

1) To clear a connection to a SRF

   This operation is used to explicitly disconnect a connection to a resource (SRF) established previously with a 'ConnectToResource' or an 'EstablishTemporaryConnection' operation. It is used for a forward disconnection from the SSF. An alternative solution is the backward disconnect from the SRF, controlled by the 'DisconnectFromIPForbidden' parameter in the 'PlayAnnouncement' and 'PromptAndCollectUserInformation' operations.

2) To clear a connection to an assisting SSF

   This operation is sent to the non-assisting SSF of a pair of SSFs involved in an assist procedure. It is used to disconnect the temporary connection between the initiating SSF and the assisting SSF, and the assisting SSF and its associated SRF.

3.3.19.1.1 Parameters

None.

3.3.19.2 Invoking Entity (SCF)

3.3.19.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.

2) An assist- or a relay procedure is in progress.

3) An SLPI has determined that a 'DisconnectForwardConnection' operation has to be sent by the SCF.

SCF Postcondition:

– SLPI execution may continue.

The 'DisconnectForwardConnection' operation is used to instruct the SSF to disconnect the concerned forward connection to the assisting SSF or the physical entity containing the SRF.

In the SCSM-FSM state "User Interaction", substate "Waiting for Response from the SRF", this operation is invoked by the SCF when the service logic determines that user interaction is finished and requests the SSF to disconnect the temporary connection to the assisting SSF or the SRF. The SCSM-FSM then transitions to state "Preparing SSF Instructions".

3.3.19.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.19.3 Responding Entity (SSF)

3.3.19.3.1 Normal procedure

SSF Preconditions:

1) Call origination attempt has been initiated.

2) Basic call processing has been suspended at a DP.

3) The initiating SSF is in the state "Waiting for End of User Interaction" or "Waiting for End of Temporary Connection".

SSF Postconditions:

1) The connection to the SRF or assisting SSF is released.

2) The SSF is waiting for instructions.

The receipt of "DisconnectForwardConnection" results in disconnecting the assisting SSF or the physical entity containing the SRF from the concerned call. It does not release the connection from the SSF back to the end user.

This operation is accepted in the SSF-FSM states "Waiting for End of Temporary Connection" or "Waiting for End of User Interaction". On receipt of this operation in these states, the SSF must perform the following actions:

– The initiating SSF releases the connection to the assisting SSF or the relay SRF.

– The SSF resets T_{SSF}.

– The SSF-FSM goes to state "Waiting for Instructions" (e8).

NOTE – The successful disconnection to the SRF causes a state transition in the SRF-FSM to "Idle". A current order ("PlayAnnouncement" or "PromptAndCollectUserInformation") is cancelled and any queued order ("PlayAnnouncement" or "PromptAndCollectUserInformation") is discarded.

3.3.19.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.20 EstablishTemporaryConnection procedure

3.3.20.1 General description

This operation is used to create a connection between an initiating SSF and an assisting SSF as part of a service assist procedure. It can also be used to create a connection between a SSF and a SRF, for the case where the SRF exists in a separately addressable physical entity.

3.3.20.1.1 Parameters

– assistingSSPIPRoutingAddress:

This parameter indicates the destination address of the SRF for assist procedure. The 'assistingSSPIPRoutingAddress' may contain embedded within it, a 'correlationID' and 'scfID', but only if 'correlationID' and 'scfID' are not specified separately.

– correlationID:

This parameter is used by the SCF to associate the 'AssistRequestInstructions' from the assisting SSF (or the SRF) with the Request from the initiating SSF. The 'correlationID' is used only if the correlation id is not embedded in the 'assistingSSPIPRoutingAddress'. The network operator has to decide about the actual mapping of this parameter on the used signalling system.

– scfID:

See Recommendation Q.1290. The 'scfID' is used only if the SCF id is not embedded in the 'assistingSSPIPRoutingAddress'. The network operator has to decide about the actual mapping of this parameter on the used signalling system.

– carrier:

See Recommendation Q.1290. In this message, the carrier selection field is null (00000000) and Carrier ID indicates the carrier to use for the call.

– serviceInteractionIndicators:

This parameter contain indicators sent from the SCP to the SSP for control of the network based services at the originating exchange and the destination exchange.

3.3.20.2 Invoking Entity (SCF)

3.3.20.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) The service logic has determined that a connection is needed between the SSF and SRF or between the SSF and an assisting SSF.
3) The call party is not connected to any other party.

SCF Postcondition:

– The SCF is "Waiting for Assist Request Instructions".

In the SCSM-FSM state "Routing to Resource", this operation is invoked by the SCF when the service logic determines that an assisting SSF or a Direct SCF-SRF relation is needed. The SCSM-FSM then transitions to state "Waiting for Assist Request Instructions".

3.3.20.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.20.3 Responding Entity (SSF)

3.3.20.3.1 Normal procedure

SSF Preconditions:

1) Call origination attempt has been initiated.
2) Basic call processing has been suspended at DP 1,2,3,4,5,6,7,12,13,14, or 15.
3) The SSF waits for instructions.
4) The SSF is not an assisting SSF.

SSF Postconditions:

1) The SSF performs the call processing actions to route the call to the assisting SSF or SRF according to the 'assistingSSPIPRoutingAddress' requested by the SCF.
2) The SSF waits for end of temporary connection.

On receipt of this operation in the SSF-FSM state "Waiting for Instructions", the SSP has to perform the following actions:

– Reset the T_{SSF} (optional).

NOTE – This "optional" means that the application timer T_{SSF} is optionally set. Whether it is used or not is network operator dependent. But it must be synchronized with $T_{SCF-SSF}$ in the SCSM.

– Route the call to assisting SSF or SRF using 'assistingSSPIPRoutingAddress'.

– The SSF-FSM goes to state "Waiting for End of Temporary Connection" (e7).

3.3.20.3.2 Error handling

Until the connection set-up has been accepted (refer to Recommendation Q.71) by the assisting SSF-SRF, all received failure indications from the network on the ETC establishment shall be reported to the SCF as ETC error ETCFailed (e.g. busy, congestion). Note that the operation timer for ETC shall be longer than the maximum allowed time for the signalling procedures to accept the connection.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.21 EventNotificationCharging procedure

3.3.21.1 General description

This operation is used by the SSF to report to the SCF the occurrence of a specific charging event type as requested by the SCF using the "RequestNotificationChargingEvent" operation. The operation supports the options to cope with the interactions concerning the charging (refer to Appendix II.4/Q.1214 "Charging scenarios").

As several charging events may occur during a connection configuration a possibility exists for the EventNotificationCharging operation to be invoked on multiple occasions. For each connection configuration EventNotificationCharging may be used several times.

3.3.21.1.1 Parameters

– eventTypeCharging:

  This parameter indicates the charging event type which has occurred. Its content is network operator specific, which may be "charge pulses" or "charge messages".

– eventSpecificInformationCharging:

  This parameter contains charging related information specific to the event. Its content is network operator specific.

– legID:

  This parameter indicates the leg id on which the charging event type applies.

– monitorMode:

  This parameter indicates how the charging event is reported. When the "monitorMode" is "interrupted", the event is reported as a request; if the "monitorMode" is "notifyAndContinue" the event is reported as a notification. The "monitorMode" "transparent" is not applicable for the "EventNotificationCharging" operation.

3.3.21.2 Invoking Entity (SSF)

3.3.21.2.1 Normal procedure

SSF Preconditions:

1) A control relationship exists between the SCF and the SSF.

2) A charging event has been detected that is requested by the SCF.

SSF Postcondition:

– No FSM state transition.

The SSF-FSM is in any state except "Idle". This operation is invoked if a charging event has been detected that is requested by the SCF. The detected charging event can be caused by:

a) another SLPI; or

b) another exchange.

Irrespective of the charging event cause, the SSF performs one of the following actions on occurrence of the charging event (according the corresponding monitorMode):

Interrupted:

– Notify the SCF of the charging event using "EventNotificationCharging" operation: do not process the event, but discard it. However, call and existing charging processing will not be suspended in the SSF.

NotifyAndContinue:

– Notify the SCF of the charging event using "EventNotificationCharging", and continue processing the event or signal.

3.3.21.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.21.3 Responding Entity (SCF)

3.3.21.3.1 Normal procedure

SCF Precondition:

– A "RequestNotificationChargingEvent" has been sent at the request of a SLPI and the SLPI is expecting an "EventNotificationCharging" from the SSF.

SCF Postcondition:

– No FSM state transition.

On receipt of this operation the SLPI which is expecting this notification can continue. If the corresponding monitor mode was set by the SLPI to Interrupted, the SLPI prepares instructions for the SSF if necessary.

3.3.21.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.22 EventReportBCSM procedure

3.3.22.1 General description

This operation is used to notify the SCF of a call related event previously requested by the SCF in an 'RequestReportBCSMEvent' operation. The monitoring of more than one event could be requested with a 'RequestReportBCSMEvent' operation, but each of these requested events is reported in a separate 'EventReportBCSM' operation.

3.3.22.1.1 Parameters

- eventTypeBCSM:

  This parameter specifies the type of event that is reported.

- bcsmEventCorrelationID:

  Used by the SCF for correlation with a previous operation.

- eventSpecificInformationBCSM:

  This parameter indicates the call related information specific to the event.
  For 'CollectedInfo' it will contain the 'calledPartyNumber'.
  For 'AnalysedInfo' it will contain the 'calledPartyNumber'.
  For 'RouteSelectFailure' it will contain the 'FailureCause', if available.
  For O- or T-Busy it will contain the 'BusyCause', if available.
  For O- or T-NoAnswer it will be empty.
  For O- or T-Answer it will be empty.
  For O- or T-MidCall it will contain 'connectTime', if available.
  For O- or T-Disconnect it will contain the 'releaseCause' and/or 'connectTime', if available.
  The connect time, if available, indicates the duration between the received answer indication from the called party side and the release of the connection in units of 100 ms.

- legID:

  This parameter indicates the party in the call for which the event is reported. SSF will use the option ’ReceivingSideID’ only.

  – receivingSideID:

    The following values for 'legID' are assumed:

    i) 'legID' = 1 indicates the party that was present at the moment of the 'InitialDP' or a DP-specific operation (in case of a midcall trigger, the party causing the trigger), or the party that was created with an 'InitiateCallAttempt' operation.

    ii) 'legID' = 2 indicates the party that was created with a 'Connect' operation, or in case of a mid call trigger, the party not causing the trigger.

    If not included, the following defaults are assumed:

    iii) legID = 1 for the events CollectedInfo, AnalysedInformation, O-Abandon and T-Abandon.

    iv) legID = 2 for the events RouteSelectFailure, O-CalledPartyBusy, O-NoAnswer, O-Answer, T-Busy, T-NoAnswer and T-Answer.

    The 'legID' parameter shall always be included for the events O-MidCall, O-Disconnect, T-MidCall and T-Disconnect.

- miscCallInfo:

  This parameter indicates detection point related information.

  – messageType:

    This parameter indicates whether the message is a request, i.e. resulting from a 'RequestReportBCSMEvent' with 'monitorMode' = interrupted, or a notification, i.e. resulting from a 'RequestReportBCSMEvent' with 'monitorMode' = 'notifyAndContinue'.

  – dPAssignment:

    This parameter shall be omitted for this operation.

3.3.22.2 Invoking Entity (SSF)

3.3.22.2.1 Normal procedure

SSF Preconditions:

1) The SSF-FSM is in the state "Monitoring"; or

   the SSF-FSM may be in state "Waiting for Instructions" if the O/TDisconnect DP is armed and encountered; or

   the SSF-FSM may be in any state if the O/TAbandon DP is armed and encountered.

2) The BCSM proceeds to an EDP that is armed.

SSF Postconditions:

1) The SSF-FSM stays in the state "Monitoring" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested.

2) The SSF-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested.

3) The SSF-FSM moves to the state "Waiting for Instructions" if the message type was request. Call processing is interrupted.

For certain service features it is necessary that the same O-BCSM instance is reused. Examples are follow-on calls.

The decision to reuse the same O-BCSM instance can only be taken by the SCF after certain armed EDP-R’s are reported.

The considered EDP-R's are:

- RouteSelectFailure;
- O_CalledPartyBusy;
- O_NoAnswer;
- O_Disconnect.

If a EDP-R is met that causes the release of the related leg, all EDPs related to that leg are disarmed and the event is reported via "EventReportBCSM". To allow the reuse of the same O-BCSM instance, the BCSM has to store all call related signalling parameters (e.g. "callingPartyNumber", "callingPartyCategory") until the BCSM instance is released.

In the case the respective EDP-R is met (see list above with "O-Disconnect" only for legID = 2), the A-leg will be held, the B-leg will be released and the SCF is informed via "EventReportBCSM".

If the A-party disconnects, the call is released whether or not a DP was armed.

3.3.22.2.2 Error handling

In case the message type is request, on expiration of T_{SSF} before receiving any operation, the SSF aborts the interaction with the SCF and the call is given final treatment, e.g. a final announcement.

Operation related error handling is not applicable, due to class 4 operation.

3.3.22.3 Responding Entity (SCF)

3.3.22.3.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SSF and the SCF.
2) The SCSM-FSM is in the state "Preparing SSF Instructions", substate "Waiting for Notification or Request".

SCF Postconditions:

1) The SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

   the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

   the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

2) The event is reported to a SLPI, based on the dialogue ID. The SCF will prepare SSF or SRF instructions in accordance with the SLPI.

3.3.22.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.23 FurnishChargingInformation procedure

3.3.23.1 General description

This operation is used to request the SSF to generate, register a call record or to include some information in the default call record. The registered call record is intended for off-line charging of the call. A possibility exists for the FurnishChargingInformation (FCI) operation to be invoked on multiple occasions. FCI could be applied at the beginning of the call in order to request to start call record generation. In addition FCI can also be applied at the end of the call or connection configuration (e.g. for follow-on calls). In this case FCI is used to include charge related information into the

call record which was started at the beginning of the call. When additional FCIs are used it is recommended to arm an EDP-R (indicating the end of call or connection configuration) to be able to apply FCI before the termination of the call record generation. The charging scenarios supported by this operation are: 2.2, 2.3 and 2.4 (refer to Appendix II/Q.1214, "Charging scenarios").

3.3.23.1.1 Parameters

– FCIBillingChargingCharacteristics:

  This parameter indicates billing and/or charging characteristics. Its content is network operator specific. Depending on the applied charging scenario, the following information elements can be included (refer to Appendix II/Q.1214, "Charging scenarios"):

  - complete charging record (scenario 2.2);
  - charge party (scenario 2.3);
  - charge level (scenario 2.3);
  - charge items (scenario 2.3);
  - correlationID (scenario 2.4).

3.3.23.2 Invoking Entity (SCF)

3.3.23.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) An SLPI has determined that a "FurnishChargingInformation" has to be sent by the SCF.

SCF Postconditions:

1) No FSM state transition.
2) SLPI execution may continue.

The SCSM-FSM is in state "Preparing SSF instruction" or is in state "Queueing FSM". This operation is invoked by the SCF if a SLPI results in the request of creating a call record to the SSF or to include some billing or charging information into the default call record. In the case of call queueing, this operation may contain information pertaining to the initiation of queueing or the call queueing time duration for call logging purpose. This causes no SCSM-FSM state transition.

3.3.23.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.23.3 Responding Entity (SSF)

3.3.23.3.1 Normal procedure

SSF Preconditions:

– SSF-FSM: State c, "Waiting for Instructions"; or
  SSF-FSM: State d, "Waiting for End of User Interaction"; or
  SSF-FSM: State e, "Waiting for End of Temporary Connection"; or
  SSF-FSM: State f, "Monitoring"; or
  Assisting/hand-off SSF-FSM State b, "Waiting for Instructions".

SSF Postcondition:

– No FSM state transition.

On receipt of this operation, the SSF performs actions to create the call record according the off line charging scenario which is applicable using the information elements included in the operation:

- registers the complete call record included in the operation;
- generates and registers a call record according the information (charge party, charge level, charge items);
- include the information received "correlationID" in the default call record which is generated and, registered by default at the SSF.

By means of a parameter at the "FurnishChargingInformation" operation the SCF can initiate the pulse metering function of the SSF.

In that case the SSF shall generate meter pulses according to the applicable charging level, account and record them.

The SSF records charge related data like for example the call duration, begin time stamp or end time stamp. Additionally the SSF records further data if required.

The charging level can be determined by:

a) the SCF; or

b) the SSF; or

c) a succeeding exchange; or

d) the post processing function.

If a) applies, the charging level is included in the "FurnishChargingInformation" operation.

If b) applies, the SSF shall determine the charging level based on the corresponding parameters contained in the operation.

If c) applies, either the "FurnishChargingInformation" operation contains the corresponding parameters indicating that the charging level shall be determined in a succeeding exchange, or the SSF detects during the determination of the charging level based on the provided parameters that the charging level shall be determined in a succeeding exchange.

The SSF can either account received pulses or convert any charging messages received from the B-side to pulses. In both cases, the accumulated pulses are included when the IN call record is generated or ignored.

3.3.23.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.24 HoldCallInNetwork procedure

3.3.24.1 General description

This operation is used to provide the capability of queueing a call during the setup phase (e.g. to provide a call completion to busy, a call would be queued until the destination becomes free).

3.3.24.1.1 Parameters

- holdcause:

  This parameter is optional. When used, it is network-specific and denotes the cause for holding the call.

3.3.24.2 Invoking Entity (SCF)

3.3.24.2.1 Normal procedure

SCF Precondition:

- The SCSM is in state 2.2.1, "Preparing SSF Instructions", and the event (e2.2.3) Busy_Line/Trunk has taken place.

SCF Postcondition:

– The SCSM enters state 2.2.2 Queueing.

3.3.24.2.2 Error handling

There is no specific error treatment associated with this operation. Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.24.3 Responding Entity (SSF)

3.3.24.3.1 Normal procedure

SSF Precondition:

– The SSF-FSM is in state c, "Waiting for Instructions".

SSF Postcondition:

– The SSF-FSM remains in the same state.

3.3.24.3.2 Error handling

There is no operation-specific error treatment associated with this operation. Generic error handling for the operation-related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.25 InitialDP procedure

3.3.25.1 General description

This operation is sent by the SSF after detection of a TDP-R in the BCSM, to request the SCF for instructions to complete the call.

3.3.25.1.1 Parameters

- serviceKey:

  This parameter identifies for the SCF unambiguously the requested IN service. It is used to address the correct application/SLP within the SCF (not for SCP addressing).

- calledPartyNumber:

  This parameter contains the number used to identify the called party in the forward direction, i.e. see Recommendation Q.762.

- callingPartyNumber:

  See Q.762 Calling Party Number signalling information.

- callingPartysCategory:

  See Q.762 Calling Party Category signalling information.

- cGEncountered:

  See Recommendation Q.1290.

- iPSSPCapabilities:

  See Recommendation Q.1290.

- iPAvailable:

  See Recommendation Q.1290.

- locationNumber:

  This parameter is used to convey the geographical area address for mobility services, see Recommendation Q.762. It is used when 'callingPartyNumber' does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

- originalCalledPartyID:

  See Q.762 Original Called Number signalling information.

- triggerType:

  This parameter indicates to the SCF the particular event which caused the detection of a valid trigger condition.

- highlayerCompatibility:

  This parameter indicates the type of the high layer compatibility, which will be used to determine the ISDN-teleservice of a connected ISDN terminal. For encoding DSS-1 (see Recommendation Q.931) is used. The highlayerCompatibility can also be transported by ISUP within the ATP (see Recommendation Q.763) parameter.

- serviceInteractionIndicators:

  This parameter contains indicators sent from the SSF to the SCF for control of the network based services at the originating exchange and the destination exchange.

- additionalCallingPartyNumber:

  The calling party number provided by the access signalling system of the calling user, e.g. provided by a PBX.

- forwardCallIndicators:

  This parameter indicates if the call shall be treated as a national or international call. It also indicates the signalling capabilities of the network access, preceding network connection and the preferred signalling capabilities of the succeeding network connection. The network access capabilities does not indicate the terminal type. For example, an ISPBX will have an ISDN type of access, but the end user terminal behind the ISPBX may be ISDN or non-ISDN.

- bearerCapability:

  This parameter indicates the type of the bearer capability connection or the transmission medium requirements to the user. It is a network option to select one of the two parameters to be used:

  – bearerCap:

    This parameter contains the value of the DSS-1 Bearer Capability parameter (see Recommendation Q.931) in case the SSF is at local exchange level or the value of the ISUP User Service Information parameter (see Recommendation Q.763) in case the SSF is at transit exchange level.

    The parameter 'bearerCapability' shall only be included in the 'InitialDP' operation in case the DSS-1 Bearer Capability parameter or the ISUP User Service Information parameter is available at the SSP.

    If two values for bearer capability are available at the SSF or if User Service Information and User Service Information Prime are available at the SSF, the 'bearerCap' shall contain the value of the preferred bearer capability respectively the value of the User Service Information Prime parameter.

  – tmr:

    The tmr is encoded as the Transmission Medium Requirement parameter of the ISUP according to Recommendation Q.763.

    If two values for transmission medium requirement are available at the SSF or if Transmission Medium Requirement and Transmission Medium Requirement Prime are available at the SSF, the 'bearerCap' shall contain the value of the preferred transmission medium requirement respectively the value of the Transmission Medium Requirement Prime parameter.

- eventTypeBCSM:

  This parameter indicates the armed BCSM detection point event, resulting in the 'InitialDP' operation.

- redirectingPartyID:

  This parameter indicates the last directory number the call was redirected from.

- redirectionInformation:

  See Q.763 Redirection Information signalling information.

3.3.25.2 Invoking Entity (SSF)

3.3.25.2.1 Normal procedure

SSF Preconditions:

1) An event fulfilling the criteria for the DP being executed has been detected.
2) Call gapping and SS No. 7 overload are not in effect for the call, and the call is not to be filtered.

SSF Postconditions:

1) A control relationship has been established if the DP was armed as a TDP-R. The SSF-FSM moves to the state "Waiting for Instructions".
2) The SSF-FSM remains in the state "Idle" if the DP was armed as a TDP-N.

Following a trigger detection (due to the DP criteria assigned being met) related to an armed TDP in the BCSM caused by a call origination attempt, the SSF checks if call gapping, SS No. 7 overload or service filtering are not in effect for the related call segment.

If these conditions are met, then the 'InitialDP' operation is invoked by the SSF. The address of the SCF the 'InitialDP' operation has to be sent to, is determined on the base of trigger related data. The SSF provide as many parameters as available. In some cases, some parameters must be available (such as 'callingPartyNumber' or 'callingPartyCategory'). This is to be handled appropriately by the SSF in its trigger table (to know that such parameters are necessary for some triggering conditions) and in conducting the necessary action to get these parameters if they are not available (for instance if non-SS No. 7 signalling is used, it may be possible to request the 'callingPartyCategory' from a preceding exchange).

Otherwise, the call control is given back to the underlying network.

If the DP was armed as a TDP-R, a control relationship is established to the SCF. The SSF application timer T_{SSF} is set when the SSF sends 'InitialDP' for requesting instructions from the SCF. It is used to prevent excessive call suspension time.

3.3.25.2.2 Error handling

If the destination SCF is not accessible, then the call is given final treatment.

On expiration of T_{SSF} before receiving any operation, the SSF aborts the interaction with the SCF and the call is given final treatment, e.g. routing to a final announcement.

If the calling party abandons after the sending of 'InitialDP', then the SSF aborts the control relationship by means of an abort to TCAP. Note that TCAP will wait until the first response message from the SCF has been received before it sends an abort to the SCF (see also 3.2).

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.25.3 Responding Entity (SCF)

3.3.25.3.1 Normal procedure

SCF Precondition:

– None.

SCF Postcondition:

– An SLPI has been invoked.

On receipt of 'InitialDP' operation the SCSM moves from "Idle" to the state "Preparing SSF Instructions", a control relationship to the related SSF is created. A Service Logic Program Instance (SLPI) is invoked for processing the 'InitialDP' operation based on the 'serviceKey' parameter. By means of this control relationship, the SCF may influence the Basic Call Processing in accordance with the service logic invoked.

The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the SLPI, i.e. the requested IN service itself.

3.3.25.3.2 Error handling

If the 'InitialDP' operation is rejected, then the SCSM remains in "Idle". The maintenance function is informed and no SLPI is invoked.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.26 InitiateCallAttempt procedure

3.3.26.1 General description

This operation is used to request the SSF to create a new call to one call party using the address information provided by the SCF (e.g. wake-up call). An EDP-R must be armed on answer and all the call failure events, in order to have the SCF, treat this call appropriately when either of these events is encountered.

3.3.26.1.1 Parameters

– destinationRoutingAddress:

  This parameter contains the called party number towards which the call is to be routed. The destinationRoutingAddress may contain one called party number only for this operation. The encoding of the parameter is defined in Recommendation Q.763.

– alertingPattern:

  See Recommendation Q.1290. It only applies if the network signalling supports this parameter or if SSF is the terminating local exchange for the subscriber.

– iSDNAccessRelatedInformation:

  Carries the same information as the protocol element ISUP Access Transport parameter in Recommendation Q.762.

– travellingClassMark:

  The SCF uses the travellingClassMark parameter to provide essential route selection information to the SSF. The SSF uses this information to populate the outgoing ISUP-IAM message, the population and mapping of this parameter is network operator specific.

– serviceInteractionIndicators:

  This parameter contains indicators sent from the SCF to the SSF for control of the network based services at the originating exchange and the destination exchange.

– callingPartyNumber:

  This parameter identifies which number shall be regarded as the calling party for the created call. If this parameter is not sent by the SCF, the SSF may supply a network dependent default value.

3.3.26.2 Invoking Entity (SCF)

3.3.26.2.1 Normal procedure

SCF Preconditions:

1) An SLPI has been invoked, no control relationship exists between SCF and SSF.

2) An SLPI has determined that an 'InitiateCallAttempt' operation should be sent to the SCF.

3) The SCSM-FSM is in state "Idle".

SCF Postconditions:

1) A control relationship is established between the SCF and SSF.

2) The SCSM-FSM is in state "Preparing SSF Instructions".

3) SLPI execution continues.

The SCSM-FSM moves to state "Preparing SSF Instructions" when the service logic invokes this operation. In order to enable the establishment of a control relationship between the SCF and SSF and to allow the SCF to control the created call appropriately, the SLPI shall monitor for the BCSM event(s) which report the result of the created call set-up. This includes DP 3 or DP 4, 5, 6, and 7. Any other Non-Call_Processing_Instructions may be sent as well. The 'InitiateCallAttempt' operation creates a BCSM instance in the SSF but the SSF suspends the call processing of this BCSM. The SLPI shall send a 'Continue' operation to request the SSF to route the call to the specified destination. The SCSM-FSM shall proceed as specified at the procedure for the 'Continue' operation.

The above described procedure shall be part of the establishment of the control relationship, i.e. operations up to and including the 'Continue' operation shall be sent together in the same message to the SSF. The SCF shall start a response Timer T_{SCF} when the 'InitiateCallAttempt' operation is sent. The response Timer shall supervise the confirmation of the dialogue from SSF, the value of T_{SCF} shall be equal or less than the network no answer timer.

3.3.26.2.2 Error handling

On expiration of T_{SCF} the SCF shall abort the dialogue, report the abort to the maintenance functions and inform the SLPI on the failure of dialogue establishment. The SCSM-FSM moves to the "Idle" state.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.26.3 Responding Entity (SSF)

3.3.26.3.1 Normal procedure

SSF Precondition:

– None.

SSF Postconditions:

1) A new originating BCSM has been created, call processing is suspended at DP 1.

2) The SSF-FSM has moved from "Idle" state to state "Waiting for Instructions".

Upon reception of 'InitiateCallAttempt', the SSF creates a new originating BCSM and suspends the call processing of this BCSM at DP 1. All subsequent operations are treated according to their normal procedures.

The properties and capabilities, normally received from or associated to the calling party, required for the call set-up shall have a network dependent default value. If a calling party number is supplied by the SCF, these properties may be dependent on the received calling party number

3.3.26.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.27 ModifyEntry procedure

3.3.27.1 General description

The X.500 'ModifyEntry' operation is used to request the SDF to make one or several modifications to a data object. The modifications concern the attributes and their values of which the object is composed. The type of modifications to perform is given in the operation argument provided by the SCF. The modifications do not concern the values of the attributes used to identify the object (i.e. the object name). For a full description of the ModifyEntry operation, see 11.3/X.511.

3.3.27.1.1 Parameters

See 11.3.2/X.511 and 11.3.3/X.511.

3.3.27.2 Invoking Entity (SCF)

3.3.27.2.1 Normal procedure

SCF Precondition:

- SCSM: "SDF Bound" or "Wait for Subsequent Requests".

SCF Postcondition:

- SCSM: "SDF Bound".

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the service logic to modify information of the SDF exists, an internal event [(e2) Request_to_SDF] occurs. Until the application process indicates, with a delimitor (or a timer expiry), that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response [(E5) Response_from_SDF_with_Bind or (E4) Bind_Error] to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the ModifyEntry operation was discarded. In the state "SDF Bound", the response of the ModifyEntry operation [(E7) Response_from_SDF] causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the ModifyEntry operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the service logic to modify information of the SDF exists, an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response [(E7) Response_from_SDF] to the ModifyEntry operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the ModifyEntry operation or an error.

3.3.27.2.2 Error handling

Generic error handling for the operation related errors is described in 11.3.4/X.511 and 11.3.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.27.3 Responding Entity (SDF)

3.3.27.3.1 Normal procedure

SDF Precondition:

- SDSM: "SCF Bound" or "Bind Pending".

SDF Postcondition:

- SDSM "SCF Bound".

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a 'ModifyEntry' operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a 'ModifyEntry' operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before updating the different attributes specified in the operation parameters, the SDF shall take the following actions:

- verify that the object accessed by the request exists;
- verify that the user on behalf of whom the request is performed has sufficient access rights to modify the object for each elementary modifications contained in the operation;
- verify that each attribute or value on which an operation should be performed exists in the object;
- verify that the proposed modifications are compatible with the object class of the object or with the syntax of the attribute.

After the specified actions, indicated above, are successfully executed for all the modifications requested in the operation, all the modifications for a same attribute take place in the order given in the "changes" parameter. A result is returned to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

3.3.27.3.2 Error handling

Generic error handling for the operation related errors is described in 11.3.4/X.511 and 11.3.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.28 OAnswer procedure

3.3.28.1 General description

This operation is sent from the SSF to the SCF at the O_Answer DP, after detecting a valid trigger condition, or to report an event requested by RequestReportBCSMEvent.

3.3.28.1.1 Parameters

- dPSpecificCommonParameters:
- callingPartyBusinessGroupID:

  See Recommendation Q.1290. The SCF can use this parameter to select SLPs based on the group and for authorization purposes. The network operators can specify that this parameter should be used if their particular network has the information available.
- callingPartySubaddress:

  See Q.931 Calling Party Subaddress.
- callingFacilityGroup:

  See Recommendation Q.1290.
- callingFacilityGroupMember:

  See Recommendation Q.1290.
- originalCalledPartyID:

  See Q.762 Original Called Number signalling information.
- redirectingPartyID:

  This parameter (if available) is the directory number of the last redirecting party.
- redirectionInformation:

  See Q.763 Redirection Information signalling information.
- routeList:

  routeList represents the list of routes which would have been used in order to route the call. The network operators can specify that this parameter should be used if their particular network has the information available.
- travellingClassMark:

  As defined in Recommendation Q.1290.

3.3.28.2 Invoking Entity (SSF)

3.3.28.2.1 Normal procedure

SSF Preconditions (TDP and EDP):

1) Call origination attempt has been initiated.
2) Indication received that the call has been accepted and the terminating party has answered.
3) Call gapping or service filtering are not in effect.
4) DP criteria are met.

5) For a TDP-R, there is no existing control relationship.

6) For an EDP-R, there is an existing control relationship and the EDP O_Answer is armed.

7) For an EDP-N, there is an existing control or monitoring relationship and the EDP O_Answer is armed.

SSF Postconditions for TDP:

1) For a TDP-R, basic call processing has been suspended at O_Answer DP, and a control relationship has been established.

2) For a TDP-N, basic call processing proceeds at O_Active PIC, and no control relationship has been established.

SSF Postconditions for EDP:

1) The SSF-FSM stays in the state "Monitoring" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested.

2) The SSF-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested.

3) The SSF-FSM moves to the state "Waiting for Instructions" if the message type was request. Call processing is interrupted.

3.3.28.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.28.3 Responding Entity (SCF)

3.3.28.3.1 Normal procedure

SCF Precondition (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.28.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.29 ODisconnect procedure

3.3.29.1 General description

This operation is sent by the SSF to the SCF after detecting a valid trigger condition at the O_Disconnect DP, or to report an event requested by RequestReportBCSMEvent. Refer to 4.2.2.2/Q.1214 for additional call modelling related semantics.

3.3.29.1.1 Parameters

– serviceAddressInformation (serviceKey, miscCallInfo, triggerType):

See Recommendation Q.1290. The triggerType indicates the particular event which caused call suspension. The miscCallInfo indicates that a request for instructions or a notification has been issued to the SCF and the category for the trigger (line, group, office). The serviceKey is used in identifying the service logic to be invoked.

– bearerCapability:

This parameter indicates the type of bearer capability connection to the user. Refer to 6.4.4/Q.1214 for population rules for the bearer capability parameter.

– calledPartyNumber:

See Q.762 Called Party Number signalling information. This parameter is used to identify the called party in the forward direction.

– callingPartyNumber:

See Q.762 Calling Party Number signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartynumber parameter.

– callingPartysCategory:

See Q.762 Calling Party's Category signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartysCategory parameter.

– iPSSPCapabilities:

See Recommendation Q.1290.

– iPAvailable:

See Recommendation Q.1290.

– iSDNAccessRelatedInformation:

This parameter contains (possibly multiple) information elements as per Recommendation Q.931. See Access Transport Parameter signalling information in Recommendations Q.762, Q.763 and Q.931. Refer to 6.4.4/Q.1214 (Analysed Information) for population rules for ISDN access related information.

– cGEncountered:

See Recommendation Q.1290.

– locationNumber:

See Q.762 Location Number signalling information. This parameter is used to convey the geographical area address for mobility services. It is used when the "callingPartyNumber" does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

– serviceProfileIdentifier:

See Annex A/Q.932. Refer to 6.4.4/Q.1214 for population rules for serviceProfileIdentifier.

– terminalType:

See Recommendation Q.1290. This parameter identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

– chargeNumber:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules for chargeNumber.

– servingAreaID:

  See Recommendation Q.1290.

– callingPartyBusinessGroupID:

  See Recommendation Q.1290.

– callingPartySubaddress:

  See Q.931 Calling Party Subaddress.

– callingFacilityGroup:

  See Recommendation Q.1290.

– callingFacilityGroupMember:

  See Recommendation Q.1290.

– releaseCause:

  Indicates the cause of the disconnect.

– routeList:

  See Recommendation Q.1290.

– carrier:

  See Recommendation Q.1290.

– connectTime:

  Indicates the duration between the received answer indication from the called party side and the release of the connection.

3.3.29.2 Invoking Entity (SSF)

3.3.29.2.1 Normal procedures

SSF Preconditions:

1) Call origination attempt has been initiated.
2) Indication received from Terminating BCSM that the call is accepted and the terminating party has answered.
3) Disconnect indication received from an originating party, or received from the terminating party via the terminating BCSM.
4) For a TDP, call gapping or service filtering are not in effect.
5) DP criteria have been met.
6) For a TDP-R or a TDP-N, there is no existing control relationship.
7) For an EDP, there is an existing control relationship and the EDP O_Disconnect is armed.

SSF Postconditions:

1) For a TDP-R, basic call processing has been suspended at O_Disconnect DP, and a control relationship has been established.
2) For a TDP-N, call processing proceeds to the O_Null and Authorize Termination Attempt PIC, and no control relationship has been established.
3) For an EDP, as for EventReportBCSM procedure (see 3.3.22.2.1).

3.3.29.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.29.3 Responding Entity (SCF)

3.3.29.3.1 Normal procedure

SCF Precondition (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.29.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.30 ONoAnswer procedure

3.3.30.1 General description

This operation is sent from the SSF to the SCF at the O_No_Answer DP, after detecting a valid trigger condition, or to report an event requested by RequestReportBCSMEvent.

3.3.30.1.1 Parameters

– dPSpecificCommonParameters:

– callingPartyBusinessGroupID:

See Recommendation Q.1290. The SCF can use this parameter to select SLPs based on the group and for authorization purposes. The network operators can specify that this parameter should be used if their particular network has the information available.

– callingPartySubaddress:

See Q.931 Calling Party Subaddress.

– callingFacilityGroup:

See Recommendation Q.1290.

– callingFacilityGroupMember:

See Recommendation Q.1290.

– originalCalledPartyID:

See Q.762 Original Called Number signalling information.

– prefix:

  As defined in Recommendation Q.1290.

– redirectingPartyID:

  This parameter (if available) is the directory number of the last redirecting party.

– redirectionInformation:

  See Q.763 Redirection Information signalling information.

– routeList:

  routeList represents the route used.

– travellingClassMark:

  As defined in Recommendation Q.1290.

– carrier:

  As defined in Recommendation Q.1290.

3.3.30.2 Invoking Entity (SSF)

3.3.30.2.1 Normal procedure

SSF Preconditions (TDP and EDP):

1) Call origination attempt has been initiated.
2) Indication received that the terminating party has not answered within the specified time period.
3) Call gapping or service filtering are not in effect.
4) DP criteria have been met (TDP or EDP).
5) For a TDP-R, there is no existing control relationship.
6) For an EDP-R, there is an existing control relationship and the EDP O_No_Answer is armed.
7) For an EDP-N, there is an existing control or monitoring relationship and the EDP O_No_Answer is armed.

SSF Postconditions for TDP:

1) For a TDP-R, basic call processing has been suspended at O_No_Answer DP, and a control relationship has been established.
2) For a TDP-N, default exception handling has been provided, and no control relationship has been established. Use of TDP-N at this DP implies that there is no further alerting of the called party and that switch based no-answer treatments, as applicable, may be invoked.

SSF Postconditions for EDP:

1) The SSF-FSM stays in the state "Monitoring" if the message type was notification and there are still EDPs armed or a 'CallInformationReport' or 'ApplyChargingReport' requested.
2) The SSF-FSM moves to the state 'Idle' if the message type was notification and there are no more EDPs armed, no 'CallInformationReport' or 'ApplyChargingReport' are requested.
3) The SSF-FSM moves to the state "Waiting for Instructions" if the message type was request. Call processing is interrupted.

3.3.30.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.30.3 Responding Entity (SCF)

3.3.30.3.1 Normal procedure

SCF Precondition (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a 'CallInformationReport' or 'ApplyChargingReport' requested; or

  The SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no 'CallInformationReport' or 'ApplyChargingReport' are requested; or

  The SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.30.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.31 OriginationAttemptAuthorized procedure

3.3.31.1 General description

This operation is sent by the SSF after Authorization for call attempt is complete from BCSM O_Null and Authorize Origination Attempt PIC processing and a TDP has been detected.

3.3.31.1.1 Parameters

– serviceAddressInformation (triggerType, serviceKey, miscCallInfo):

  The triggerType indicates the particular event which caused call suspension. The miscCallInfo indicates that a request for instructions has been issued to the SCF and the category for the trigger (line, group, office). The serviceKey is used in identifying the service logic to be invoked.

– bearerCapability:

  See Q.762 User Service Information and Q.931 Bearer Capability information element. Refer to 6.4.4/Q.1214 for population rules for the bearer capability parameter.

– calledPartyNumber:

  See Q.762 Called Party Number signalling information. This parameter is used to identify the called party in the forward direction (i.e. it is used to populate the bearer signalling protocol’s Called Party Number information element).

– callingPartyNumber:

  See Q.762 Calling Party Number sigalling information. This parameter identifies the directory number of the originating line or the calling party number received in ISUP signalling. Refer to 6.4.4/Q.1214 for population rules for the callingPartyNumber parameter.

– callingPartysCategory:

  See Q.762 Calling Party's Category signalling information. This parameter indicates the type of the calling party (e.g. operator, pay phone, ordinary subscriber). Refer to 6.4.4/Q.1214 for population rules for the callingPartysCategory parameter.

– iPSSPCapabilities:

  See Recommendation Q.1290.

– iPAvailable:

  See Recommendation Q.1290.

– iSDNAccessRelatedInformation:

  Carries the same information as the protocol element ISUP Access Transport parameter in Recommendation Q.762. This parameter contains information elements provided across an ISDN interface. Refer to 6.4.4/Q.1214 for population rules for ISDN access related information.

– cGEncountered:

  This parameter indicates that the related call has passed call gapping.

– locationNumber:

  See Q.762 Location Number signalling information. This parameter is used to convey the geographical area address for mobility services. It is used when the 'callingPartyNumber' does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

– serviceProfileIdentifier:

  See Annex A/Q.932. This parameter allows ISDN to support identification and selection of specific terminals on a multi-point user-network interface to support multiple user service profiles when Q.931 information elements are not sufficient. See 6.4.4/Q.1214 for population rules for serviceProfileIdentifier.

– terminalType:

  See Recommendation Q.1290. Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

– chargeNumber:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules for chargeNumber.

– servingAreaID:

  See Recommendation Q.1290.

– dialledDigits:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules for dialledDigits.

– callingPartyBusinessGroupID:

  See Recommendation Q.1290.

– callingPartySubaddress:

  See Q.931 Calling Party Subaddress.

– callingFacilityGroup:

  See Recommendation Q.1290.

– callingFacilityGroupMember:

  See Recommendation Q.1290.

– travellingClassMark:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules for the travellingClassMark parameter.

– Carrier:

  See Recommendation Q.1290.

3.3.31.2 Invoking Entity (SSF)

3.3.31.2.1 Normal procedure

SSF Preconditions:

1) Call processing has progressed through the O_Null and Authorize Origination Attempt PIC of the O-BCSM and the call origination attempt has been authorized.

2) Call gapping and SS No. 7 overload are not in effect for the call, and the call is not to be filtered.

3) A trigger is armed as a TDP at the Origination_Attempt_Authorized detection point.

SSF Postconditions:

1) For TDP-R, a control relationship has been established and the SSF waits for instructions from the SCF.

2) For TDP-N, call processing continues at the Collect_Info PIC.

3.3.31.2.2 Error handling

If the destination SCF is not accessible, then the call is given final treatment (other treatments are for further study). On expiration of T_{SSF} before receiving any operation, the SSF aborts the interaction with the SCF and the call is given final treatment (e.g. routing to a final announcement). If the calling party abandons after the sending of the OriginationAttemptAuthorized operation, then the SSF aborts the control relationship after the first answer message from the SCF has been received: the Transaction ID is held open until T_{SSF} expires.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.31.3 Responding Entity (SCF)

3.3.31.3.1 Normal procedure

SCF Precondition:

– No control relationship exists.

SCF Postconditions:

1) An SLPI has been invoked; a control relationship is established for TDP-R.

2) For TDP-N: None.

On receipt of the OriginationAttemptAuthorized operation, the SCSM moves from "Idle" state to the state "Preparing SSF Instructions". A control relationship to the related SSF is created. A Service Logic Program Instance (SLPI) is invoked for processing the OriginationAttemptAuthorized operation. By means of this control relationship, the SCF may influence the Basic Call Processing in accordance with the service logic invoked. The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the SLPI (i.e. the requested IN service itself).

3.3.31.3.2 Error handling

If the OriginationAttemptAuthorized operation is rejected, then the SCSM remains in the same state. The maintenance function is informed and no SLPI is invoked. Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.32 PlayAnnouncement procedure

3.3.32.1 General description

This operation is used for inband interaction with an analogue user or for interaction with an ISDN user.

3.3.32.1.1 Parameters

- informationToSend:

  This parameter indicates an announcement, a tone or display information to be sent to the end user by the SRF.

  a) inbandInfo:

     This parameter specifies the inband information to be sent.

     – messageID:

       This parameter indicates the message(s) to be sent, this can be one of the following:

       i) elementaryMessageID:

          This parameter indicates a single announcement.

       ii) text:

           This parameter indicates a text to be sent. The text shall be transformed to inband information (speech) by the SRF. This parameter consists of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

       iii) elementaryMessageIDs:

            This parameter specifies a sequence of announcements.

       iv) variableMessage:

           This parameter specifies an announcement with one or more variable parts.

     – numberOfRepetitions:

       This parameter indicates the maximum number of times the message shall be sent to the end-user.

     – duration:

       This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

     – interval:

       This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

  b) tone:

     This parameter specifies a tone to be sent to the end-user.

     – toneID:

       This parameter indicates the tone to be sent.

     – duration:

       This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

  c) displayInformation:

     This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

NOTE – As the current signalling systems (DSS-1/ISUP) do not provide an indication whether or not information can be displayed by the user's terminal, in case of user interaction with an ISDN user two consecutive 'PlayAnnouncement' operations are sent. The first contains the display information, the second contains the inband information to be sent to the user. Since the execution of the display information by the SRF should take a limited amount of time, the inband information will be immediately sent by the SRF to the user, in sequence with the display information.

- disconnectFromIPForbidden:

  This parameter indicates whether or not the SRF should be disconnected from the user when all information has been sent.

- requestAnnouncementComplete:

  This parameter indicates whether or not a 'SpecializedResourceReport' shall be sent to the SCF when all information has been sent.

3.3.32.2 Invoking Entity (SCF)

3.3.32.2.1 Normal procedure

SCF Preconditions:

1) The SLPI detects that information should be sent to the user.
2) A connection between the user and a SRF has been established.
3) The SCSM-FSM is in the state "User interaction", substate "Waiting for response from the SRF".

SCF Postconditions:

1) If "RequestAnnouncementComplete" was set TRUE, the SCSM will stay in substate "Waiting for Response from the SRF" and wait for the "SpecializedResourceReport".
2) If "RequestAnnouncementComplete" was set FALSE and more information needs to be sent ("DisconnectFromIPForbidden" was set to TRUE), the SCSM will stay in substate "Waiting for Response from the SRF".
3) If "RequestAnnouncementComplete" was set FALSE and no more information needs to be sent ("DisconnectFromIPForbidden" was set to FALSE), the SCSM will move to the state "Preparing SSF Instructions".

3.3.32.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.32.3 Responding Entity (SRF)

3.3.32.3.1 Normal procedure

SRF Precondition:

– The SRSM-FSM is in the state "Connected" or in the state "User Interaction" if the SRF received previously an operation from the SCF.

SRF Postconditions:

1) The SRF sends the information to the user as indicated by "informationToSend".
2) The SRSM-FSM moves to the state "User Interaction" or

   remains in the same state.
3) If all information has been sent and "RequestAnnouncementComplete" was set TRUE, the SRSM sends a "SpecializedResourceReport" operation to the SCF.
4) If all information has been sent and "disconnectFromIPForbidden" was set FALSE, the SRSM disconnects the SRF from the user.

The announcement send to the end-user is ended in the following conditions:

– if neither "duration" or "numberOfRepetitions" is specified, then the network specific announcement ending conditions shall apply; or

– if "numberOfRepetitions" is specified, when all repetitions have been sent; or

– if duration is specified, when the duration has expired. The announcement is repeated until this condition is met; or

– if "duration" and "numberOfRepetitions" is specified, when one of both conditions is satisfied (whatever comes first).

3.3.32.3.2 Error handling

If a Cancel operation is received before or during the processing of the operation, then the operation is immediately cancelled and the error "Cancelled" is reported to the invoking entity.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.33 PromptAndCollectUserInformation procedure

3.3.33.1 General description

This operation is used to interact with a call party in order to collect information.

3.3.33.1.1 Parameters

a) collectedInfo

i) collectedDigits

- minimumNbOfDigits:

  If this parameter is missing, the default value is defined to be 1. The "minimumNbOfDigits" specifies the minimum number of valid digits to be collected.

- maximumNbOfDigits:

  This parameter should always be present and specifies the maximum number of valid digits to be collected. The following applies: "maximumNbOfDigits" >= "minimumNbOfDigits".

- endOfReplyDigit:

  This parameter indicates the digit used to signal the end of input.

  In case the "maximumNbOfDigits" = "minimumNbOfDigits", the "endOfReplyDigit" (could be present but) has no further meaning. This parameter can be one or two digits.

  In case the "maximumNbOfDigits" > "minimumNbOfDigits", the following applies:

  – If "endOfReplyDigit" is not present, the end of input is indicated:

    1) when the inter-digit timer expires; or

    2) when the number of valid digits received equals the "maximumNbOfDigits".

  – If "endOfReplyDigit" is present, the end of input is indicated:

    1) when the inter-digit timer expires; or

    2) when the end of reply digit is received; or

    3) when the number of valid digits received equals the "maximumNbOfDigits".

  When the end of input is attained, the collected digits are send from SRF to the SCF, including the "endOfReplyDigit" if received by the SRF.

  In the case the number of valid digits received is less than the "minimumNbOfDigits" when the inter-digit timer expires or when the end of reply digit is received, the input is specified as being erroneous.

- cancelDigit:

  If this parameter is present, the cancel digit can be entered by the user to request a possible retry. All digits already received by the SRF are discarded and the same "PromptAndCollectInformation" procedure is performed again, thus, e.g. the same announcement to request user information is given to the user and information is collected. This parameter can be one or two digits .

  If this parameter is not present, the user is not able to request a possible retry.

- startDigit:

  If this parameter is present, the start digit indicates the start of the valid digits to be collected. The digits that are received by the SRF before this start digit is received, are discarded and are not considered to be valid. This parameter can be one or two digits.

  If this parameter is not present, all received digits are considered to be valid.

- firstDigitTimeout:

  If this parameter is present, the first digit should be received by the SRF before the first-digit timer expiration. In case the first digit is not received before first-digit timer expiration, the input is regarded to be erroneous. After receipt of the first valid or non-valid input digit, the corresponding first-digit timer is stopped.

  If this parameter is not present, then the SRF uses a default value (network operator specific) for the first-digit timer in which the first valid or non-valid input digit is received.

  If "startDigit" is present, the first-digit timer is stopped after the start digit is received.

- interDigitTimeOut:

  If this parameter is present any subsequent valid or non-valid digit should be received by the SRF before the inter-digit timer expires. As result the inter-digit timer is reset and restarted.

  In case a subsequent valid or non-valid digit is not received before the inter-digit timer expires and the number of received valid digits is less than the "minimumNbOfDigits", the input is regarded to be unsuccessful.

  In case a subsequent valid or non-valid digit is not received before the inter-digit timer expires and the number of received valid digits is greater than the "minimumNbOfDigits", and less than or equal to the "maximumNbOfDigits", the input is regarded to be successful.

  If the "interDigitTimeOut" is not present, then the SRF uses a default value for the inter-digit time.

- errorTreatment:

  This optional parameter defines what specific action should be taken by the SRF in the event of error conditions occurring. The default value is reportErrorToSCF.

- interruptableAnnInd:

  This parameter is optional, where the default value is specified being TRUE.

  If this parameter is TRUE, the announcement is interrupted after the first valid or non-valid digit is received by the SRF. If the announcement is interrupted, a possible start-digit timer will not apply anymore. However, if the announcement has not been interrupted, a possible start-digit timer is started after the announcement has been finished.

  If this parameter is present and explicitly set to FALSE, the announcement will not be interrupted after the first digit is received by the SRF. The received digits during the announcement are discarded and considered to be non-valid. All other specified parameters ("minimumNbOfDigits", "maximumNbOfDigits", "endOfReplyDigit", etc.) do not apply before the announcement has been finished. The possible start-digit timer is started after the announcement has been finished.

- voiceInformation:

  This parameter is optional, where the default value is specified being FALSE. If the "voiceInformation" parameter is FALSE, all valid or non-valid digits are entered by DTMF.

  If this parameter is present and explicitly set to TRUE, calling user is required to provide all valid or non-valid information by speech. The SRF will perform voice recognition and translation of the provided information into digits. A possible end of reply digit will also have to be provided by speech.

- voiceBack:

  This parameter is optional, where the default value is specified being FALSE. If the "voiceBack" parameter is FALSE, no voice back information is given by the SRF.

  If this parameter is present and explicitly set to TRUE, the valid input digits received by the SRF will be announced back to the calling user immediately after the end of input is received. The non-valid input digits will not be announced back to the calling user.

  A possible end of reply digit is not voiced back.

b) disconnectFromIPForbidden:

This parameter indicates whether the SRF should initiate disconnection to the SSF-CCF after the interaction has been completed. If the parameter is not present or set to TRUE, the SRF shall not initiate disconnection.

c) informationToSend:

This parameter indicates an announcement, a tone or display information to be sent to the end user by the SRF.

i) inbandInfo:

This parameter specifies the inband information to be sent.

- messageID:

  This parameter indicates the message(s) to be sent, this can be one of the following:

  – elementaryMessageID:

    This parameter indicates a single announcement.

  – text:

    This parameter indicates a text to be sent. The text shall be transformed to inband information (speech) by the SRF. This parameter consist of two subparameters, messageContent and attributes. The attributes of text may consist of items such as language.

  – elementaryMessageIDs:

    This parameter specifies a sequence of announcements.

  – variableMessage:

    This parameter specifies an announcement with one or more variable parts.

- numberOfRepetitions:

  This parameter indicates the maximum number of times the message shall be sent to the end-user.

- duration:

  This parameter indicates the maximum time duration in seconds that the message shall be played/repeated. ZERO indicates endless repetition.

- interval:

  This parameter indicates the time interval in seconds between repetitions, i.e. the time between the end of the announcement and the start of the next repetition. This parameter can only be used when the number of repetitions is > 1.

ii) tone:

This parameter specifies a tone to be sent to the end-user.

- toneID:

  This parameter indicates the tone to be sent.

- duration:

  This parameter indicates the time duration in seconds of the tone to be sent. ZERO indicates infinite duration.

iii) displayInformation:

This parameter indicates a text string to be sent to the end-user. This information can not be received by a PSTN end-user.

NOTE – As the current signalling systems (DSS 1/ISUP) do not provide an indication whether or not information can be displayed by the user's terminal, in case of user interaction with an ISDN user, the 'displayInformation' parameter is not used in the 'PromptAndCollectUserInformation' operation. Instead, a 'PlayAnnouncement' operation containing the 'displayInformation' parameter followed by a 'PromptAndCollectUserInformation' operation containing inband information are sent to the user. Since the execution of the displayed information by the SRF should take a limited amount of time, the inband information will be immediately sent after by the SRF to the user, in sequence with the displayed information.

– digitsResponse:

This parameter contains the information collected from the end-user.

3.3.33.2 Invoking Entity (SCF)

3.3.33.2.1 Normal procedure

SCF Preconditions:

1) The SLPI detects that information should be collected from the end-user.
2) A connection between the end-user and a SRF has been established.
3) The SCSM-FSM is in state "User Interaction", substate "Waiting for Response from the SRF".

SCF Postconditions:

1) The collected information is received from the SRF as response to the 'PromptAndCollectUserInformation' operation.
2) If the 'disconnectFromIPForbidden' was set to FALSE, the SCSM-FSM will move to the state "Preparing SSF Instructions".
3) Otherwise the SCSM-FSM remains in the same state.

The SLPI may continue execution before the response is received from the 'PromptAndCollectUserInformation' operation, more than one operation may be sent to the SRF before the response is received. The 'disconnectFromIPForbidden' parameter may only be set to FALSE if the 'PromptAndCollectUserInformation' operation is the last operation sent to the SRF.

3.3.33.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.33.3 Responding Entity (SRF)

3.3.33.3.1 Normal procedure

SRF Precondition:

– The SRSM-FSM is in the state "Connected", or in state "User Interaction" if the SRF received previously an operation from the SCF.

SRF Postconditions:

1) The SRF has sent the information to the end-user as indicated by "informationToSend".
2) The collected information from the end-user is sent to the SCF as RETURN RESULT of the "PromptAndCollectUserInformation".
3) If the "disconnectFromIPForbidden" was set to FALSE, the SRF initiates a bearer channel disconnect to the SSF and the SRSM-FSM moves to the state "Idle".
4) Otherwise the SRSM-FSM moves to the state "User Interaction", or remains in the same state.

The announcement send to the end-user is ended in the following conditions:

- if neither "duration" or "numberOfRepetitions" is specified, then the network specific announcement ending conditions shall apply; or
- if "numberOfRepetitions" is specified, when all repetitions have been sent; or
- if duration is specified, when the duration has expired. The announcement is repeated until this condition is met; or
- if "duration" and "numberOfRepetitions" is specified, when one of both conditions is satisfied (whatever comes first).

The above conditions are overruled if the parameter "interruptableAnnInd" is not set to FALSE and the end-user has responded with a digit during the sending of the announcement. In this case the announcement is ended immediately. The above procedures apply only to inband information and tones send to the end-user, for "displayInformation" the end conditions are met upon sending, i.e. no interruption can occur.

The parameter "errorTreatment" specifies how the SRF shall treat the error. The default value "reportErrorToSCF" means that the error shall be reported to SCF by means of Return Error with "ImproperCallerResponse". The value "help" indicates that no error shall be reported to SCF but assistance shall be given to the end-user in form of a network dependent default announcement (which may be dependent on the context, i.e. the send message). The value "repeatPrompt" indicates that no error shall be reported to the SCF but the prompt shall be repeated to the end-user. The last two procedures shall only be done once per "PromptAndCollectUserInformation" operation.

NOTE – Processing "endOfInput": The receipt of any "endOfInput" condition (e.g. endOfReplyDigit, cancelDigit, firstDigitTimeout, interDigitTimeout) terminates immediately the ongoing input. In other words, when e.g. an endOfReplyDigit is received, the receipt of a subsequent cancelDigit will not be processed anymore.

3.3.33.3.2 Error handling

If a Cancel operation is received before or during the processing of the operation, then the operation is immediately cancelled and the error "Cancelled" is reported to the invoking entity.

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

If any of the parameter restrictions are violated (e.g. minimumNbOfDigits > maximumNbOfDigits), then an operation error has occurred.

3.3.34 ReleaseCall procedure

3.3.34.1 General description

This operation is used to tear down by the SCF an existing call at any phase of the call for all parties involved in the call. This operation may not be sent to an assisting SSF, except in the case of hand-off procedure.

3.3.34.1.1 Parameters

- Cause:

  A number giving an indication to the SSF about the reason of releasing this specific call. This may be used by SSF for generating specific tones to the different parties in the call or to fill in the 'cause' in the release message.

3.3.34.2 Invoking Entity (SCF)

3.3.34.2.1 Normal procedure

SCF Precondition:

- State 2.1, "Preparing SSF instructions" or State 2.3, "Waiting for Notification or Request".

SCF Postcondition:

- State 1, "Idle", if neither 'CallInformationReport' nor 'ApplyChargingReport' has to be received from the SSF. All resources (e.g. queue) related to the call are released by the SCF; or

State 2.3, "Waiting for Notification or Request" if a 'CallInformationReport' or 'ApplyChargingReport' still has to be received from the SSF.

3.3.34.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.34.3 Responding Entity (SSF)

3.3.34.3.1 Normal procedure

SSF Precondition:

– State C," Waiting for Instructions"; or

State F, "Monitoring" or hand-off waiting for instructions.

SSF Postcondition:

– "Idle", state a, after sending any outstanding 'CallInformationReport'. Possible armed EDPs are ignored. All connections and resources related to the call are released.

3.3.34.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.35 RemoveEntry procedure

3.3.35.1 General description

The X.500 'RemoveEntry' operation is used to request the SDF to remove a leaf entry (either an object entry or an alias entry) from the DIT. For a full description of the RemoveEntry operation, see 11.2/X511.

3.3.35.1.1 Parameters

See 11.2.2/X.511 and 11.2.3/X.511.

3.3.35.2 Invoking Entity (SCF)

3.3.35.2.1 Normal procedure

SCF Precondition:

– SCSM: "SDF Bound" or "Wait for Subsequent Requests".

SCF Postcondition:

– SCSM: "SDF Bound".

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the service logic to remove an entry from the SDF exists, an internal event [(e2) Request_to_SDF] occurs. Until the application process has not indicated with a delimitor (or a timer expiry) that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response [(E5) Response_from_SDF_with_Bind or (E4) Bind_Error] to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the RemoveEntry operation was discarded. In the State "SDF Bound", the response of the RemoveEntry operation [(E7) Response_from_SDF] causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the RemoveEntry operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the service logic to remove an entry from the SDF exists, an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response [(E7) Response_from_SDF] to the RemoveEntry operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the RemoveEntry operation or an error.

3.3.35.2.2 Error handling

Generic error handling for the operation related errors is described in 11.2.4/X.511 and 11.2.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.35.3 Responding Entity (SDF)

3.3.35.3.1 Normal procedure

SDF Precondition:

- SDSM: "SCF Bound"or "Bind Pending".

SDF Postcondition:

- SDSM "SCF Bound".

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a 'RemoveEntry' operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a 'RemoveEntry' operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before removing the entry item, the SDF takes the following actions:

- verify that the object to be removed exists and is a leaf entry;
- verify that the access rights to remove the entry are sufficient.

After the specified actions indicated above are successfully executed, the entry is removed from the SDF database. A null result is returned to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

3.3.35.3.2 Error handling

Generic error handling for the operation related errors is described in 11.2.4/X.511 and 11.2.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.36 RequestCurrentStatusReport procedure

3.3.36.1 General description

This operation is used to request the SSF to watch the current status (busy or idle) of particular termination resource.

3.3.36.1.1 Parameters

- resourceID:

  This parameter indicates the physical termination resource which is requested by the SCF to be monitored by the SSF. This parameter is one of the lineID, facilityGroupID, facilityGroupMemberID, or trunkGroupID.

- resourceStatus:

3.3.36.2 Invoking Entity (SCF)

3.3.36.2.1 Normal procedure

SCF Preconditions:

1) The SLPI has been determined that a "Request Current Status Report" operation has to be sent.
2) The SCME is in the state "Status Report Idle".

SCF Postcondition:

- The SCME is in the state "Waiting for SSF Response Status Report".

When the SPLI requests to watch the status of physical termination resource, the SCF sends "Request Current Status Report" operation to the SSF to request the SSF watch the status of a specific termination resource. Then the SCME is moved to the state "Waiting for SSF Response Status Report" from the state "Status Report Idle". The SCME is returned to the State "Status Report Idle", when the SCF receives a response from the SSF.

Clarification on the use of this operation within or outside the context of a call is for further study.

3.3.36.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.36.3 Responding Entity (SSF)

3.3.36.3.1 Normal procedure

SSF Precondition:

– The SSME is in one of the following states:

state ma: "IdleManagement";

state mb: "Non_Call_Associated_Treatment".

SSF PostCondition:

– The SSME is in the state "Non_Call_Associated_Treatment".

On receipt of this operation, the SSF watches the status of specific termination resource, and the SSF sends ReturnResult of this operation with result of the watched status to the SCF. If an error is occurred (e.g. the SSF cannot find specific termination resource), then the SSF sends ReturnError of this operation with proper error type to the SCF.

3.3.36.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.37 RequestFirstStatusMatchReport procedure

NOTE – For RequestEveryStatusChangeReport procedure, see 3.3.55.

3.3.37.1 General description

This operation is used to request the SSF start to monitor the status of particular termination resource is changed in the specific status (busy or idle).

3.3.37.1.1 Parameters

– resourceID:

This parameter indicates the physical termination resource which is requested by the SCF to be monitored by the SSF. This parameter is one of the lineID, facilityGroupID, facilityGroupMemberID, or trunkGroupID.

– resourceStatus:

This parameter indicates the status which is requested to detect by SCF of termination resource.

– correlationID:

Used by the SCF for correlation with a previous operation.

– monitorDuration:

This parameter indicates the maximum time duration for monitoring in the SSF.

3.3.37.2 Invoking Entity (SCF)

3.3.37.2.1 Normal procedure

SCF Preconditions:

1) The SLPI has been determined that a "Request First Status Match Report" operation has to be sent.

2) The SCME is in the state "Status Report Idle".

SCF Postcondition:

– The SCME is in the state "Waiting for SSF Response Status Report".

When the SPLI requests to monitor the status of physical termination resource, the SCF sends "Request First Status Match Report" operation to the SSF to request the SSF start to monitor a specific termination resource. Then the SCME is moved to the state "Waiting for SSF Response Status Report" from the state "Status Report Idle". After this, in the case that the SCF receives ReturnResult of this operation, the SCME is remained in same status. When the SCF receives ReturnError of this operation, the SCME is returned to the state "Status Report Idle".

The SCME is returned to the state "Status Report Idle", when the SCF is received a Status Report operation from the SSF.

Clarification on the use of this operation within or outside the context of a call is for further study.

3.3.37.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.37.3 Responding Entity (SSF)

3.3.37.3.1 Normal procedure

SSF Precondition:

– The SSME is in one of the following states:

state ma: "IdleManagement";

state mb: "Non_Call_Associated_Treatment".

SSF Postcondition:

– The SSME is in one of the following states:

state ma: "IdleManagement";

state mb: "Non_Call_Associated_Treatment".

On receipt of this operation, the SSF starts to monitor the status of particular termination resource and the SSF sends ReturnResult of this operation to the SCF. If an error is occurred (e.g. the SSF cannot find specific termination resource), then the SSF sends ReturnError of this operation with proper error type to the SCF.

The SSF continuously monitors the status of particular termination resource till the status is changed in specific state or timer which is specified by monitorDuration parameter is expired. If the SSF finds the change of status in specific state, the SSF sends "Status Report" operation to the SCF with monitorCondition parameter set to "statusReport". If the timer expiration is occurred, the SSF sends "StatusReport" operation to the SCF with monitorCondition parameter is set to "timerExpired". Another case, the SSF receives "Cancel Status Report" operation from the SCF, the SSF sends "Status Report" operation to the SCF with monitorCondition parameter set to "Canceled". After the SSF sends this operation, the SSME should move to the state "IdleManagement" unless there are another processes of Non-Call Associated operation, in which case the SSF should remain in the "Non_Call_Associated_Treatment".

3.3.37.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.38 RequestNotificationChargingEvent procedure

3.3.38.1 General description

This operation is used to instruct the SSF how to manage the charging events which are received from other FE's not under the control of the service logic instance. The operation supports the options to cope with the interactions concerning the charging (refer to Appendix II.4/Q.1214 "Charging scenarios"). As several connection configurations may be established during a call, a possibility exists for the RequestNotificationChargingEvent (RNC) operation to be invoked on multiple occasions. For each connection configuration an RNC may be used several times.

3.3.38.1.1 Parameters

- Sequence of ChargingEvent:

  This parameter contains a list of the charging events and the corresponding monitor types and corresponding legs. For each element in the list the following information elements are included:

  – eventTypeCharging:

    This subparameter indicates the charging event type. Its content is network operator specific, which may be "charge pulses" or "charge messages".

  – monitorMode:

    This subparameter indicates the monitorMode applicable for the corresponding "eventTypeCharging" subparameter. Monitor may be "interrupted", "notifyAndContinue" or "transparent".

  – legID:

    This subparameter indicates the leg id of the corresponding event type charging subparameter.

3.3.38.2 Invoking Entity (SCF)

3.3.38.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) An SLPI has determined that a "RequestNotificationChargingEvent" has to be sent by the SCF.

SCF Postconditions:

1) No FSM state transition;
2) SLPI execution may continue.

The SCSM-FSM is in state "Preparing SSF Instruction" or is in state "Queueing FSM". This operation is invoked by the SCF if a SLPI results in the instruction of SSF how to cope with the interactions concerning the charging. This causes no SCSM-FSM state transition.

3.3.38.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.38.3 Responding Entity (SSF)

3.3.38.3.1 Normal procedure

SSF Preconditions:

– SSF-FSM: State c: "Waiting for Instructions"; or
SSF-FSM State d: "Waiting for End of User Interaction"; or

SSF-FSM State e: "Waiting for End of Temporary Connection"; or

SSF-FSM State f: "Monitoring"; or

Assisting/Hand-off SSF-FSM State b: "Waiting for Instructions".

SSF Postcondition:

– No FSM state transition.

On receipt of this operation the SSF performs actions to cope with the interactions concerning the charging according to the information elements included in the operation. The requested charging event can be caused by:

a) another SLPI; or

b) another exchange.

Irrespective of by what the charging event is caused, the SSF performs one of the following actions on occurrence of the charging event (according the corresponding monitorMode):

– **Interrupted:**

Notify the SCF of the charging event using "EventNotificationCharging" operation, do not process the event or propagate the signal. However, call and existing charging processing will not be suspended in the SSF.

– **NotifyAndContinue:**

Notify the SCF of the charging event using "EventNotificationCharging", and continue processing the event or signal without waiting for SCF instructions (handled like EDP-N for BCSM events).

– **Transparent:**

Do not notify the SCF of the event. This ends the monitoring of a previously requested charging event.

Requested charging events are monitored until ended by a transparent monitor mode (or in the case of charging events) until the end of the connection configuration.

In the case that multiple "RequestNotificationChargingEvent" operations are received for the same connection configuration with the same "eventTypeCharging" and "legID", only the last received "monitorMode" will apply.

3.3.38.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.39 RequestReportBCSMEvent procedure

3.3.39.1 General description

This operation is used to request the SSF to monitor for a call-related event (e.g. BCSM events such as busy or no answer), then send a notification back to the SCF when the event is detected.

NOTE – If the RequestReportBCSMEvent requests arming of the current DP from which the call processing was suspended, the next occurrence of the DP encountered during BCSM processing will be detected (i.e. not the current one from which the call was suspended).

3.3.39.1.1 Parameters

- bcsmEvents:

  This parameter specifies the event or events of which a report is requested.

  a) eventTypeBCSM:

     This parameter specifies the type of event of which a report is requested. Values origAttemptAuthorized and termAttemptAuthorized are not valid for the eventTypeBCSM parameter.

b) monitorMode:

This parameter indicates how the event should be reported. When the 'monitorMode' is 'interrupted', the event shall be reported as a request, if the 'monitorMode' is 'notifyAndContinue', the event shall be reported as a notification, if the 'monitorMode' is 'transparent', the event shall not be reported.

c) legID:

This parameter indicates the party in the call for which the event shall be reported. SCF will use the option 'sendingSideID' only.

– sendingSideID:

The following values for 'legID' are assumed:

legID = 1 indicates the party that was present at the moment of the 'InitialDP' or a DP-specific operation (in case of a mid call trigger, the party causing the trigger), or the party that was created with an 'InitiateCallAttempt' operation.

legID = 2 indicates the party that was created with a 'Connect' operation, or in case of a mid call trigger, the party not causing the trigger.

If not included, the following defaults are assumed:

legID = 1 for the events CollectedInfo, AnalysedInformation, O-Abandonand T-Abandon.

legID = 2 for the events RouteSelectFailure, O-CalledPartyBusy, O-NoAnswer, O-Answer, T-Busy, T-NoAnswer and T-Answer.

The 'legID' parameter shall always be included for the events O-MidCall, O-Disconnect, T-MidCall and T-Disconnect.

d) dPSpecificCriteria:

This parameter indicates information specific to the EDP to be armed.

– numberOfDigits:

This parameter indicates the number of digits to be collected by the SSF for the CollectedInfo event. If the indicated number of digits is collected, SSF reports the event to the SCF.

– applicationTimer:

This parameter indicates the application timer for the NoAnswer event. If the user does not answer the call within the allotted time, the SSF reports the event to the SCF. This timer is expected to be shorter than the network no-answer timer.

e) bcsmEventCorrelationID:

Used by the SCF for correlation with a previous operation.

3.3.39.2 Invoking Entity (SCF)

3.3.39.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.

2) The SLPI has decided that a request for an event report BCSM is needed.

3) The SCSM-FSM is in the appropriate state to send 'RequestReportBCSMEvent'.

SCF Postconditions:

1) The SCSM-FSM remains in the same state.

2) SLPI execution continues.

3) If all EDPs have been disarmed and neither a CallInformationReport nor an ApplyChargingReport is pending, the controlrelationship with the concerned SSF is ended. If no other relationship persists, the SCSM shall return to "Idle" state.

3.3.39.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.39.3 Responding Entity (SSF)

3.3.39.3.1 Normal procedure

SSF Precondition:

– The SSF-FSM is in either the state "Waiting for Instructions" or the state "Monitoring".

SSF Postconditions:

1) The requested EDPs have been armed as indicated.
2) Previously requested events are monitored until ended by a transparent monitor mode, until the end of the call, until the EDPs are detected or until the corresponding leg is released.
3) The SSF-FSM remains in the same state.
4) If all EDPs have been disarmed and no "CallInformationReport" or "ApplyChargingReport" has been requested, the SSF-FSM moves to the state "Idle".

3.3.39.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.40 ResetTimer procedure

3.3.40.1 General description

This class 2 operation is used by the SCF to refresh the T_{SSF} application timer, in order to avoid the T_{SSF} time-out at the SSF.

3.3.40.1.1 Parameters

– timerID:

This parameter has a default value identifying the T_{SSF} timer.

– timerValue:

This parameter specified the value to which the T_{SSF} is to be set.

3.3.40.2 Invoking Entity (SCF)

3.3.40.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) An SLPI has determined by the $T_{SCF\text{-}SSF}$ guard timer expiration that the 'ResetTimer' operation has to be sent in order to avoid T_{SSF} time-out at the SSF.

SCF Postcondition:

– The SLPI reset the $T_{SCF\text{-}SSF}$ guard timer.

3.3.40.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.40.3 Responding Entity (SSF)

3.3.40.3.1 Normal procedure

SSF Preconditions:

1) Call origination attempt has been initiated.
2) Basic call processing has been suspended at a DP.

3) The SSF-FSM is in the "Waiting for Instruction" state or in the "Waiting for End of User Interaction" state or in the "Waiting for End of Temporary Connection" state or Assisting/hand-off waiting for instructions or Assisting/hand-off waiting for end of user interaction.

NOTE – Whether the T_{SSF} is used or not in the state "Waiting for End of User Interaction" or in the state "Waiting for End of Temporary Connection" is network operator dependent.

SSF Postconditions:

1) The T_{SSF} timer has been reset.

2) The SSF-FSM remains in the same state.

3.3.40.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.41 RouteSelectFailure Procedure

3.3.41.1 General description

This operation is sent by the SSF to the SCF after detecting a valid trigger condition at the RouteSelectFailure DP, or to report an event requested by RequestReportBCSMEvent. Refer to 4.2.2.2/Q.1214 for additional call modelling related semantics.

3.3.41.1.1 Parameters

– serviceAddressInformation (serviceKey, miscCallInfo, triggerType):

See Recommendation Q.1290. The triggerType indicates the particular event which caused call suspension. The miscCallInfo indicates that a request for instructions or a notification has been issued to the SCF and the category for the trigger (line, group, office). The serviceKey is used in identifying the service logic to be invoked.

– bearerCapability:

This parameter indicates the type of bearer capability connection to the user. Refer to 6.4.4/Q.1214 for population rules for the bearer capability parameter.

– calledPartyNumber:

See Q.762 Called Party Number signalling information. This parameter is used to identify the called party in the forward direction.

– callingPartyNumber:

See Q.762 Calling Party Number signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartyNumber parameter.

– callingPartysCategory:

See Q.762 Calling Party's Category signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartysCategory parameter.

– iPSSPCapabilities:

See Recommendation Q.1290.

– iPAvailable:

See Recommendation Q.1290.

– iSDNAccessRelatedInformation:

This parameter contains (possibly multiple) information elements as per Recommendation Q.931. See Access Transport Parameter signalling information in Recommendations Q.762, Q.763 and Q.931. Refer to 6.4.4/Q.1214 (Analysed Information) for population rules for ISDN access related information.

– cGEncountered:

See Recommendation Q.1290.

– locationNumber:

  See Q.762 Location Number signalling information. This parameter is used to convey the geographical area address for mobility services. It is used when the "callingPartyNumber" does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

– serviceProfileIdentifier:

  See Annex A/Q.932. Refer to 6.4.4/Q.1214 for population rules for serviceProfileIdentifier.

– terminalType:

  See Recommendation Q.1290. Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

– chargeNumber:

  See Recommendation Q.1290.

– servingAreaID:

  See Recommendation Q.1290.

– dialledDigits:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules.

– callingPartyBusinessGroupID:

  See Recommendation Q.1290.

– callingPartySubaddress:

  See Recommendation Q.931.

– callingFacilityGroup:

  See Recommendation Q.1290.

– callingFacilityGroupMember:

  See Recommendation Q.1290.

– failureCause:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules.

– originalCalledPartyID:

  See Q.762 Original Called Number signalling information.

– prefix:

  See Recommendation Q.1290. Refer to 6.4.4/Q.1214 for population rules.

– redirectingPartyID:

  This parameter (if available) is the directory number of the last redirecting party.

– redirectionInformation:

  See Q.763 Redirection Information signalling information.

– routeList:

  See Recommendation Q.1290.

– travellingClassMark:

  See Recommendation Q.1290.

– carrier:

  See Recommendation Q.1290.

3.3.41.2 Invoking Entity (SSF)

3.3.41.2.1 Normal procedure

SSF Preconditions:

1) Call origination attempt has been initiated.
2) Called Party Number is available and nature of address determined.
3) Call gapping or service filtering are not in effect for the call segment.
4) DP criteria have been met.
5) For a TDP-R, there is no existing control relationship influencing the call segment.

SSF Postconditions:

1) For a TDP-R, basic call processing has been suspended at Route_Select_Failure DP, and a control relationship has been established.
2) For a TDP-N, basic call processing proceeds at O_Exception, and no control relationship has been established.
3) For an EDP, as for EventReportBCSM procedure (see 3.3.22.2.1).

3.3.41.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.41.3 Responding Entity (SCF)

3.3.41.3.1 Normal procedure

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.
2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.
2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.
3) For a TDP-R, an SSF instruction is being prepared.
4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.41.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.42 Search procedure

3.3.42.1 General description

The X.500 'Search' operation is used to search a portion of the SDF resident DIT for entries of interest and to return selected information from those entries. For a full description of the Search operation, see 10.2/X.511.

3.3.42.1.1 Parameters

See 10.2.2/X.511 and 10.2.3/X.511.

3.3.42.2 Invoking Entity (SCF)

3.3.42.2.1 Normal procedure

SCF Precondition:

1) SCSM: "SDF Bound" or "Wait for Subsequent Requests".

SCF Postcondition:

– SCSM: "SDF Bound".

When the SCSM is in the state "Wait for Subsequent Requests" and a need of the service logic to search and/or read information of the SDF exists, an internal event [(e2) Request_to_SDF] occurs. Until the application process has not indicated with a delimitor that the operation should be sent, the SCSM remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the SDF in a message containing a Bind argument. The SCSM waits for the response from the SDF. The reception of the response [(E5) Response_from_SDF_with_Bind or (E4) Bind_Error] to the Bind operation previously issued to the SDF causes a transition of the SCF to the state "SDF Bound" or to the state "Idle". When the SCSM has moved to state "Idle", the Search operation was discarded. In the State "SDF Bound", the response of the Search operation [(E7) Response_from_SDF] causes a transition of the SCF to the same state ("SDF Bound"). It may be either the result of the Search operation or an error.

When the SCSM is in the state "SDF Bound" and a need of the service logic to search and/or read information of the SDF exists, an internal event occurs. This event, called (e6) Request_to_SDF causes a transition to the same state "SDF Bound" and the SCSM waits for the response from the SDF. The reception of the response [(E7) Response_from_SDF] to the Search operation previously issued to the SDF causes a transition of the SCF to the same state "SDF Bound". The response from the SDF may be either the result of the Search operation or an error.

3.3.42.2.2 Error handling

Generic error handling for the operation related errors is described in 10.2.4/X.511 and 10.2.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.42.3 Responding Entity (SDF)

3.3.42.3.1 Normal procedure

SDF Precondition:

1) SDSM: "SCF Bound"or "Bind Pending".

SDF Postcondition:

– SDSM "SCF Bound".

When the SDF is in the state "Bind Pending", the external event (E3) Request_from_SCF caused by the reception of a 'Search' operation from the SCF occurs. The SDF does not proceed to the operation until a Bind operation has been successfully executed. It remains in the same state.

When the SDF is in the state "SCF Bound", the external event (E7) Request_from_SCF caused by the reception of a 'Search' operation from the SCF occurs. The SDF waits for the response to the operation.

On the receipt of the event (E7) and before retrieving the data as specified in the operation parameters, the SDF takes the following actions:

- verify that the objects accessed by the request exists;
- verify that the user on behalf of whom the request is performed has sufficient access rights to access the objects and attributes reached during the execution of the operation;
- verify that attributes on which an operation should be performed exist in the object.

After the specified actions indicated above are successfully executed, the SDF returns all the possible attributes that satisfy the retrieval criteria to the SCF. The sending of the result corresponds to the event (e6) Response_to_SCF.

3.3.42.3.2 Error handling

Generic error handling for the operation related errors is described in 10.2.4/X.511 and 10.2.5/X.511 and the TCAP services that are used for reporting operating errors are described in 2.2.2.

3.3.43 SelectFacility procedure

3.3.43.1 General description

This operation is sent by the SCF to the SSF and requests the SSF to perform the terminating basic call processing actions to select the terminating line if it is idle, or selects an idle line from a multi-line hunt group, or selects an idle trunk from a trunk group, as appropriate. If no idle line or trunk is available, the SSF determines that the terminating facility is busy. Refer to 4.2.2.2/Q.1214 for additional call modelling related semantics.

3.3.43.1.1 Parameters

- alertingPattern:

  See Recommendation Q.1290. It only applies if the network signalling supports this parameter or if SSF is the terminating local exchange for the subscriber.

- destinationNumberRoutingAddress:

  This parameter contains the called party number towards which the call is to be routed. The encoding of the parameter is defined in Recommendation Q.763.

- iSDNAccessRelatedInformation:

  This parameter contains (possibly multiple) information elements as per Recommendation Q.931. See Access Transport Parameter signalling information in Recommendations Q.762, Q.763 and Q.931. Refer to 6.4.4/Q.1214 (Analysed Information) for population rules for ISDN access related information.

- calledFacilityGroup:

  See Recommendation Q.1290.

- calledFacilityGroupMember:

  See Recommendation Q.1290.

- originalCalledPartyID:

  See Q.762 Original Called Number signalling information.

3.3.43.2 Invoking Entity (SCF)

3.3.43.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) An SLPI has determined that a “SelectFacility” is to be sent by the SCF.

SCF Postcondition:

- SLPI execution may continue.

In the SCSM-FSM state “Preparing SSF Instructions”, this operation is invoked by an SCF if the service logic results in the request to an SSF to route a call to a specific destination and to continue call processing at the Select_Facility_and_Present_Call PIC. If no event monitoring has been requested and no reports (CallInformationReport and ApplyChargingReport) have been requested in a previously sent operation, an SCSM-FSM transition to the state “Idle” occurs. Otherwise, if event monitoring has been requested or any report (CallInformationReport and ApplyChargingReport) has been requested, the SCSM-FSM transitions to the state “Waiting for Notification or Request”.

3.3.43.2.2 Error handling

If reject or error messages are received, then the SCSM informs the SLPI of the message and remains in the state “Preparing SSF Instructions”.

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.43.3 Responding Entity (SSF)

3.3.43.3.1 Normal procedure

SSF Preconditions:

1) Call termination attempt has been initiated.

2) Basic call processing has been suspended at a DP.

3) The SSF waits for instructions.

SSF Postconditions:

1) The SSF performs call processing action to route the call to the specified destination and applies the appropriate alerting pattern.

2) In the T-BCSM, call processing resumes at the Select_Facility _and_Present_Call PIC.

3.3.43.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.44 SelectRoute procedure

3.3.44.1 General description

This operation requests the SSF to perform the originating basic call processing actions to determine routing information and select routing for a call, based either on call information available at the SSF or on call information provided by the SCF. Based on the routing information provided, the SSF attempts to select a primary route for the call. If the route is busy, the SSF attempts to select an alternative route. The SSF may fail to select a route for the call if all routes are busy.

3.3.44.1.1 Parameters

– destinationRoutingAddress:

  Represents a list of called party numbers (primary and alternates).

– alertingPattern:

  See Recommendation Q.1290. It only applies if the network signalling supports this parameter or if SSF is the terminating local exchange for the subscriber.

– correlationID:

  Used by the SCF to associate the 'AssistRequestInstructions' operation from the assisting SSF with the Request from the initiating SSF. The 'correlationID' is used in the context of a hand-off procedure and only if the correlation id is not embedded in the 'destinationRoutingAddress'.

– iSDNAccessRelatedInformation:

  Carries the same information as the protocol element ISUP Access Transport parameter in Recommendation Q.762.

– originalCalledPartyID:

  See Q.762 Original Called Number signalling information.

– routeList:

  See Recommendation Q.1290.

– scfID:

  Used by an assisting SSF to identify which SCF the 'destinationRoutingAddress' should be sent to.

– travellingClassMark:

  See Recommendation Q.1290.

– carrier:

  See Recommendation Q.1290. In this message, the carrier selection field is null (00000000) and Carrier ID indicates the carrier to use for the call.

3.3.44.2 Invoking Entity (SCF)

3.3.44.2.1 Normal procedure

SCF Preconditions:

1) Call origination attempt has been initiated.
2) Authority/ability to place outgoing call has been verified.
3) Destination information is available in the SSF or provided by the SCF.
4) Basic call processing has been suspended at one of the following DPs:
   - Origination_Attempt_Authorized;
   - Collected_Info;
   - Analysed_Info;
   - Route_Select_Failure;
   - O_Called_Party_Busy;
   - O_No_Answer;
   - O_Disconnect (called party disconnect only).
5) A control relationship has been established and the SLPI is processing the incoming request.

SCF Postconditions:

1) SLPI execution is terminated if no monitoring is requested.
2) SLPI execution is suspended pending the monitored event occuring, if monitoring is requested.

The SelectRoute message requests the SSF-CCF to resume call origination processing taking into account the address and routing information provided in the message parameters. Call Processing resumes at the ROUTING & ALERTING PIC.

3.3.44.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.44.3 Responding Entity (SSF)

3.3.44.3.1 Normal procedure

SSF Precondition:

– A TDP or EDP request has been invoked.

SSF Postcondition:

– The SSF performs the call processing actions to select the route.

The SSF-CCF shall resume call processing at the ROUTING & ALERTING PIC and proceed according to the BCSM description to attempt to complete the call over the designated route.

The following additional requirements also apply to the SelectRoute message:

- Alternate route/carrier selection as indicated in the SelectRoute shall proceed until either an idle facility is found or until processing of the call encounters a screening feature which blocks the call. (Network operator specific.)
- If a route requires a TCM, the value in the TCM parameter shall be used. If a TCM is required and the TCM parameter is not supplied in the SelectRoute, the SSF-CCF shall derive the TCM in accordance with requirements that apply assuming no IN involvement with the call.

SelectRoute and O_Called_Party_Busy, O_No_Answer:

- When the SSF-CCF receives the SelectRoute message in response to an O_Called_Party_Busy message the SSF-CCF shall do the following:

  1) The SSF-CCF shall release any resources that were used to process the call between the ANALYSE_INFORMATION and ROUTING & ALERTING PICs.

  2) The SSF-CCF shall resume call processing at the ROUTING & ALERTING PIC, and process the message as described in this procedure description.

  3) If the originating access is DSS 1, then the SSF-CCF shall not send another CALL PROCeeding message (INAP-DSS 1 interworking is for further study).

- When the SelectRoute message is received in response to an O_No_Answer TDP-Request Message, the SSF-CCF shall do the following:

  1) The SSF-CCF, if it is providing audible ringing tone to the calling party, shall remove this tone.

  2) The SSF-CCF shall release any resources that were used to process the call between the ANALYSE_INFORMATION and ROUTING & ALERTING PICs.

  3) The SSF-CCF shall resume call processing at the ROUTING & ALERTING PIC, and process the message as described in this procedure description.

  4) If the originating access is DSS 1, then the SSF-CCF shall not send another CALL PROCeeding message (INAP-DSS 1 interworking is for further study).

- When the SelectRoute message is received in response to an O_No_Answer EDP-Request message, the SSF-CCF shall do the following:

  1) The SSF-CCF, if it is providing audible ringing tone to the calling party, shall remove this tone.

  2) The SSF-CCF shall resume call processing at the ROUTING & ALERTING PIC, and process the message as described in this Recommendation.

  3) If the originating access is DSS 1, then the SSF-CCF shall not send another CALL PROCeeding message (INAP-DSS 1 interworking is for further study).

SelectRoute with OriginalCalledPartyID:

- When the SSF-CCF receives a SelectRoute message that contains the OriginalCalledPartyID parameter, the SSF-CCF shall map the received OriginalCalledPartyID parameter to, for example (ISUP interworking is for further study):

  1) the OriginalCalledNumber information element in the Facility information element if the SSF-CCF does not yet have a value for the OriginalCalledNumber AND if the SSF-CCF routes the call to an ISDN line (for Basic rate and Primary rate ISDN signalling);

  2) the OriginalCalledNumber parameter in the IAM if the SSF-CCF does not already have a value for the OriginalCalledNumber AND if the SSF-CCF routes the call to an SS7 trunk (for ISDN-UP signalling).

3.3.44.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.45 SendChargingInformation procedure

3.3.45.1 General description

This operation is used to instruct the SSF on the charging information to be sent by the SSF. The sending of charging information can either be by charge pulses or signalling or internal if SSF is located in the Local Exchange (LE). In the LE, either charge meter can be updated or a standard call record created. A possibility exists for the SendChargingInformation (SCI) operation to be invoked on multiple occasions. The charging scenario supported by this operation is: 3.2 (refer to Appendix II/Q.1214, "Charging scenarios").

NOTE – The interworking between SSF and PSTN is network operator specific. This operation has much PSTN/IN interactions.

3.3.45.1.1 Parameters

- sCIBillingChargingCharacteristics:

  This parameter indicates billing and/or charging characteristics. Its content is network operator specific. Depending on the applied charging scenario, the following information elements can be included (refer to Appendix II/Q.1214, "Charging scenarios"):

  – charge level (scenario 3.2);

  – chargePulses;

  – chargeMessages.

- partyToCharge:

  This parameter indicates where the charging information must be sent.

3.3.45.2 Invoking Entity (SCF)

3.3.45.2.1 Normal procedure

SCF Preconditions:

1) A control relationship exists between the SCF and the SSF.
2) An SLPI has determined that a "SendChargingInformation" has to be sent by the SCF.

SCF Postconditions:

1) No FSM state transition;
2) SLPI execution may continue.

The SCSM-FSM is in state "Preparing SSF Instruction" or is in state "Queueing FSM". The SendChargingInformation procedure shall be invoked by the SCF in accordance with the demands of the SLPI for relevant charging information. If appropriate this information shall be sent back down the call path.

This causes no SCSM-FSM state transition.

3.3.45.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.45.3 Responding Entity (SSF)

3.3.45.3.1 Normal procedure

SSF Preconditions:

– SSF-FSM: State c: "Waiting for Instructions"; or
SSF-FSM: State d: "Waiting for End of User Interaction"; or
SSF-FSM: State e: "Waiting for End of Temporary Connection"; or

SSF-FSM: State f: "Monitoring"; or

Assisting/hand-off SSF-FSM State b: "Waiting for Instructions".

SSF Postcondition:

– No FSM state transition.

On receipt of this operation the SSF performs actions to send the charging information. The sending of charging information can either be by charge pulses or signalling or internal if SSF is located in LE. In the LE, either charge metre can be updated or a standard call record created. The interworking between SSF and PSTN is network operator specific. This operation has much PSTN/IN interactions.

For instance, by sending an operation "SendChargingInformation" the SCF instructs the SSF to initiate the PSTN/ISDN charging functions according to the given information about the charging level to use.

The charging level can be determined either by one of the following functions:

a) the SCF; or

b) the SSF; or

c) the charging function in a succeeding exchange.

In case the SCF has determined the charging level, the "SendChargingInformation" operation contains the charging level to be applied.

In case the SSF determines the charging level, the "SendChargingInformation" operation contains the parameters to determine the charging level.

If the charging level was determined by the IN (SCF or SSF), the SSF provides the charging level to be applied to the PSTN/ISDN charging functions [cases a) and b)].

In case c, the charging level is determined in a succeeding exchange. The "SendChargingInformation" operation either contains the corresponding parameters indicating this fact or the SSF detects during trying to determine the charging level based on SCF provided parameters that the charging level shall be determined in a succeeding exchange. Based on already existing PSTN/ISDN capabilities, the SSF provides the PSTN/ISDN charging functions with the necessary information and backward charge messages shall be transferred down the call path when allowed by the SCF (generated by a succeeding exchange, for example an international gateway).

In the scenario described above, charging/billing is performed by means of existing mechanisms of the PSTN/ISDN initiated and controlled by the IN.

That means the determination of the charging method – on-line or off-line – and the items to be charged for shall be done in the basic network, just like the charge generation and the charge registration.

3.3.45.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.46 ServiceFilteringResponse procedure

3.3.46.1 General description

This operation is used to report the values of counters specified in a previous sent 'ActivateServiceFiltering' operation to the SCF.

3.3.46.1.1 Parameters

– countersValue:

The parameter contains the count of calls filtered during the filtering period. It is a list of counter identifications and the related values.

– filteringCriteria:

This parameter is used to address the concerned service logic at the SCF.

– responseCondition:

This parameter is used to identify the reason why the ServiceFilteringResponse is sent.

intermediateResponse indicates that service filtering is active, a call is received and the interval timer is expired; or

that service filtering is active and the threshold value, 'numberOfCalls', is reached.

lastResponse indicates that the duration time is expired and service filtering is stopped, or that the stop time is met and service filtering is stopped.

3.3.46.2 Invoking Entity (SSF)

3.3.46.2.1 Normal procedure

SSF Preconditions:

1) Service filtering is running and the interval time is expired and a call is received; or
2) Service filtering is running and the threshold value is reached; or
3) Service filtering has been finished (duration time expired or stop time met); or
4) The operation 'ActivateServiceFiltering' is received and encounters an active service filtering entity.

SSF Postcondition:

– Service filtering proceeds or is ended depending on the duration time.

The SSF sends the 'ServiceFilteringResponse' operation to the SCF. The 'filteringCriteria' parameter is provided to enable the addressing of the concerned service logic at the SCF.

Before 'ServiceFilteringResponse' is sent, it is checked whether call gapping criteria are met. If so, the 'ServiceFiltering-Response' is not sent and the counting continues without resetting the counters. The last 'ServiceFilteringResponse' (stop time is met or duration time expired) is sent without checking any call gap criteria.

After sending 'ServiceFilteringResponse' the service filtering counters are reset.

If service filtering proceeds after sending 'ServiceFilteringResponse' (e.g. interval time expired), the SSME-FSM remains in the state "Non_Call_Associated_Treatment".

If service filtering is stopped after sending 'ServiceFilteringResponse' (duration time expired or stop time is met), then the SSME-FSM moves to the "Idle Management" state. All allocated resources are released, i.e. the SSME-FSM is removed as well.

3.3.46.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.46.3 Responding Entity (SCF)

3.3.46.3.1 Normal procedure

SCF Preconditions:

1) Service filtering is running.
2) The SCME is in the state "Waiting for Service Filtering Response".

SCF Postcondition:

– The SCME forwards the received counter values to the SLPI.

The operation is handled by the Service Filtering FSM part of the SCF Management Entity (SCME). The SCME passes the received counter values to the SLPI where they are added to previously received counter values.

The "filteringCriteria" parameter as provided in "ServiceFilteringResponse" is used to address the SCME and the concerned service logic instance.

The Service Filtering FSM of the SCME remains in the state "Waiting For SSF Service Filtering Response" until the internal service filtering duration time in the SLPI expires. Then the SLPI informs the SCME about timer expiration. Now the SCME moves to the state "Service Filtering Idle".

3.3.46.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.47 SpecializedResourceReport procedure

3.3.47.1 General description

This operation is used as the response to a "PlayAnnouncement" operation when the announcement completed indication is set.

3.3.47.1.1 Parameters

– None.

3.3.47.2 Invoking Entity (SRF)

3.3.47.2.1 Normal procedure

SRF Preconditions:

1) The SRSM-FSM is in the state "User Interaction".
2) A "PlayAnnouncement" operation is being executed for which the parameter "RequestAnnouncementCompelete" was set TRUE.
3) All information has been sent to the user.

SRF Postconditions:

1) The SRSM-FSM remains in the same state.
2) If the "DisconnectFromIPForbidden" parameter was set FALSE, the SRSM initiates a bearer channel disconnect sequence to the SSF using the applicable bearer channel signalling system after sending the "SpecializedResourceReport" operation to the SCF. The SRSM-FSM moves to the state "Idle".

3.3.47.2.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.47.3 Responding Entity (SCF)

3.3.47.3.1 Normal procedure

SCF Precondition:

– The SCSM-FSM is in the state "User Interaction", substate "Waiting for response from the SRF".

SCF Postconditions:

1) The SCSM-FSM remains in the same state.
2) If the "SpecializedResourceReport" relates to a "PlayAnnouncement" operation with permission of SRF initiated disconnection, the SCSM-FSM moves to the state "Preparing SSF Instructions".

3.3.47.3.2 Error handling

Operation related error handling is not applicable, due to class 4 operation.

3.3.48 StatusReport procedure

3.3.48.1 General description

This operation is used to notify the result of monitoring that is requested by "RequestFirstStatusMatchReport" or “RequestEveryStatusChangeReport” operation to the SCF.

3.3.48.1.1 Parameters

- resourceStatus:

  this parameter indicates the detected status of physical termination resource which is requested to monitor by the SCF.

- CorrelationID:

  Used by the SCF for correlation with a previous operation.

- ResourceID:

- ReportCondition:

  Specifies the reason of issuing this operation (normal status report or monitor duration expired).

3.3.48.2 Invoking Entity (SSF)

3.3.48.2.1 Normal procedure

SSF Preconditions:

- The SSME is in the state "Non_Call_Associated_Treatment".

SSF Postconditions:

- The SSME is in one of the following states:

  state ma: "IdleManagement";

  state mb: "Non_Call_Associated_Treatment".

The SSF sends "StatusReport" operation to the SCF, when the following events are occurred:

- The SSF find the change of status in specific state.
- The SSF received "CancelStatusReportRequest" operation from the SCF.
- The timer expiration is occurred in the SSF.

After the SSF sends this operation, the SSME should move to the state "IdleManagement" unless there are other processes of Non-Call Associated operation, in which case the SSF should remain in the "Non_Call_Associated_Treatment".

Clarification on the use of this operation within or outside the context of a call is for further study.

3.3.48.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.48.3 Responding Entity (SCF)

3.3.48.3.1 Normal procedure

SCF Preconditions:

- The SCME is in the state "Waiting for SSF Response Status Report".

SCF Postconditions:

1) The SCME is in the state "Status Report Idle".
2) The SCF notifies SLPI of the result of resource monitoring in the SSF.

On receipt of this operation the SLPI which is expecting this operation will continue.

3.3.48.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.49 TAnswer procedure

3.3.49.1 General description

This operation is sent from the SSF to the SCF at the T_Answer DP, after detecting a valid trigger condition, or to report an event requested by RequestReportBCSMEvent.

3.3.49.1.1 Parameters

- dPSpecificCommonParameters:
  - calledPartyBusinessGroupID:

    See Recommendation Q.1290.
  - calledPartySubaddress:

    See Recommendation Q.931.
  - calledFacilityGroup:

    See Recommendation Q.1290.
  - calledFacilityGroupMember:

    See Recommendation Q.1290.

3.3.49.2 Invoking Entity (SSF)

3.3.49.2.1 Normal procedure

SSF Preconditions (TDP):

1) Incoming call received from originating BCSM.
2) Call has been accepted and the terminating party has answered.
3) For TDP, call gapping or service filtering are not in effect for the call segment.
4) DP criteria have been met.
5) For a TDP-R, there is no existing control relationship influencing the call segment.

SSF Preconditions (EDP):

1) For an EDP-R, there is an existing control relationship and the EDP T_Answer is armed.
2) For an EDP-N, there is an existing monitoring or control relationship and the EDP T_Answer is armed.

SSF Postconditions (TDP):

1) For a TDP-R, basic call processing has been suspended at T_Answer DP, and a control relationship has been established.
2) For a TDP-N, basic call processing proceeds at T_Answer PIC, and no control relationship has been established.

SSF Postconditions (EDP):

1) The SSF-FSM stays in the state "Monitoring" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested.
2) The SSF-FSM moves to the state "idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested.
3) The SSF-FSM moves to the state "Waiting for Instructions" if the message type was request. Call processing is interrupted.

3.3.49.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.49.3 Responding Entity (SCF)

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.49.3.1 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.50 TBusy procedure

3.3.50.1 General description

This operation is sent by the SSF to the SCF after detecting a valid trigger condition at the TBusy DP, or to report an event requested by RequestReportBCSMEvent. Refer to 4.2.2.2/Q.1214 for additional call modelling related semantics.

3.3.50.1.1 Parameters

– serviceAddressInformation (serviceKey, miscCallInfo, triggerType):

  See Recommendation Q.1290. The triggerType indicates the particular event which caused call suspension. The miscCallInfo indicates that a request for instructions or a notification has been issued to the SCF and the category for the trigger (line, group, office). The serviceKey is used in identifying the service logic to be invoked.

– bearerCapability:

  This parameter indicates the type of bearer capability connection to the user. Refer to 6.4.4/Q.1214 for population rules for the bearer capability parameter.

– calledPartyNumber:

  See Q.762 Called Party Number signalling information. This parameter is used to identify the called party in the forward direction.

– callingPartyNumber:

  See Q.762 Calling Party Number signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartyNumber parameter.

– callingPartysCategory:

  See Q.762 Calling Party's Category signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartysCategory parameter.

– iPSSPCapabilities:

  See Recommendation Q.1290.

– iPAvailable:

  See Recommendation Q.1290.

– iSDNAccessRelatedInformation:

  This parameter contains (possibly multiple) information elements as per Recommendation Q.931. See Access Transport Parameter signalling information in Recommendations Q.762, Q.763 and Q.931. Refer to 6.4.4/Q.1214 (Analysed Information) for population rules for ISDN access related information.

– cGEncountered:

  See Recommendation Q.1290.

– locationNumber:

  See Q.762 Location Number signalling information. This parameter is used to convey the geographical area address for mobility services. It is used when the "callingPartyNumber" does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

– serviceProfileIdentifier:

  See Annex A/Q.932. See 6.4.4/Q.1214 for population rules for serviceProfileIdentifier.

– terminalType:

  See Recommendation Q.1290. Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

– chargeNumber:

  See Recommendation Q.1290.

– servingAreaID:

  See Recommendation Q.1290.

– busyCause:

  See Recommendation Q.1290. See 6.4.4/Q.1214 for population rules.

– calledPartyBusinessGroupID:

  See Recommendation Q.1290.

– calledPartySubaddress:

  See Recommendation Q.931.

– originalCalledPartyID:

  See Q.762 Original Called Number signalling information.

– redirectingPartyID:

  This parameter (if available) is the directory number of the last redirecting party.

– redirectionInformation:

  See Q.763 Redirection Information signalling information.

– routeList:

  See Recommendation Q.1290.

– travellingClassMark:

  See Recommendation Q.1290.

3.3.50.2 Invoking Entity (SSF)

3.3.50.2.1 Normal procedure

SSF Preconditions:

1) Call termination attempt has been initiated.
2) Called Party Number is available and nature of address determined.
3) Call gapping or service filtering are not in effect for the call segment.
4) DP criteria have been met.
5) For a TDP-R, there is no existing control relationship influencing the call segment.

SSF Postconditions:

1) For a TDP-R, basic call processing has been suspended at T_Busy DP, and a control relationship has been established.
2) For a TDP-N, basic call processing proceeds at T_Exception, and no control relationship has been established.
3) For an EDP, as for EventReportBCSM procedure (See 3.3.22.2.1).

The SSF has sufficient information available associated with the terminating call portion. The SSF shall detect T_Busy when the terminating access is network-determined user-busy. The conditions that result in the detection of network-determined user-busy depend on the type of terminating access and subscribed services (an analogue line access, DSS 1, multiline hunt group, etc.).

3.3.50.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.50.3 Responding Entity (SCF)

3.3.50.3.1 Normal procedure

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.
2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.
2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.
3) For a TDP-R, an SSF instruction is being prepared.
4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.50.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.51 TDisconnect procedure

3.3.51.1 General description

This operation is sent by the SSF to the SCF after detecting a valid trigger condition at the T_Disconnect DP, or to report an event requested by RequestReportBCSMEvent. Refer to 4.2.2.2/Q.1214 for additional call modelling related semantics.

3.3.51.1.1 Parameters

– serviceAddressInformation (serviceKey, miscCallInfo, triggerType):

  See Recommendation Q.1290. The triggerType indicates the particular event which caused call suspension. The miscCallInfo indicates that a request for instructions or a notification has been issued to the SCF and the category for the trigger (line, group, office). The serviceKey is used in identifying the service logic to be invoked.

– bearerCapability:

  This parameter indicates the type of bearer capability connection to the user. Refer to 6.4.4/Q.1214 for population rules for the bearer capability parameter.

– calledPartyNumber:

  See Q.762 Called Party Number signalling information. This parameter is used to identify the called party in the forward direction.

– callingPartyNumber:

  See Q.762 Calling Party Number signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartynumber parameter.

– callingPartysCategory:

  See Q.762 Calling Party's Category signalling information. Refer to 6.4.4/Q.1214 for population rules for the callingPartysCategory parameter.

– iPSSPCapabilities:

  See Recommendation Q.1290.

– iPAvailable:

  See Recommendation Q.1290.

– iSDNAccessRelatedInformation:

  This parameter contains (possibly multiple) information elements as per Recommendation Q.931. See Access Transport Parameter signalling information in Recommendations Q.762, Q.763 and Q.931. Refer to 6.4.4/Q.1214 (Analysed Information) for population rules for ISDN access related information.

– cGEncountered:

  See Recommendation Q.1290.

– locationNumber:

  See Q.762 Location Number signalling information. This parameter is used to convey the geographical area address for mobility services. It is used when the "callingPartyNumber" does not contain any information about the geographical location of the calling party (e.g. origin dependent routing when the calling party is a mobile subscriber).

– serviceProfileIdentifier:

  See Annex A/Q.932. See 6.4.4/Q.1214 for population rules for serviceProfileIdentifier.

– terminalType:

  See Recommendation Q.1290. Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability (voice recognition, DTMF, display capability, etc.).

– chargeNumber:

  See Recommendation Q.1290. See 6.4.4/Q.1214 for population rules for chargeNumber.

– servingAreaID:

  See Recommendation Q.1290.

– calledPartyBusinessGroupID:

  See Recommendation Q.1290.

– calledPartySubaddress:

  See Recommendation Q.931 Called Party Subaddress.

– calledFacilityGroup:

  See Recommendation Q.1290.

– calledFacilityGroupMember:

  See Recommendation Q.1290.

– releaseCause:

  Indicates the cause of the disconnect.

– connectTime:

  Indicates the duration between the received answer indication from the called party side and the release of the connection.

3.3.51.2 Invoking Entity (SSF)

3.3.51.2.1 Normal procedures

SSF Preconditions:

1) Incoming call received from originating BCSM.
2) Call has been accepted and the terminating party has answered.
3) Disconnect indication received from terminating party, or received from originating party via the originating BCSM.
4) For a TDP, call gapping or service filtering are not in effect.
5) DP criteria have been met.
6) For a TDP-R or a TDP-N, there is no existing control relationship.
7) For an EDP, there is an existing control relationship and the EDP T_Disconnect is armed.

SSF Postconditions:

1) For a TDP-R, basic call processing has been suspended at T_Disconnect DP, and a control relationship has been established.
2) For a TDP-N, call processing proceeds to the T_Null & Authorize Termination Attempt PIC, and no control relationship has been established.
3) For an EDP, as for EventReportBCSM procedure (see 3.3.22.2.1).

3.3.51.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.51.3 Responding Entity (SCF)

3.3.51.3.1 Normal procedure

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

– For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

  the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

  the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.51.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP service which are used for reporting operation errors are described in 3.4.

3.3.52 TermAttemptAuthorized procedure

3.3.52.1 General description

This operation is sent from the SSF to the SCF at the TermAttemptAuthorized DP, after detecting a valid trigger condition, or to report an event requested by RequestReportBCSMEvent.

3.3.52.1.1 Parameters

– dPSpecificCommonParameters:

– calledPartyBusinessGroupID:

  See Recommendation Q.1290.

– calledPartySubaddress:

  See Recommendation Q.931.

– callingPartyBusinessGroupID:

  See Recommendation Q.1290.

– originalCalledPartyID:

  See Q.762 Original Called Number signalling information.

– redirectingPartyID:

Contains the directory number of the last redirecting party.

– redirectionInformation:

See Q.763 Redirection Information signalling information.

– routeList:

routeList represents the list of routes which would have been used in order to route the call. The network operators can specify that this IE should be used if their particular network has the information available.

– travellingClassMark:

See Recommendation Q.1290.

3.3.52.2 Invoking Entity (SSF)

3.3.52.2.1 Normal procedure

SSF Preconditions:

1) Incoming call received.
2) Authority to route the call to specified terminating resources verified.
3) Call gapping or service filtering are not in effect.
4) TDP criteria have been met.
5) For a TDP-R there is no existing control relationship.

SSF Postconditions:

1) For a TDP-R, basic call processing has been suspended at Term_Attempt_Authorized DP, and a control relationship has been established.
2) For a TDP-N, basic call processing proceeds at PIC Select_Facility&Present Call, and no control relationship has been established.

3.3.52.2.2 Error handling

If the destination SCF is not accessible, then the call is given final treatment (other treatments are for further study). If the calling party abandons after the sending of the TermAttemptAuthorized operation, then the SSF aborts the control relationship after the first answer message from the SCF has been received: the Transaction ID is held open until T_{SSF} expires.

Generic error handling for the operation related errors is described in 3.2 the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.52.3 Responding Entity (SCF)

3.3.52.3.1 Normal procedure

SCF Preconditions:

1) For TDP, none.
2) EDP case does not apply.

SCF Postconditions:

1) An SLPI has been invoked.
2) For a TDP-R, an SSF instruction is being prepared.

On receipt of the TermAttemptAuthorized operation, the SCSM moves from "Idle" state to the state "Preparing SSF Instructions". A control relationship to the related SSF is created. A Service Logic Program Instance (SLPI) is invoked for processing the TermAttemptAuthorized operation. By means of this control relationship, the SCF may influence the Basic Call Processing in accordance with the service logic invoked. The actions to be performed in the SLPI depend on the parameters conveyed via this operation and the SLPI (i.e. the requested IN service itself).

3.3.52.3.2 Error handling

If the TermAttemptAuthorized operation is rejected, then the SCSM remains in the same state. The maintenance function is informed and no SLPI is invoked. Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.53 TNoAnswer procedure

3.3.53.1 General description

This operation is sent from the SSF to the SCF after detecting a valid trigger condition at the T_No_Answer DP, or to report an event requested by RequestReportBCSMEvent.

This operation requests the SSF-CCF to send a T_No_Answer TDP-Request message when it encounters a T_No_Answer trigger.

3.3.53.1.1 Parameters

- dPSpecificCommonParameters:
- calledPartyBusinessGroupID:

  See Recommendation Q.1290.
- calledPartySubaddress:

  See Recommendation Q.931.
- calledFacilityGroup:

  See Recommendation Q.1290.
- calledFacilityGroupMember:

  See Recommendation Q.1290.
- originalCalledPartyID:

  See Q.762 Original Called Number signalling information.
- redirectingPartyID:
  Contains the directory number of the last redirecting party.
- redirectionInformation:
  See Q.763 Redirection Information signalling information.
- travelingClassMark:
  See Recommendation Q.1290.

3.3.53.2 Invoking Entity (SSF)

3.3.53.2.1 Normal procedure

SSF Preconditions (TDP):

1) Incoming call has been received.
2) The terminating party has not answered within a specified time period.
3) Call gapping or service filtering are not in effect.
4) DP criteria have been met.
5) For a TDP-R, there is no existing control relationship.

SSF Preconditions (EDP):

1) For an EDP-R, there is an existing control relationship and the T_No_Answer EDP is armed.
2) For an EDP-N, there is an existing control or monitoring relationship and the T_No_Answer EDP is armed.

SSF Postconditions (TDP):

1) For a TDP-R, basic call processing has been suspended at T_No_Answer DP, and a control relationship has been established.
2) For a TDP-N, default exception handling has been provided, and no control relationship has been established.

SSF Postconditions (EDP):

1) The SSF-FSM stays in the state "Monitoring" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested.

2) The SSF-FSM moves to the state "idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested.

3) The SSF-FSM moves to the state "Waiting for Instructions" if the message type was request. Call processing is interrupted.

3.3.53.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.53.3 Responding Entity (SCF)

3.3.53.3.1 Normal procedure

SCF Preconditions (TDP):

– None.

SCF Preconditions (EDP):

1) For an EDP-R at the SSF, an existing control relationship is in place and an SLPI is running.

2) For an EDP-N, an existing monitoring relationship is in place and an SLPI is running.

SCF Postconditions (TDP):

1) An SLPI has been invoked.

2) For a TDP-R, a control relationship is established, and an SLPI has been invoked.

3) For a TDP-R, an SSF instruction is being prepared.

4) For a TDP-N, no relationship is established. An SLPI has been invoked, executes and terminates.

SCF Postconditions (EDP):

1) For an EDP, the SCSM-FSM stays in the substate "Waiting for Notification or Request" if the message type was notification and there are still EDPs armed or a "CallInformationReport" or "ApplyChargingReport" requested; or

   the SCSM-FSM moves to the state "Idle" if the message type was notification and there are no more EDPs armed, no "CallInformationReport" or "ApplyChargingReport" are requested; or

   the SCSM-FSM moves to the state "Preparing SSF Instructions" if the message type was request.

3.3.53.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services which are used for reporting operation errors are described in 3.4.

3.3.54 Unbind procedure

3.3.54.1 General description

The X.500 'Unbind' operation is used by the SDF to end an authenticated association between an SCF and an SDF on behalf of the end user. For a full description of the Unbind operation, see 8.2/X.511.

3.3.54.1.1 Parameters

None.

3.3.54.2 Invoking Entity (SCF)

3.3.54.2.1 Normal procedure

SCF Preconditions:

– SCSM: "SDF Bound".

SCF Postconditions:

– SCSM: "Idle".

The SCSM has previously initiated a successful Bind operation to the SDF directory. It is in state "SDF Bound". The service logic determines that the authenticated access to the SDF is to be terminated. It issues an Unbind operation [(e8) Unbind_request] that causes the SCSM to transit back to the state "Idle".

3.3.54.2.2 Error handling

The 'Unbind' operation does not have operation related errors.

3.3.54.3 Responding Entity (SDF)

3.3.54.3.1 Normal procedure

SDF Preconditions:

– SDSM: "SCF Bound".

SDF Postconditions:

– SDSM: "Idle".

A Bind operation was previously issued and the SDSM is in State "SCF Bound" waiting for a request from the SCF and/or performing an operation. The reception of the Unbind operation causes a transition to State "Idle" with the transition (E5) Unbind_from_SCF.

3.3.54.3.2 Error handling

The 'Unbind' operation does not have operation related errors.

3.3.55 RequestEveryStatusChangeReport procedure

3.3.55.1 General description

This operation is used to request the SSF start to monitor every change of the busy/idle status of a particular termination.

3.3.55.1.1 Parameters

– resourceID:

  This parameter indicates that physical termination resource which is requested by the SCF to be monitored by the SSF. This parameter is one of the lineID, facilityGroupID, faliciltyGroupMemberID, or trunkGroupID.

– correlationID:

  This parameter is used by the SCF to associate the “StatusReport” from the SSF with the request in the SCF.

– monitorDuration:

  This parameter indicates the maximum time duration for monitoring in the SSF.

3.3.55.2 Invoking Entity (SCF)

3.3.55.2.1 Normal procedure

SCF Preconditions:

1) The SLPI has been determined that a "Request Every Status Change Report" operation has to be sent.
2) The SCME is in the state "Status Report Idle".

SCF Postcondition:

– The SCME is in the state "Waiting for SSF Response Status Report".

When the SPLI requests to monitor every change of the busy/idle status of a physical termination resource, the SCF sends "Request Every Status Change Report" operation to the SSF to monitor every change of the status of a particular termination resource. Then the SCME is moved to the state "Waiting for SSF Resource Status Report" from the state "Status Report Idle". After this, in the case that the SCF receives ReturnResult of this operation, the SCME remains in same state. When the SCF receives ReturnError of this operation, the SCME is returned to the state "Status Report Idle".

When the SCF receives a Status Report operation from the SSF, the SCME remains in the state “Waiting for SSF Resource Status Report”. Alternatively when the SCF receives a “Status Report” operation with the monitorCondition parameter set to “timerExpired” or “cancelled” from SSF, the SCME is returned to the state “Status Report Idle”.

Clarification on the use of this operation within or outside the context of a call is for further study.

3.3.55.2.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.3.55.3 Responding Entity (SSF)

3.3.55.3.1 Normal procedure

SSF Precondition:

– The SSME is in one of the following states:

state ma: "IdleManagement";

state mb: "Non-Call Associated Treatment".

SSF Postcondition:

– The SSME is in one of the following states:

state ma: "IdleManagement";

state mb: "Non-Call Associated Treatment".

On receipt of this operation, the SSF starts to monitor every change of the busy/idle status of a particular termination resource. If an error is occurred (e.g. the SSF can not find the specific termination resource), then the SSF sends ReturnError of this operation with proper error type to the SCF.

The SSF continuously monitors every change of the busy/idle status of a particular termination resource until a timer which is specified by monitorDuration parameter expires. Whenever the SSF finds a change of status, the SSF sends the "Status Report" Operation to the SCF with monitorCondition parameter set to "statusReport". After sending this operation, the SSF should remain in the “Non-Call Associated Treatment” state. If timer expiration occurs, the SSF sends the "StatusReport" Operation to the SCF with monitorCondition parameter set to "timerExpired". Alternatively, when the SSF receives a "Cancel Status Report" operation from the SCF, the SSF sends the "Status Report" Operation to the SCF with monitorCondition parameter set to "Cancelled". After the SSF sends this operation, the SSME should move to the state "IdleManagement" unless there are other processes of Non-Call Associated operation, in which case the SSF should remain in the state "Non-Call AssociatedTreatment".

3.3.55.3.2 Error handling

Generic error handling for the operation related errors is described in 3.2 and the TCAP services used for reporting operation errors are described in 3.4.

3.4 Services assumed from TCAP

This subclause describes the procedures and TCAP primitives that shall be used for transmitting messages between SSF, SCF and SRF. For the SCF-SDF interface, refer to 2.2.2.2.

3.4.1 Normal procedures

This subclause describes the procedures and TCAP primitives that shall be used for transmitting messages between SSF, SCF and SRF 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) A dialogue shall be established – The TC-user issues a TC-BEGIN request primitive.

b) A dialogue shall be maintained – The TC-user issues a TC-CONTINUE request primitive.

c) 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:

   i) 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 Recommendation Q.774.

   - In the 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.

   ii) Pre-arranged End:

   - In the case, an entity is interested in possible ERROR or REJECT messages on response to sent operations leading to a termination of the relationship, the dialogue is ended with a TC-END request primitive (pre-arranged end) after the last associated operation timer expires. The receiving entity can end the dialogue with a TC-END request primitive (pre-arranged end) after successful processing of these operations (i.e. the relationship is terminated).

d) In general, the use of pre-arranged end shall be limited to the case for both communicating entities clearly recognizable that peer entity applies pre-arranged end. In all other cases, basic end shall be used.

e) 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 time out by means of a pre-arranged end. Upon reception of the TC-BEGIN indication primitive the receiving TC-user shall end the dialogue locally.

3.4.1.1 SSF-to-SCF messages

3.4.1.1.1 SSF-FSM related messages

A dialogue shall be established when the SSF-FSM moves from the state **Idle** to the state **Waiting for Instructions**. The relevant INAP operation, which can be the InitialDP operation or one of DP specific operations for TDP-R, shall be transmitted in the same message.

No dialogue shall be established when the SSF-FSM moves from the state **Idle** and back to the state **Idle** on the detection of TDP-N. The relevant INAP operation, which can be the InitialDP operation or one of DP specific operations for TDP-N, shall be sent with a TC-BEGIN request primitive and the dialogue is locally ended by means of TC-END request primitive with pre-arranged end.

For all other operations sent from the SSF-FSM, the dialogue shall be maintained except for the following cases.

When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is one or more pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with component(s). When the SSF sends the last EventReportBCSM, ApplyChargingReport or CallInformationReport, the dialogue may be ended from the SSF by a TC-END request primitive with basic end.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component or pre-arranged end. When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components, or the dialogue is locally ended by means of a TC-END request primitive with pre-arranged end.

The SSF can end a dialogue with a TC-END request primitive with zero component or pre-arranged end depending on that TCAP dialogue is established or not, in the case call release is initiated by any other entity, then the SCF and the SSF has no pending call information requests (or pending requests which should be treated in the same way, see Note 1 of 3.1.1.5) nor any armed EDP to notify the SCF of the call release (for alternative way, see 3.4.2.2).

When the SSF has sent the last EventReportBCSM, ApplyChargingReport or CallInformationReport, the dialogue may be ended from the SCF by a TC-END request primitive with basic end.

3.4.1.1.2 Assisting/Handoff SSF-FSM related messages

A dialogue shall be established when the Assisting/Handoff SSF-FSM moves from the state **Idle** to the state **Waiting for Instructions**. The AssistRequestInstructions operation shall be transmitted with a TC-BEGIN request primitive.

For all other operations sent from the Assisting/Handoff SSF-FSM, the dialogue shall be maintained except for the following cases.

When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is one or more pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with component(s). When the SSF sends the last ApplyChargingReport, the dialogue may be ended from the SSF by a TC-END request primitive with basic end.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component or pre-arranged end. When the SSF-FSM makes a non-error case state transition to the state **Idle** and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components, or the dialogue is locally ended by means of a TC-END request primitive with pre-arranged end.

When the SSF has sent the last ApplyChargingReport, the dialogue may be ended from the SCF by a TC-END request primitive with basic end.

3.4.1.1.3 SSME-FSM related messages

The following procedures shall be followed:

- The dialogue shall be maintained when the ActivityTest Return Result is sent.
- No dialogue shall be established when the ServiceFilteringResponse operation is sent. The operation is sent with a TC-BEGIN request primitive and the dialogue is ended by means of a TC-END request primitive with pre-arranged end.
- A dialogue shall no longer be maintained when the Return Result of the ActivateServiceFiltering operation is sent. The dialogue is ended by means of a TC-END request primitive with basic end, the Return Result is transmitted with the same request.
- The dialogue is locally terminated by means of a TC-END request primitive with pre-arranged end, upon reception of a TC-BEGIN indication primitive with a CallGap operation.
- The dialogue shall be maintained when the RequestCurrentStatusReport, RequestEveryStatusChangeReport or RequestFirstStatusMatchReport operation is received inside the call context.

– The dialogue shall be maintained on sending following operations inside the call context if the operations are not final one:

  - Return Result of the RequestCurrentStatusReport operation;
  - Return Result of the RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation; and
  - StatusReport operation in reply to RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation.

– The dialogue shall no longer be maintained on sending following operations inside the call context if the operations are final one:

  - Return Result of the RequestCurrentStatusReport operation;
  - Return Result of the RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation; and
  - StatusReport operation in reply to RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation.

  The dialogue is ended from the SSF by means of TC-END request primitive with basic end, one of the above operations is transmitted with the same request.

  If monitor duration expires for RequestFirstStatusMatchReport or RequestEveryStatusChangeReport which has been received inside the call context and there is no need to maintain dialogue, the dialogue is ended from the SSF by means of a TC-END request primitive (basic) with zero component.

– The dialogue shall be established when the RequestCurrentStatusReport, RequestEveryStatusChangeReport or RequestFirstStatusMatchReport operation is received outside the call context.

– The dialogue shall no longer be maintained when the Return Result of the RequestCurrentStatusReport operation is sent outside the call context. The dialogue is ended from the SSF by means of a TC-END request primitive with basic end, the Return Result is transmitted with the same request.

– The dialogue shall be maintained when the Return Result of the RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation is sent outside the call context.

– The dialogue shall no longer be maintained when the StatusReport operation is sent in reply to the RequestFirstStatusMatchReport operation outside the call context. The dialogue is ended from the SSF by means of a TC-END request primitive with basic end, the StatusReport operation is transmitted with the same request.

– The dialogue shall be maintained when the StatusReport operation is sent in reply to the RequestEveryStatusChangeReport operation outside the call context.

– The dialogue shall no longer be maintained when the monitor duration expires for RequestFirstStatusMatchReport or RequestEveryStatusChangeReport which has been received outside the call context. The dialogue is ended from the SSF by means of a TC-END request primitive (basic) with zero component.

3.4.1.2 SCF-to-SSF messages

3.4.1.2.1 SCSM-FSM related messages

No dialogue shall be established when the SCSM-FSM moves from state **Idle** to state **Idle** upon receipt of InitialDP operation or one of DP specific operations for TDP-N. The operation is received with a TC-BEGIN indication primitive and the dialogue is locally terminated by means of a TC-END request primitive with pre-arranged end.

A dialogue shall be established when the SCSM-FSM moves from state **Idle** to state **Preparing SSF Instructions** upon the receipt of InitialDP operation for TDP-R, one of DP specific operations for TDP-R or AssistRequestInstructions operation.

A dialogue shall be established when the SCSM-FSM moves from state **Idle** to state **Preparing SSF Instructions** on sending an InitiateCallAttempt operation to the SSF.

For subsequent operations sent from the SCSM-FSM, the dialogue shall be maintained except for the following cases, i.e. all other operations are sent after a dialogue was established from the SSF (the SCF has previously received a TC-BEGIN indication primitive with an InitialDP operation, one of DP specific operations or an AssistRequestInstructions operation).

The dialogue shall no longer be maintained when the pre-arranged end condition is met in the SCF. When the SCF does not expect any messages other than possibly REJECT or ERROR messages for the operations sent and when the last associated operation timer expires, the dialogue is locally ended by means of a TC-END request primitive with pre-arranged end. Alternatively, the sending of operations, leading to the termination of the relationship, by means of a TC-END request primitive (basic end) is possible.

3.4.1.2.2 SCME-FSM related messages

The operations sent from the SCME-FSM shall be issued according to the following procedures:

- The dialogue shall be maintained when the ActivityTest operation is sent.
- A dialogue shall not be established when a CallGap operation is sent without using a SCSM associated dialogue. The operation is sent using a TC-BEGIN request primitive and the dialogue is terminated with a pre-arranged end.
- For sending one or more CallGap operations, the SCME-FSM may use an existing SCSM-FSM associated dialogue which was initiated by a SSF-FSM (i.e. established for the transmission of the InitialDP operation or one of DP specific operations). The dialogue shall be maintained and the CallGap operation(s) shall be sent with the first response of the SCSM-FSM to the InitialDP operation or one of DP specific operations.
- A dialogue shall be established when an ActivateServiceFiltering operation is sent. The operation shall be transmitted with a TC-BEGIN request primitive.
- The dialogue is locally terminated upon reception of a ServiceFilteringResponse operation using a TC-END request primitive with pre-arranged end.
- The dialogue shall be maintained when the RequestCurrentStatusReport, RequestEveryStatusChangeReport or RequestFirstStatusMatchReport operation is sent inside the call context.
- The dialogue shall be maintained on receiving following operations inside the call context if the operations are not the final ones:
  - Return Result of the RequestCurrentStatusReport operation;
  - Return Result of the RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation; and
  - StatusReport operation in reply to RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation.
- The dialogue shall no longer be maintained on receiving following operations inside the call context if the operations are the final ones:
  - Return Result of the RequestCurrentStatusReport operation;
  - Return Result of the RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation; and
  - StatusReport operation in reply to RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation.

  The dialogue is ended from the SSF by means of TC-END indication primitive with basic end, one of above operations is transmitted with the same request.

  If monitor duration expires for RequestFirstStatusMatchReport or RequestEveryStatusChangeReport which has been received inside the call context and there is no needs to maintain dialogue, the dialogue is ended from the SSF by means of a TC-END indication primitive (basic) with zero component.

– The dialogue shall be established when the RequestCurrentStatusReport, RequestEveryStatusChangeReport or RequestFirstStatusMatchReport operation is sent outside the call context.

– The dialogue shall no longer be maintained when the Return Result of the RequestCurrentStatusReport operation is received outside the call context. The dialogue is ended from the SSF by means of a TC-END indication primitive with basic end, the Return Result is transmitted with the same indication.

– The dialogue shall be maintained when the Return Result of the RequestFirstStatusMatchReport or RequestEveryStatusChangeReport operation is received outside the call context.

– The dialogue shall no longer be maintained when the StatusReport operation is received in reply to the RequestFirstStatusMatchReport operation outside the call context. The dialogue is ended from the SSF by means of a TC-END indication primitive with basic end, the StatusReport operation is received with the same indication.

– The dialogue shall be maintained when the StatusReport operation is received in reply to the RequestEveryStatusChangeReport operation outside the call context.

– The dialogue shall no longer be maintained when the monitor duration expires for RequestFirstStatusMatchReport or RequestEveryStatusChangeReport outside the call context. The dialogue is ended from the SSF by means of a TC-END indication primitive (basic) with zero component.

3.4.1.3 SCF-to/from-SRF messages

A dialogue is established when the SRF sends an AssistRequestInstructions operation to the SCF. For all other operations sent to/from the SRF, the dialogue shall be maintained.

In the case that there is no pending operation and TCAP dialogue is established, TCAP dialogue can be terminated by TC-END primitive with zero component. When the SCSM makes a non-error case state transition to end user interaction and there is no operation to be sent, the dialogue is ended by means of a TC-END request primitive (basic) with zero components.

The dialogue shall no longer be maintained when sending the SRReport operation for PlayAnnoucement with disconnection from the SRF set to TRUE or Return Result of the PromptAndCollectUserInformation with disconnection from the SRF set to TRUE. The dialogue is ended by means of a TC-END request primitive with basic end, and one of the above operations is transmitted with the same request.

Regardless of whether pending operation exists or not, when the SRSM-FSM is informed of the disconnection of bearer connection (in the case of SCF initiated disconnection or call abandon from call party) and dialogue is established, the dialogue is ended by means of a TC-END request primitive (basic) with zero components or TC-END request primitive (pre-arranged end).

The dialogue shall no longer be maintained when the pre-arranged end condition is met in the SRF. When the SRSM-FSM is informed the disconnection of bearer connection and TCAP dialogue is not established, TCAP dialogue is locally terminated by TC-END primitive with pre-arranged end.

When the SCF does not expect any messages other than possibly REJECT or ERROR messages for the operations sent and when the last associated operation timer expires, the dialogue is locally ended by means of a TC-END request primitive with pre-arranged end. Alternatively, the sending of operations, leading to the termination of the relationship, by means of a TC-END request primitive (basic end) is possible.

In the relay case, the SRF-SCF relationship uses the SSF-SCF TCAP dialogue. This is possible, because begin and end of the SRF-SCF relationship are embedded in the SSF-SCF relationship. SRF-SCF information shall be exchanged with TC-CONTINUE request primitives.

In the case of the SSF relay, it is for further study how to map messages to ROSE capability of bearer signalling system between the SSF and the SRF, and what services are assumed from ROSE.

3.4.2 Abnormal procedures

This subclause describes the procedures and TCAP primitives that shall be used for reporting abnormal situations between SSF, SCF and SRF. The error cases are defined in 3.2.

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 or the SRF 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_{SSF} or T_{SRF}, 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 such as application time out).

In error situations pre-arranged 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 when a dialogue needs to be closed by the initiating entity before its establishment has been completed (before the first TC indication primitive to the TC-BEGIN request primitive has been received from the responding entity), the TC-user shall issue a TC-END request primitive with pre-arranged 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 Recommendation Q.774.

3.4.2.1 SCF-to-SSF/SRF messages

Considering that both SSF and SRF do not have the logic to recover from error cases detected on the SCF-SSF/SRF interface, the following shall apply:

- Operation errors and rejection of TCAP components shall be transmitted to the SSF and, respectively, the SRF with a TC-END request primitive, basic end.

If, in violation of the above procedure, an ERROR or REJECT component is received with a TC-CONTINUE indication primitive, the SSF and, respectively, the SRF shall abort the dialogue with a TC-U-ABORT request primitive.

In the case of the SSF relay, it is for further study how to map messages to ROSE capability of bearer signalling system between the SSF and the SRF, and what services are assumed from ROSE.

3.4.2.2 SSF/SRF-to-SCF messages

Operation errors and rejection of TCAP components shall be transmitted to the SCF according to the following rules:

- The dialogue shall be maintained when the preceding message, which contained the erroneous component, indicated that the dialogue shall be maintained, i.e. the error or reject shall be transmitted with a TC-CONTINUE request primitive if the erroneous component was received with a TC-CONTINUE indication primitive.
- On receipt of an ERROR or REJECT component the SCF decides on further processing. It may either continue, explicitly end or abort the dialogue.
- In all other situations the dialogue shall no longer be maintained, i.e. the error or reject shall be transmitted with a TC-END request primitive, basic end, if the erroneous component was received with a TC-BEGIN indication primitive.
- On expiration of application timer T_{SSF} or T_{SRF}, dialogue shall be terminated by means of a TC-U-ABORT primitive with an Abort reason, regardless of whether TCAP dialogue is established or not.

If the error processing in the SSF-SRF leads to the case where the SSF-SRF is not able to process further SCF operations while the dialogue is to be maintained, the SSF-SRF aborts the dialogue with a TC-END request primitive with basic end or a TC-U-ABORT request primitive, depending on whether any pending ERROR or REJECT component is to be sent or not.

The SSF can end a dialogue with a TC-U-ABORT request primitive in case call release is initiated by any other entity, then the SCF and the SSF has no pending call information requests (or pending requests which should be treated in the same way, i.e. ApplyCharging nor any armed EDP to notify the SCF of the call release (for alternative way, see 3.4.1.1.1).

In the case of the SSF relay, it is for further study how to map messages to ROSE capability of bearer signalling system between the SSF and the SRF, and what services are assumed from ROSE.

3.4.3 Dialogue establishment

The establishment of an INAP dialogue involves two application processes as described in 0.4, 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.

3.4.3.1 Sending of a TC-BEGIN request primitive

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. See 3.4.8 for a description of the invocation procedure.

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 "pre-arranged release".

If no TC indication primitive is expected because no dialogue is to be established according to the rules as stated in 3.4.1 and 3.4.2, 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 "pre-arranged release".

3.4.3.2 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 as described in 3.4.8.
- If no dialogue is to be established according to the rules as stated in 3.4.1 and 3.4.2, SACF will wait for the last indication primitive from TC and issue a TC-END request with the release method parameter set to "pre-arranged release".
- If the application-context-name included in the primitive is not supported, issue a TC-U-ABORT request primitive. If an alternative application-context can be offered its name is included in the TC-U-ABORT request primitive.

It is for further study whether or not the application-context-negotiation is limited only for using the TC-U ABORT primitive.

3.4.3.3 Receipt of the first TC-CONTINUE ind

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 as described in 3.4.8, 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.

3.4.3.4 Receipt of a TC-END ind

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 matches the one used in the TC-BEGIN request primitive, then the SACF shall process the following TC component handling indication primitives as described in 3.4.8.

3.4.3.5 Receipt of a TC-U-ABORT ind

Receipt of a TC-U-ABORT indication primitive is described as part of user abort procedure (see 3.4.6.2). 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.

3.4.3.6 Receipt of a TC-P-ABORT ind

Receipt of a TC-P-ABORT indication primitive is described as part of provider abort procedure (see 3.4.7.1).

3.4.4 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.

3.4.4.1 Sending entity

SACF shall process any component handling request primitives as described in 3.4.8.

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

3.4.4.2 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 as described in 3.4.8.

3.4.5 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 3.4.1 and 3.4.2.

The dialogue termination procedure is driven by the following events:

- A TC-END request primitive.
- A TC-END indication primitive.

3.4.5.1 Sending of TC-END request

When the dialogue shall no longer be maintained, SACF shall process any component handling request primitives as described in 3.4.8

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 "pre-arranged release", according to the rules as stated in 3.4.1 and 3.4.2.

When no dialogue is to be established, refer to 3.4.3.1 and 3.4.3.2.

3.4.5.2 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 as described in 3.4.8.

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

3.4.6 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.

3.4.6.1 Sending of TC-U-ABORT request

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

3.4.6.2 Receipt of a TC-U-ABORT indication

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

3.4.7 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.

3.4.7.1 Receipt of a TC-P-ABORT indication

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

3.4.8 Procedures for INAP operations

This subclause describes the procedures for INAP operations.

3.4.8.1 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.

3.4.8.2 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, requests the transfer of the reject component using the TC-U-REJECT request primitive with the problem code (Initiating Release).
- If sufficient INAP related resources are not available to perform the requested operation, request the transfer of a reject component using the TC-U-REJECT request primitive with the problem code (Resource Limitation).
- Otherwise, accept the TC-INVOKE indication primitive. If the operation is to be user confirmed, SACF waits for the corresponding response.

3.4.8.3 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.

3.4.8.4 Receipt of a response

3.4.8.4.1 Receipt of TC-RESULT-NL indication

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).

3.4.8.4.2 Receipt of TC-RESULT-L indication

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.

3.4.8.4.3 Receipt of TC-U-ERROR indication

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.

3.4.8.4.4 Receipt of TC-U-REJECT indication

On receipt of a TC-U-REJECT indication primitive which affects a pending operation, SACF shall accept the TC-U-REJECT indication primitive.

3.4.8.4.5 Receipt of a TC-L-REJECT indication

This event occurs when the local TC detects a protocol error in an incoming component which affects an operation.

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.

Note that when the problem code indicates a general problem, it is considered that the event cannot be related to an active operation even if the invoke Id is provided by TC. This is because it is unclear whether the invoke Id refers to a local or remote invocation. The behaviour of SACF in such a case is described in 3.4.8.5.3.

3.4.8.4.6 Receipt of a TC-L-CANCEL indication

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.

3.4.8.5 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.

3.4.8.5.1 Receipt of a TC-U-REJECT indication

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 3.4.2. This is also applicable for invoke problems related to a class 4 linked operation.

3.4.8.5.2 Receipt of a TC-R-REJECT indication

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 3.4.2.

3.4.8.5.3 Receipt of a TC-L-REJECT indication

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.

3.4.8.5.4 Receipt of a TC-NOTICE indication

This informs the SACF that a message cannot be delivered by the Network Layer, this can only occur if the Return Option has been set (see 3.4.9.1.8). It is for the application process to decide whether to terminate the dialogue or retry.

3.4.9 Mapping on to TC services

3.4.9.1 Dialogue control

The TC-UNI service is not used by INAP.

3.4.9.1.1 Destination address

This parameter is set by the dialogue initiating application process, and may optionally be modified by the responding dialogue in the first backward TC-CONTINUE, but will not be seen by initiating user.

3.4.9.1.2 Originating address

This parameter is set by the dialogue initiating application process.

3.4.9.1.3 Dialogue Id

The value of this parameter is associated with the INAP invocation in an implementation dependent manner.

3.4.9.1.4 Application-context-name

The application-context-name parameter is set by SACF as defined in 2.1.5 and 2.2.2.5.3.

3.4.9.1.5 User information

This parameter may be used by both initiating and responding application process.

3.4.9.1.6 Component present

This parameter is used by SACF as described in Recommendation Q.771.

3.4.9.1.7 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 3.4.1 and 3.4.2.

3.4.9.1.8 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.

3.4.9.2 Operation procedures

3.4.9.2.1 Invoke Id

This parameter is set by the sending application process.

3.4.9.2.2 Linked Id

This parameter is set by the sending application process.

3.4.9.2.3 Dialogue Id

The value of this parameter is associated with the INAP invocation in an implementation dependent manner.

3.4.9.2.4 Class

The value of this parameter is set by SACF according to the type of the operation to be invoked according to 2.1.

3.4.9.2.5 Operation

The operation code of a TC-INVOKE request primitive is set by the sending application process as defined in 2.4

SACF shall set the operation code of the TC-RESULT-L primitive (if required) to the same value as the one received at invocation time.

3.4.9.2.6 Error

The error parameter of the TC-U-ERROR request primitive is set by the sending application process as defined in 2.4.

3.4.9.2.7 Parameters

The argument parameter of TC-INVOKE primitives is set by the sending application process as defined in 2.

The result parameter of TC-RESULT-L primitives is set by the sending application process as defined in 2.

The parameter of TC-U-ERROR primitives are set by the sending application process as defined in 2.

3.4.9.2.8 Time out

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

3.4.9.2.9 Last component

This parameter is used by SACF as described in Recommendation Q.771.

3.4.9.2.10 Problem code

This parameter is used by SACF as described in 3.4.8.

3.4.9.2.11 Abort reason

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

Annex A

INAP SDL Diagrams

(This annex forms an integral part of this Recommendation)

A.1 Introduction

This annex contains SDL diagrams for the FSMs for the SSF, assisting/handoff SSF, SCF and SRF, as described in the text and Figures of 3.1.1.5, 3.1.1.6, 3.1.2 and 3.1.3, together with information from relevant detailed operation procedures. SDL diagrams for the SDF-related states of the SCF and for the SDF itself are contained in Annex B.

Notes on the diagrams give further details where necessary. The SDL diagrams are at the same level of detail as the relevant text and figures, and do not contain anything not in the main body of this Recommendation unless specifically noted.

The SDL used is compliant with the latest Recommendations for SDL [Recommendation Z.100 (1993)], but no symbols are used which are not contained in Recommendation Z.100 (1988).

In the case of inconsistency between SDL diagrams and text, text shall be taken as correct.

A.2 SDL diagrams

The SDL in these diagrams is very high-level. It is incomplete in that data declarations are given only for timers and signals, but data carried by the signals is not declared.

Names used for operations are exactly those of this Recommendation, and other locally-defined names in general relate to event names used in this Recommendation. In some cases, however, new names have been introduced for signals where this Recommendation does not provide a name. Where operation names have been split to contain them within a symbol, the SDL technique of breaking lines with an underline followed by a space is used; this combination is ignored in SDL so A_B is the same name as AB.

In general receipt of a signal (input) from an FE outside the FE being modelled will require some service logic processing, but the detail of such processing is not of interest to the FSM (only its results). The existence of this service logic processing has been shown by inclusion of a general procedure called Process_Input. In other places SDL tasks are used to indicate the scope of processing required.

It should be noted that in SDL the word “reset” applied to timers means to stop the timer, not restart it. “Set” in SDL means both start and restart a timer.

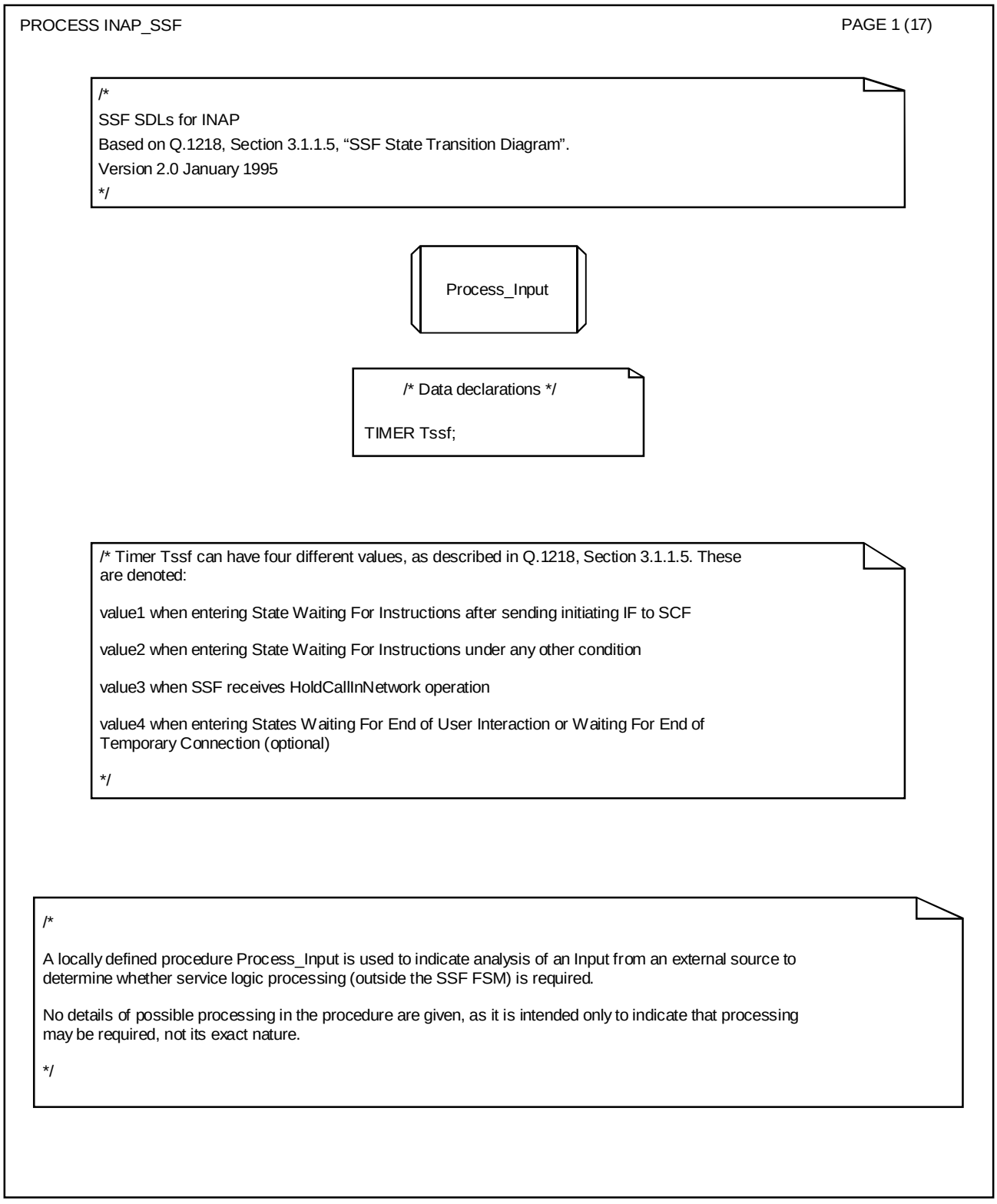


FIGURE A.1/Q.1218 (sheet 1 of 17)

SDL for SSF-FSM

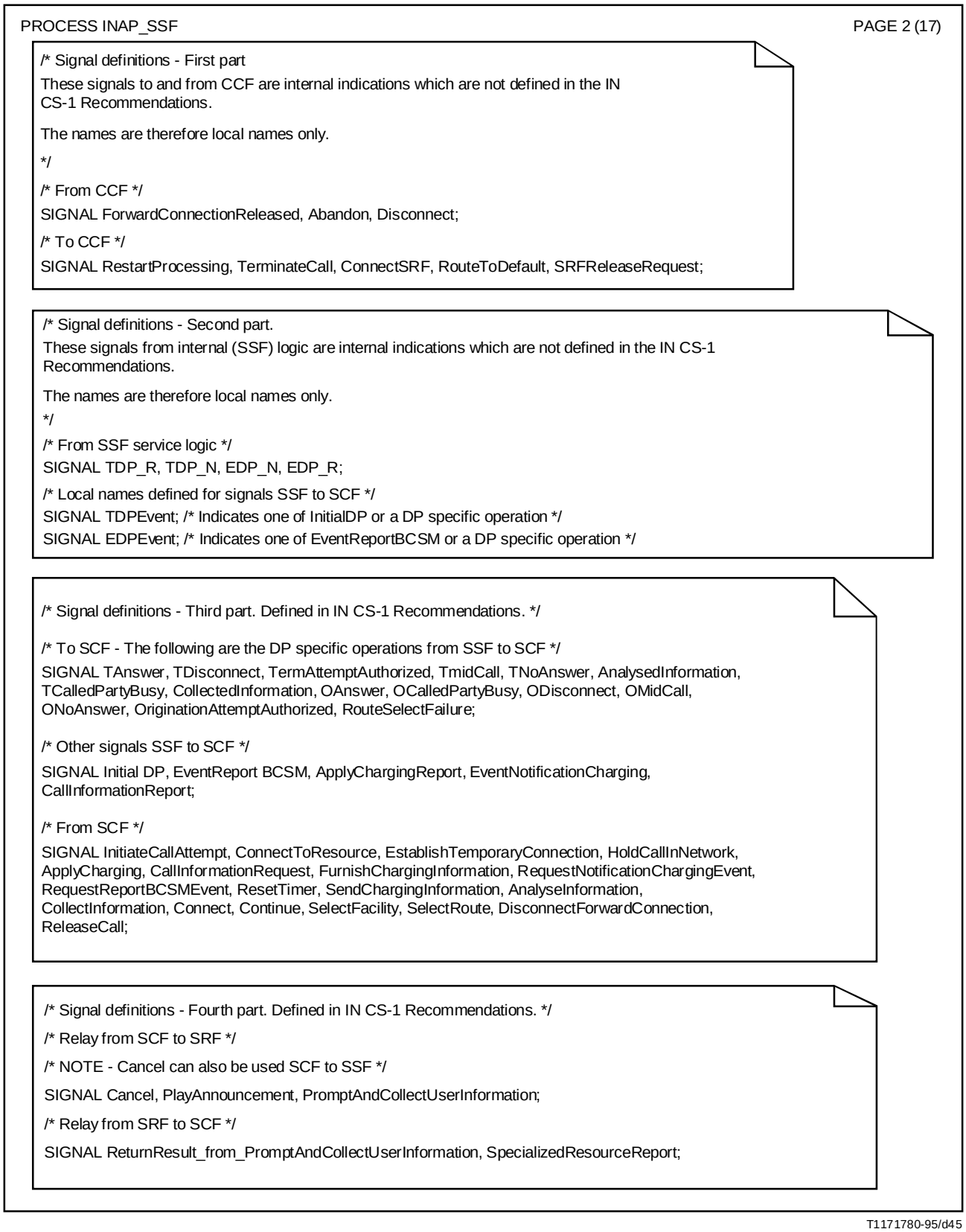


FIGURE A.1/Q.1218 (sheet 2 of 17)

SDL for SSF-FSM

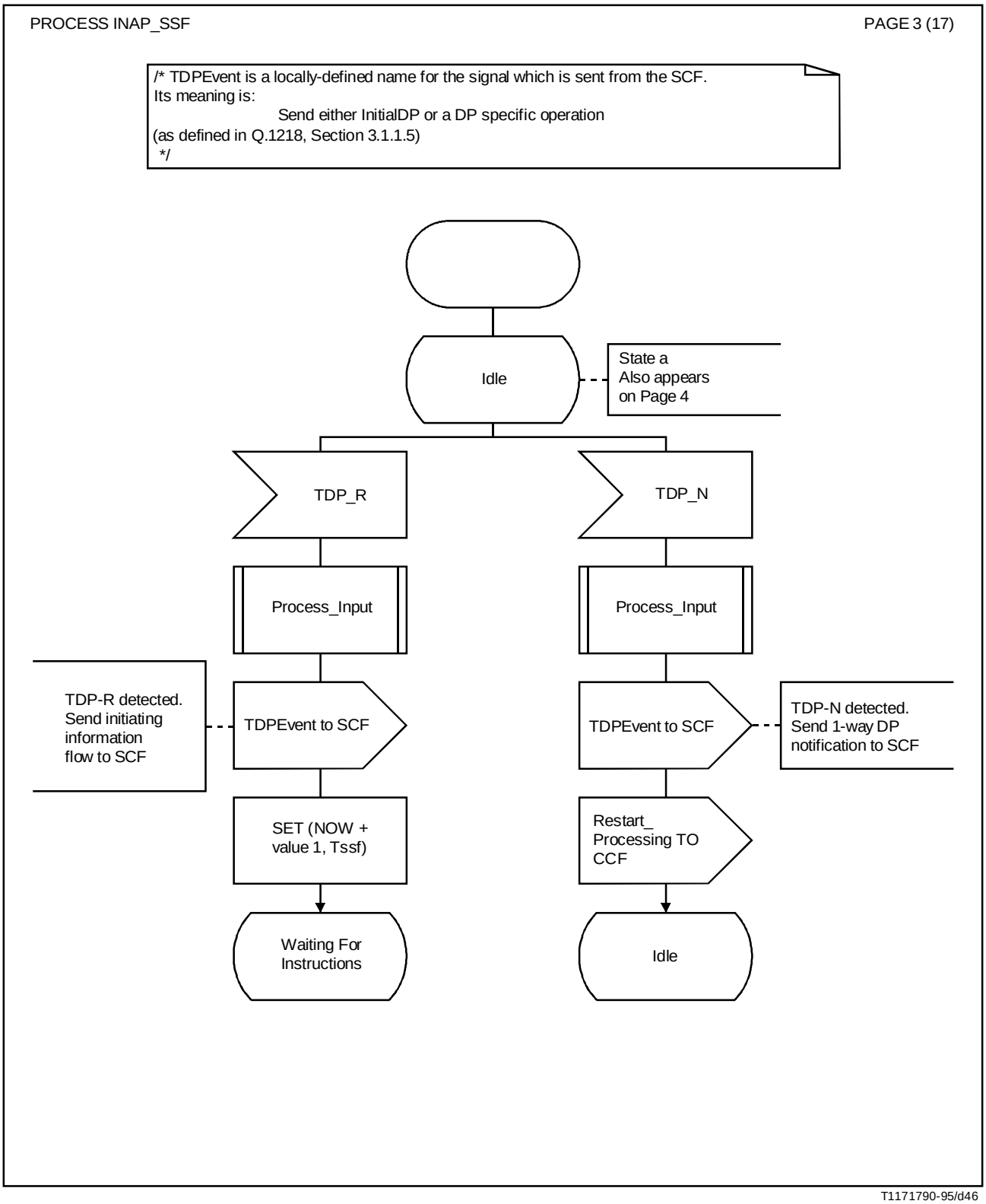


FIGURE A.1/Q.1218 (sheet 3 of 17)

SDL for SSF-FSM

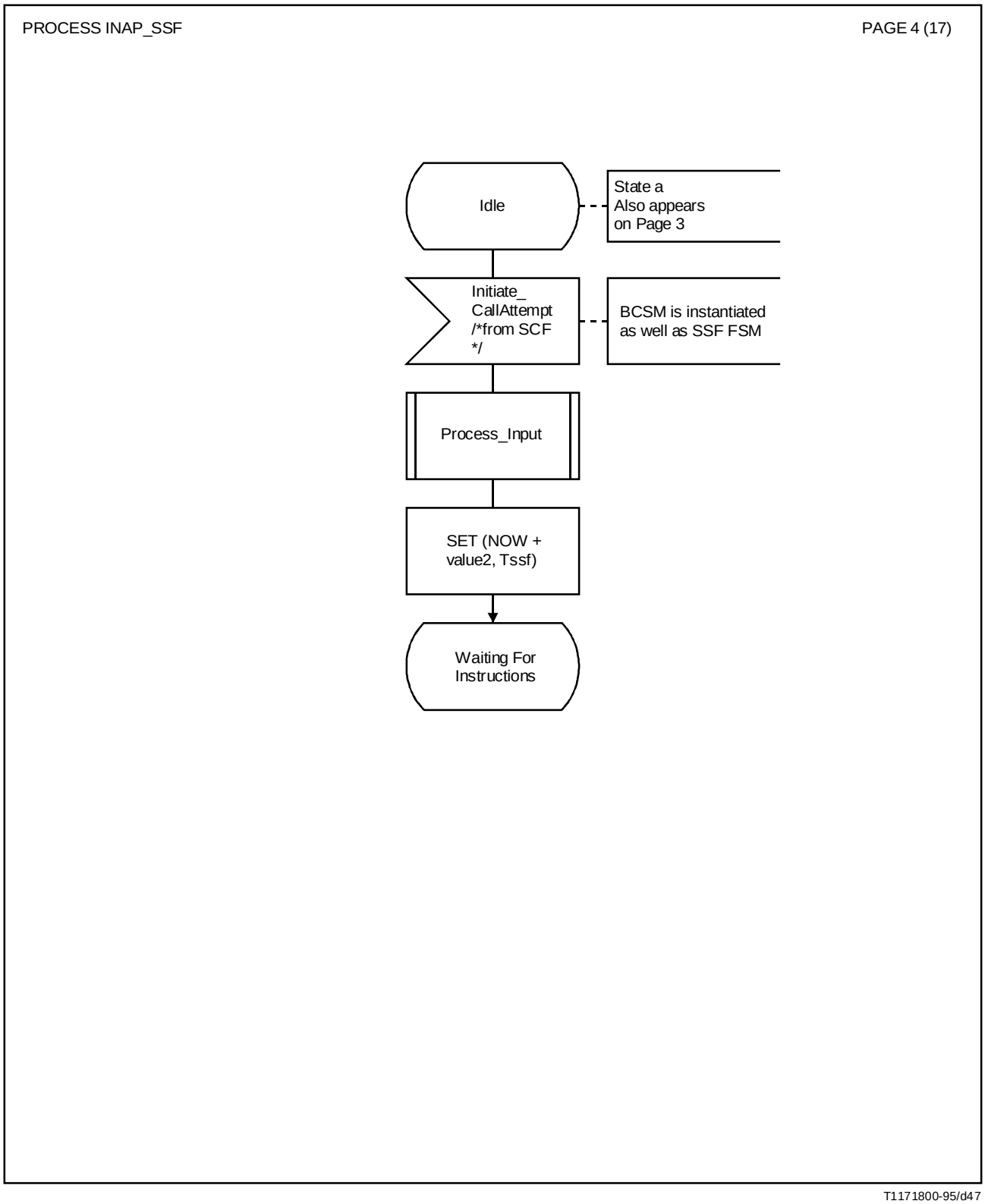


FIGURE A.1/Q.1218 (sheet 4 of 17)

SDL for SSF-FSM

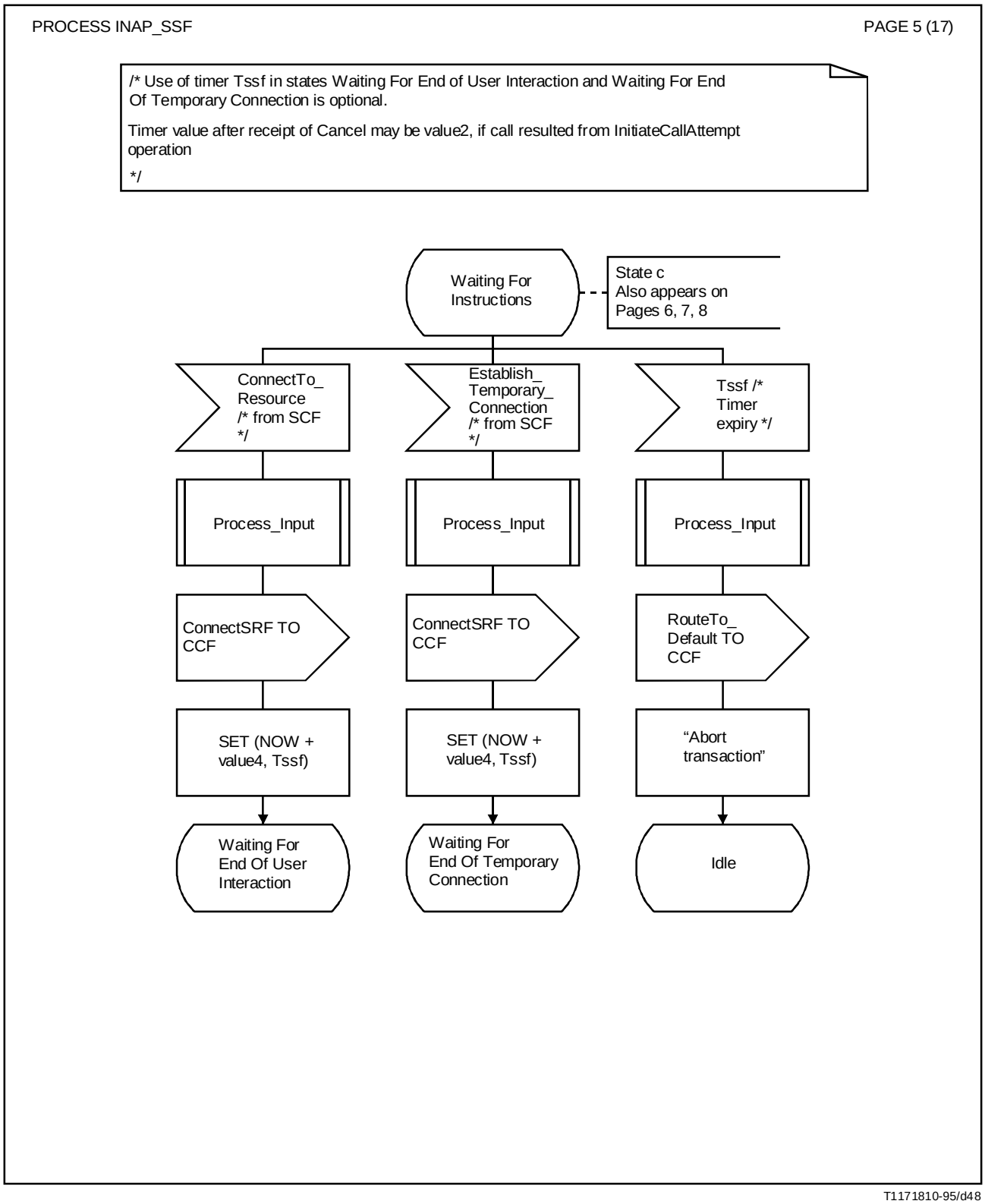


FIGURE A.1/Q.1218 (sheet 5 of 17)

SDL for SSF-FSM

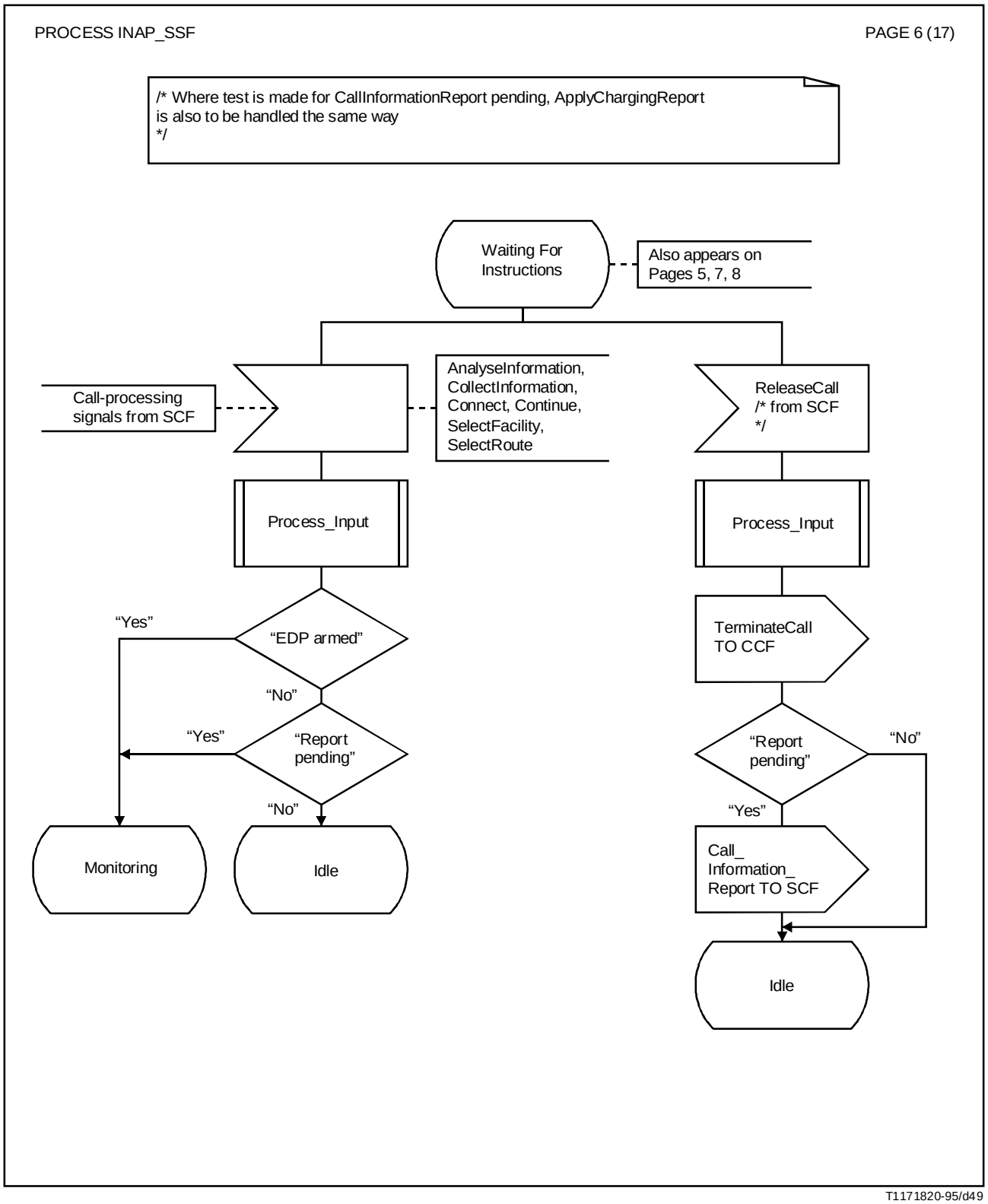


FIGURE A.1/Q.1218 (sheet 6 of 17)

SDL for SSF-FSM

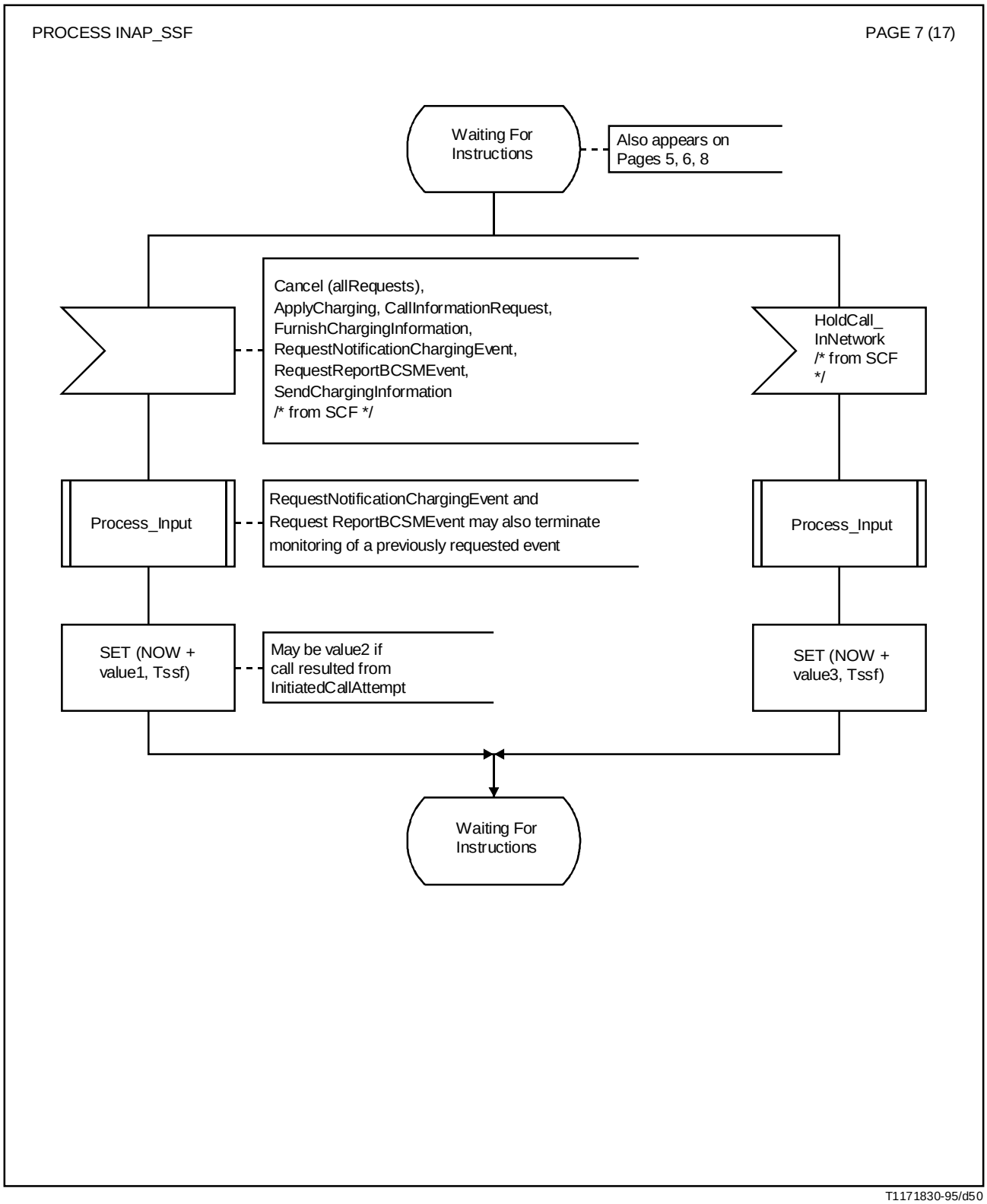


FIGURE A.1/Q.1218 (sheet 7 of 17)

SDL for SSF-FSM

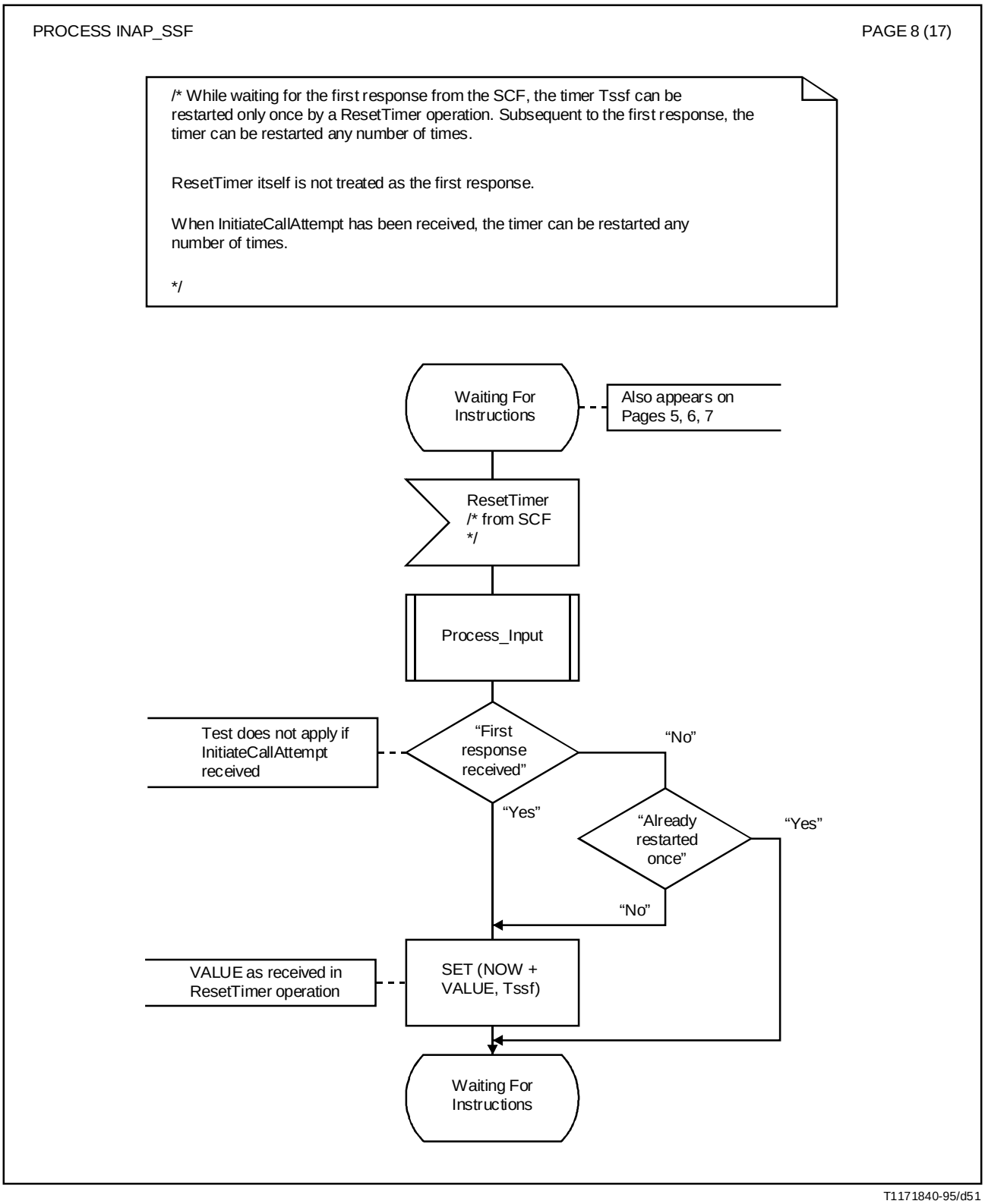


FIGURE A.1/Q.1218 (sheet 8 of 17)

SDL for SSF-FSM

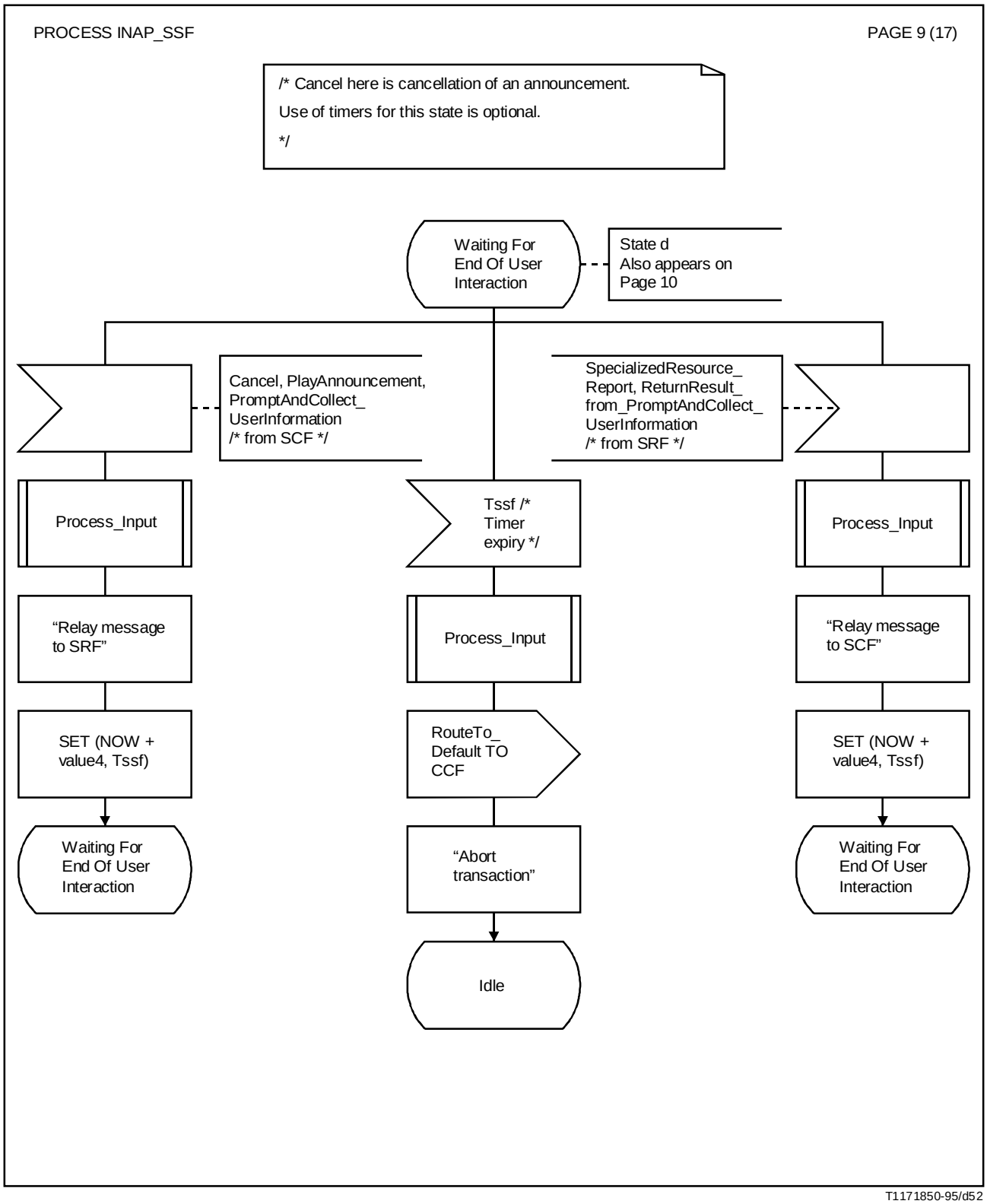


FIGURE A.1/Q.1218 (sheet 9 of 17)

SDL for SSF-FSM

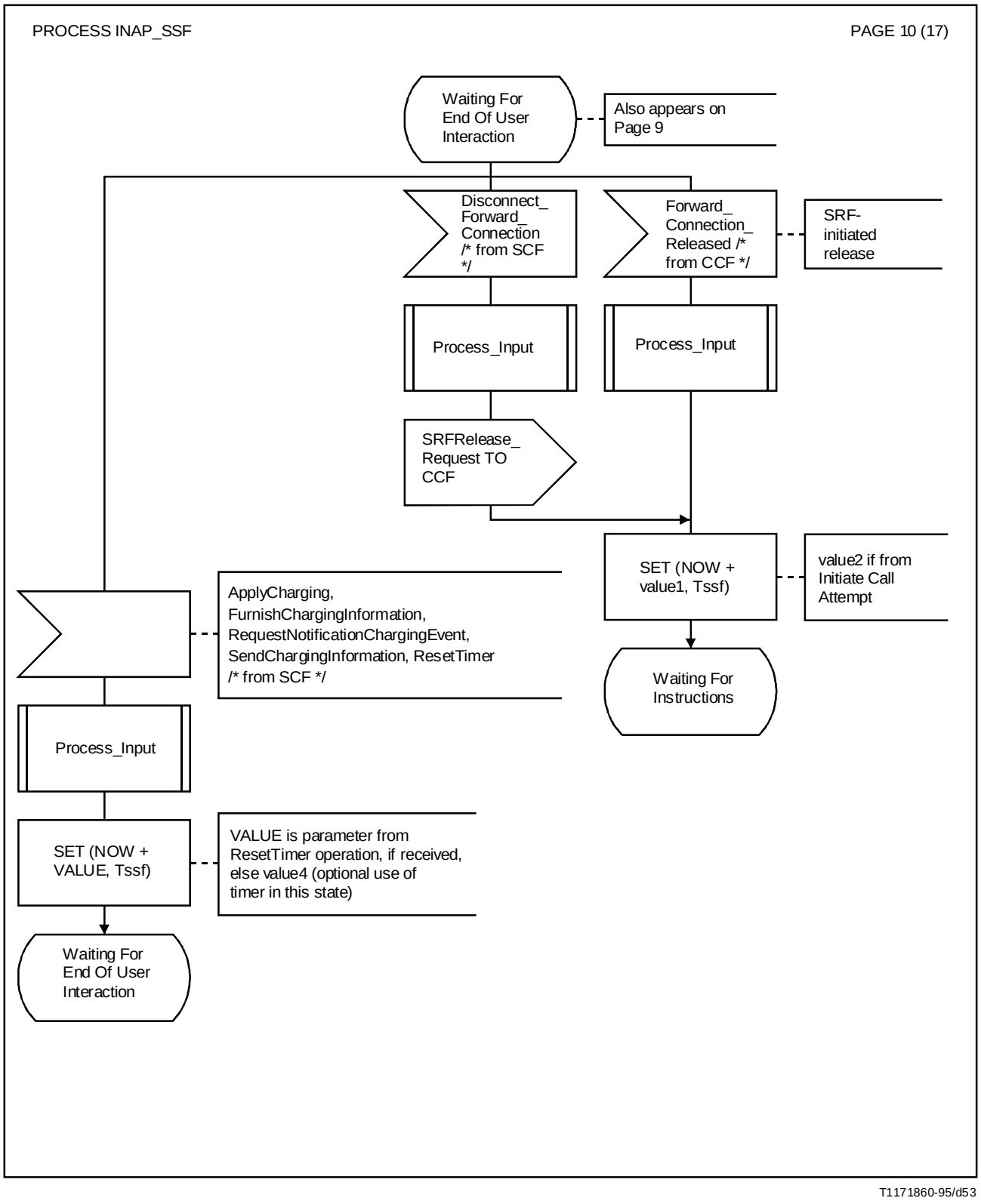


FIGURE A.1/Q.1218 (sheet 10 of 17)

SDL for SSF-FSM

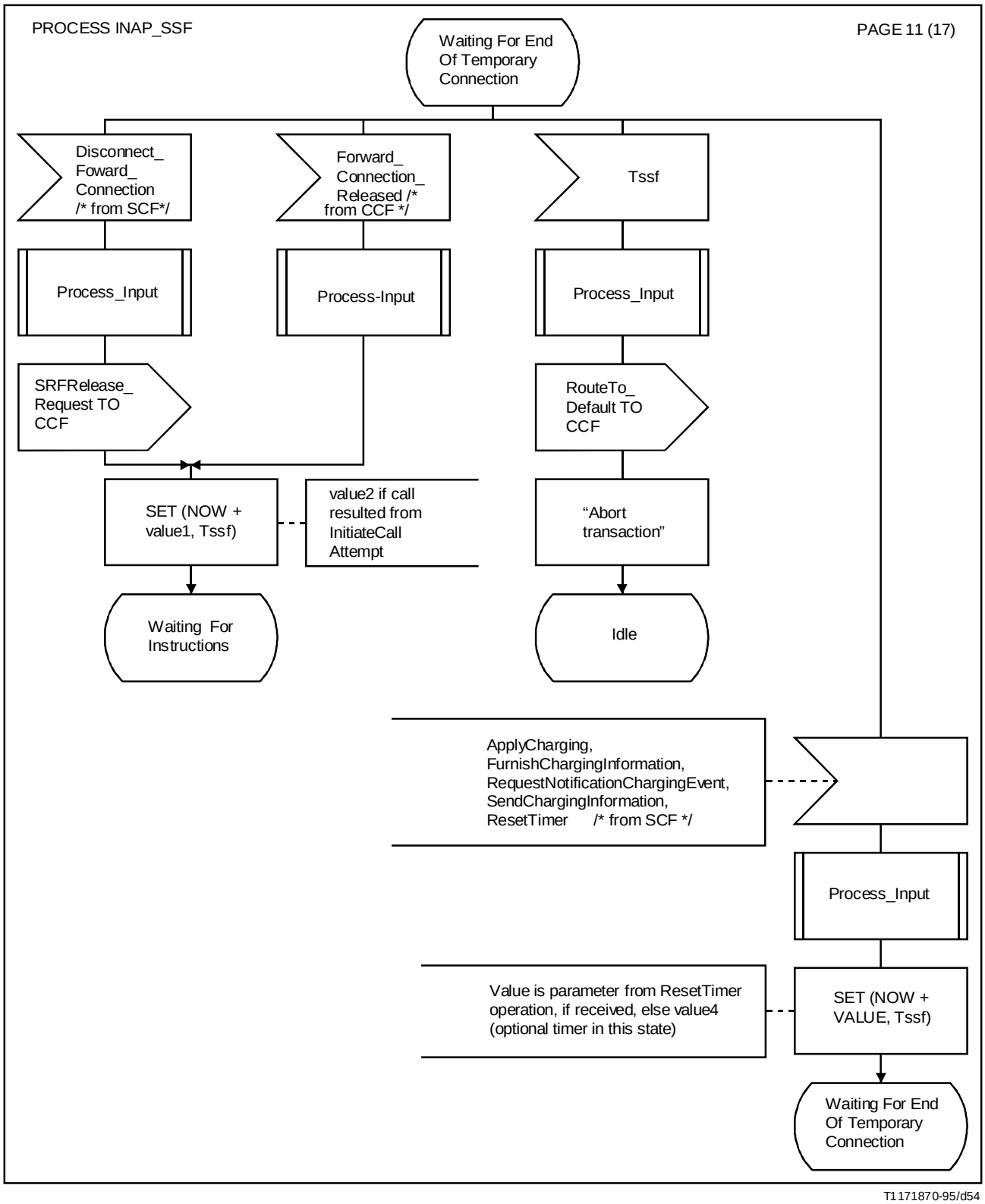


FIGURE A.1/Q.1218 (sheet 11 of 17)

SDL for SSF-FSM

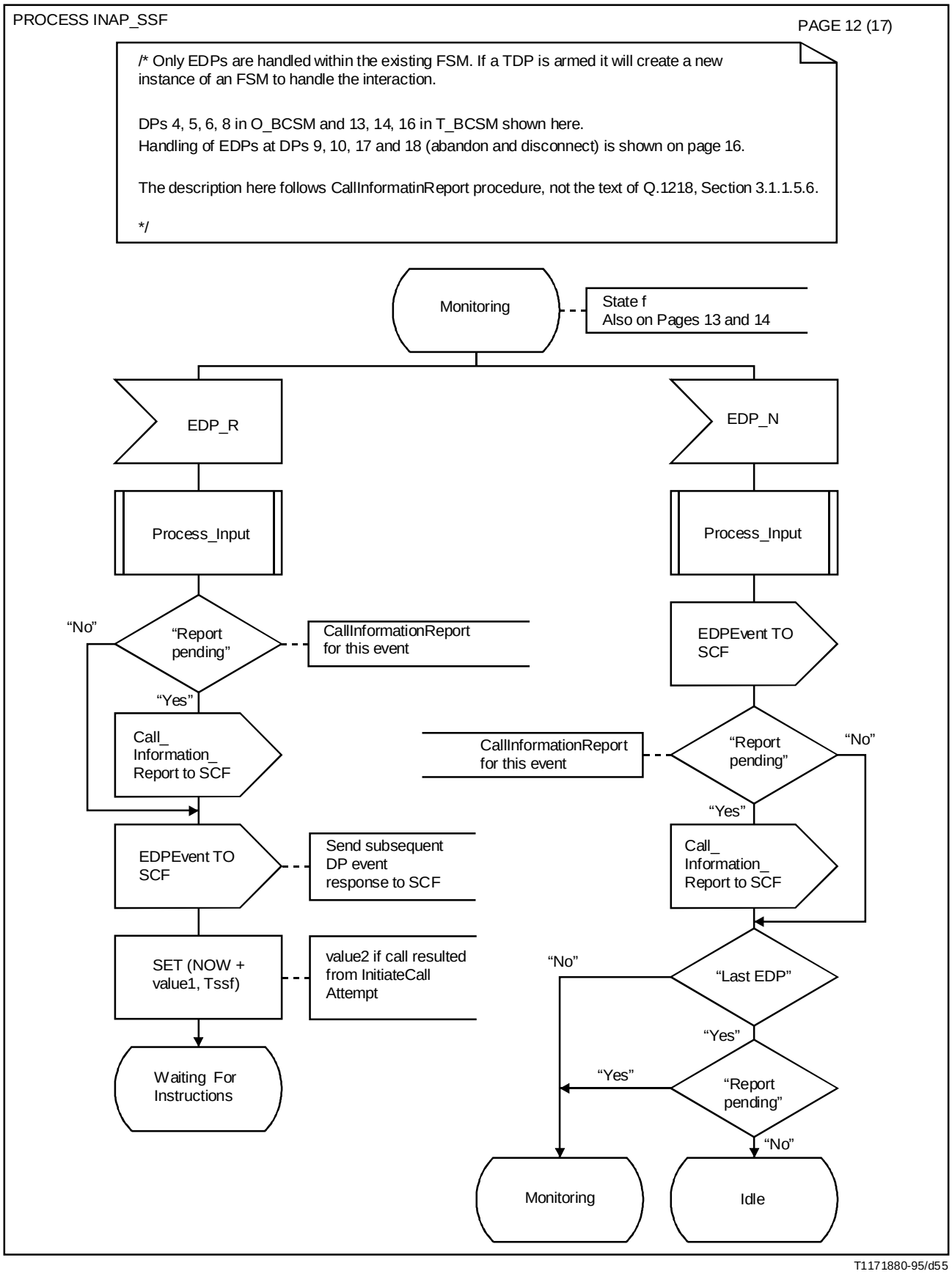


FIGURE A.1/Q.1218 (sheet 12 of 17)

SDL for SSF-FSM

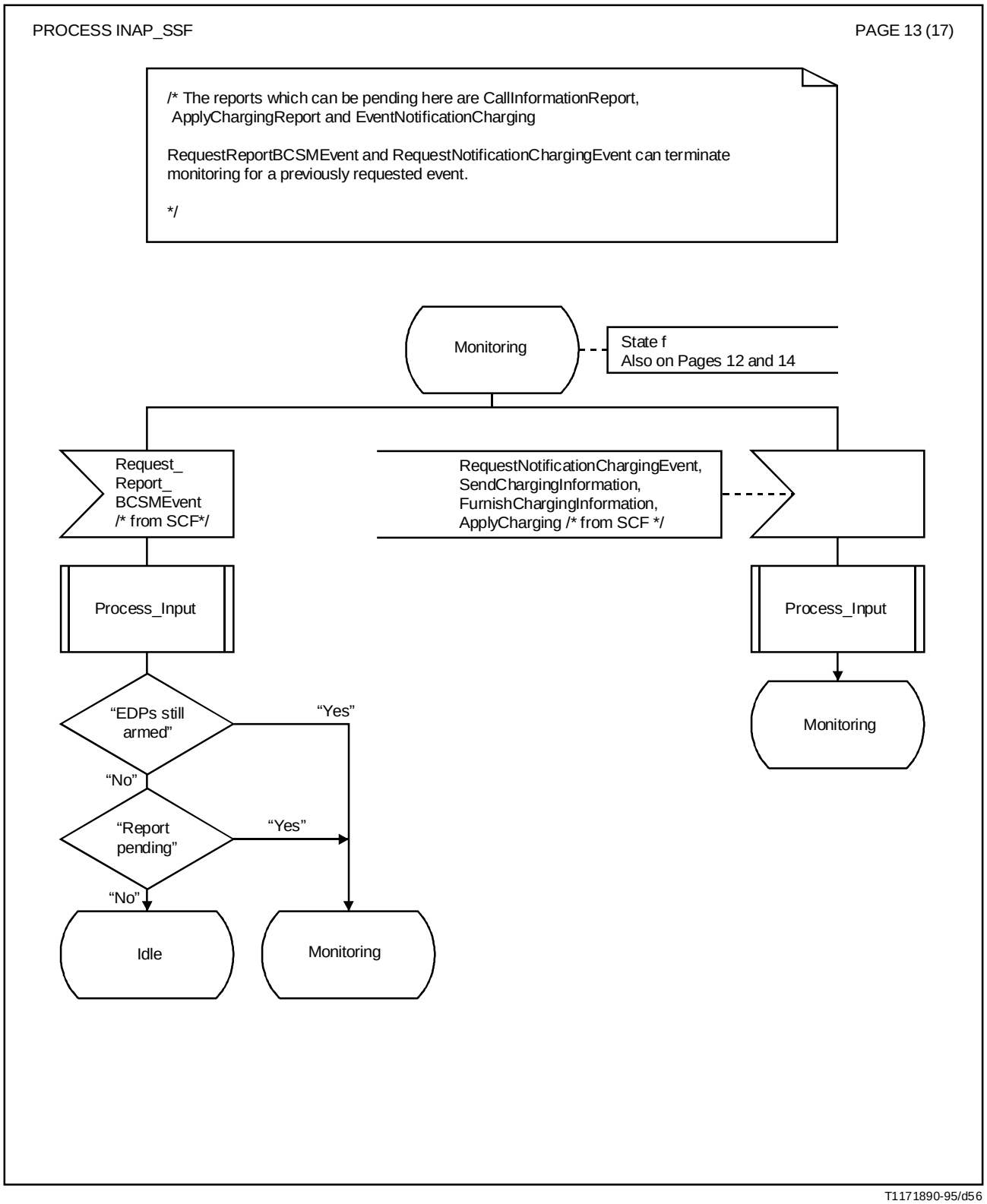
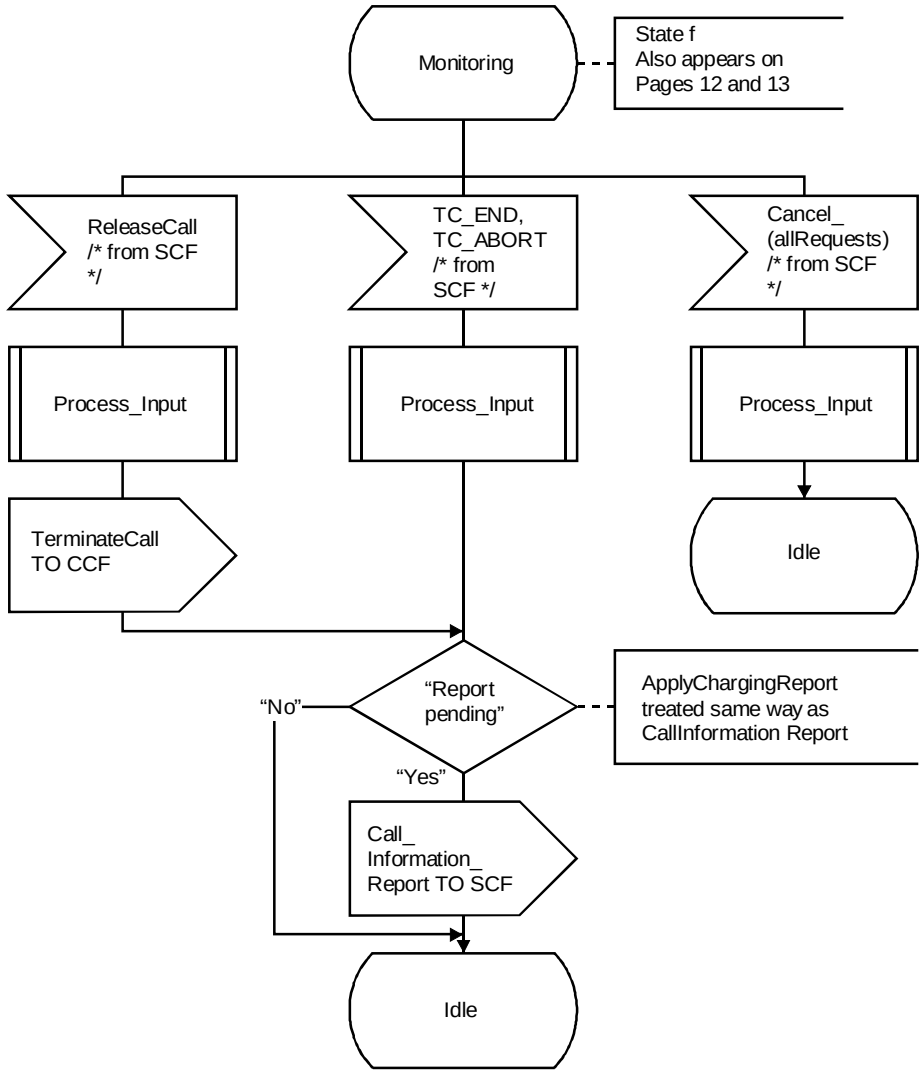


FIGURE A.1/Q.1218 (sheet 13 of 17)

SDL for SSF-FSM



T1171900-95/d57

FIGURE A.1/Q.1218 (sheet 14 of 17)

SDL for SSF-FSM

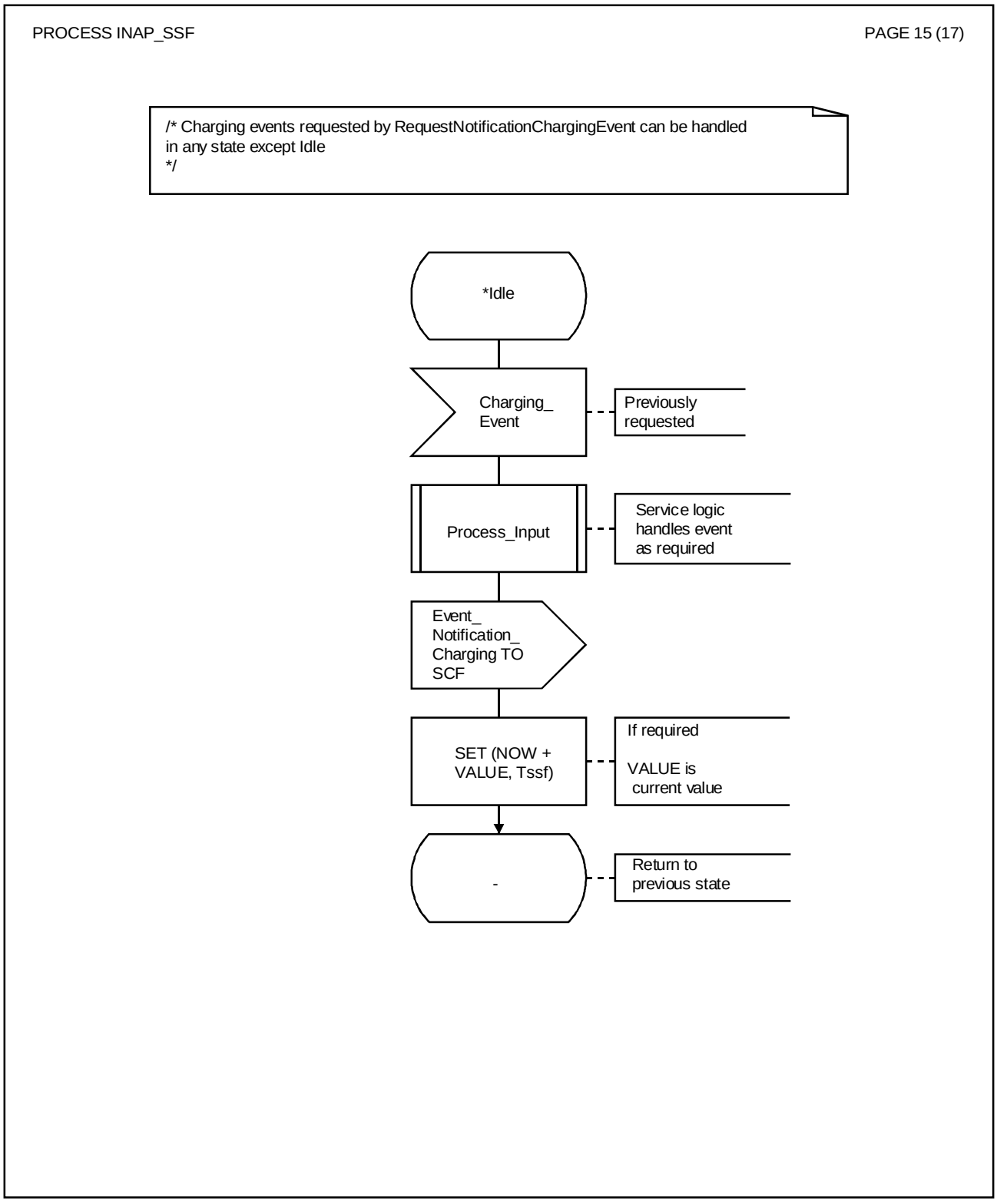


FIGURE A.1/Q.1218 (sheet 15 of 17)

SDL for SSF-FSM

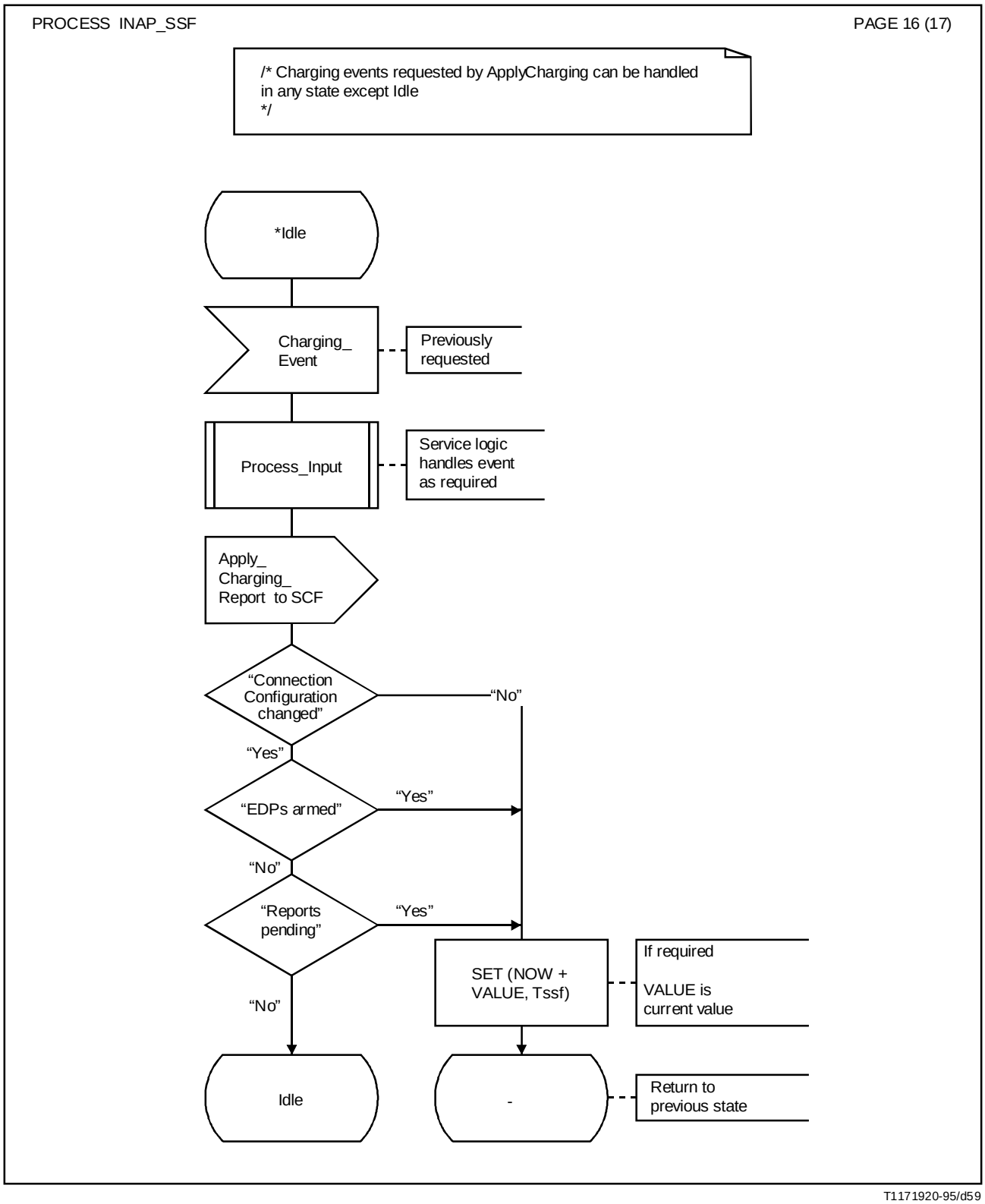


FIGURE A.1/Q.1218 (sheet 16 of 17)

SDL for SSF-FSM

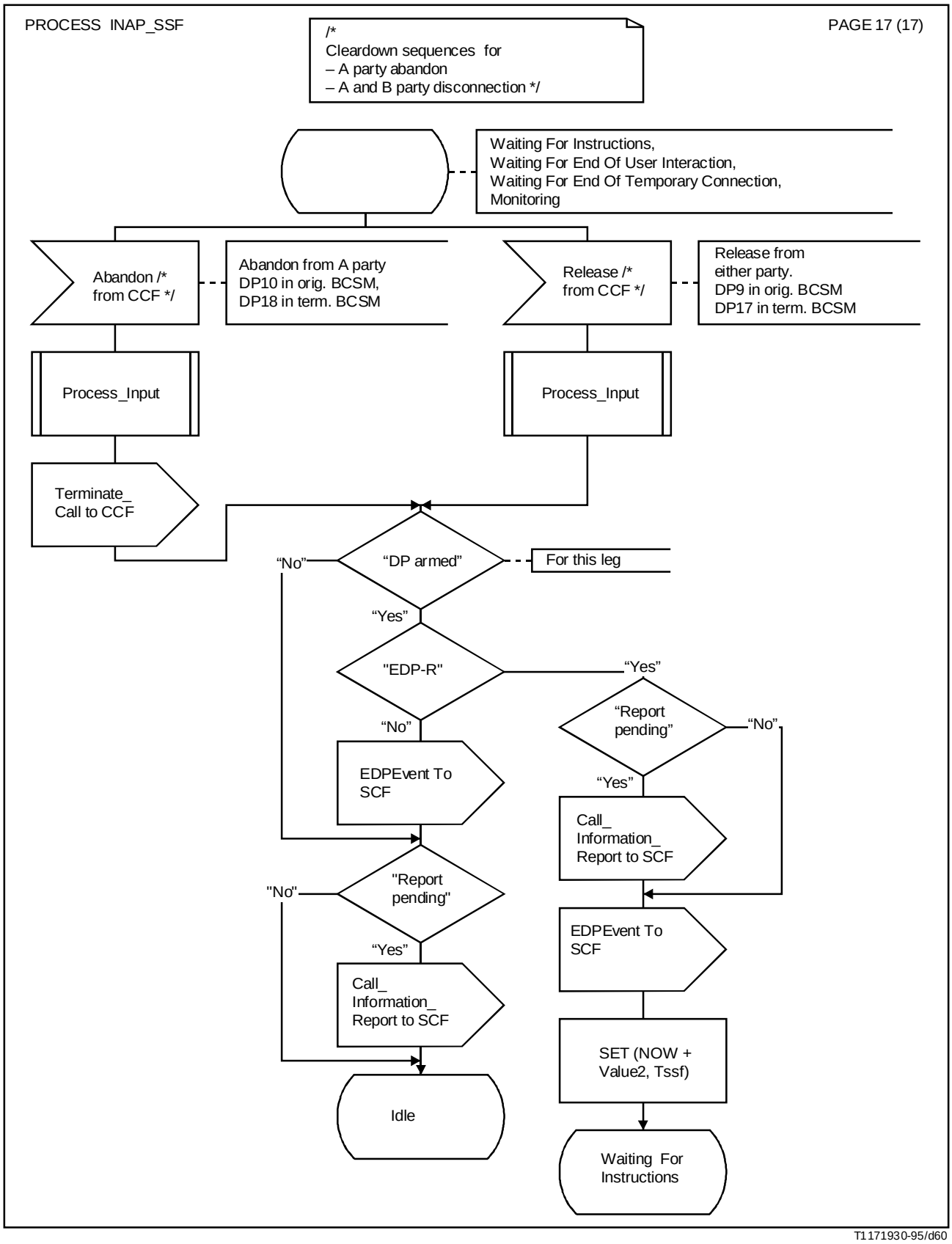


FIGURE A.1/Q.1218 (sheet 17 of 17)

SDL for SSF-FSM

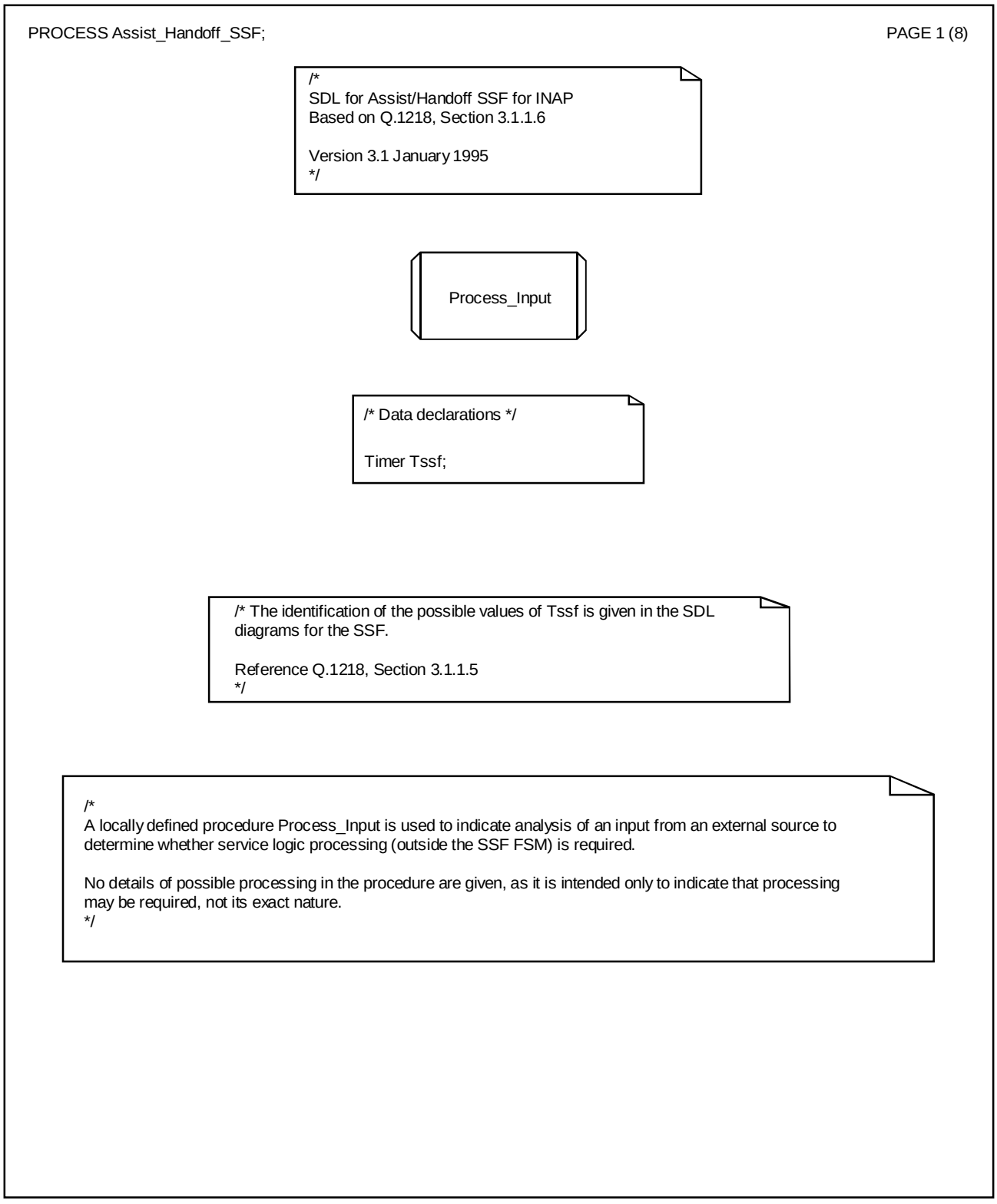


FIGURE A.2/Q.1218 (sheet 1 of 8)

SDL for assist/handoff SSF-FSM

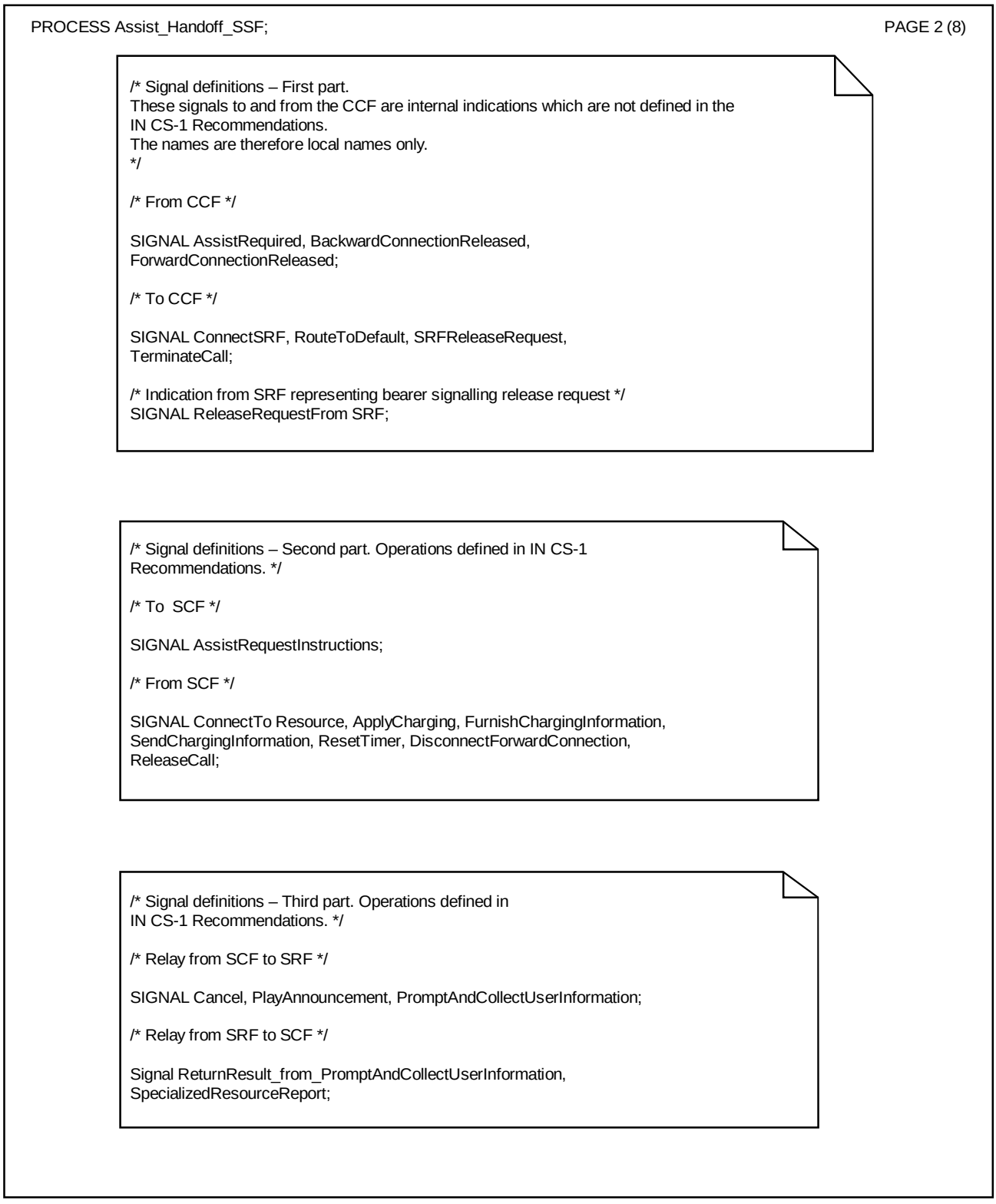


FIGURE A.2/Q.1218 (sheet 2 of 8)

SDL for assist/handoff SSF-FSM

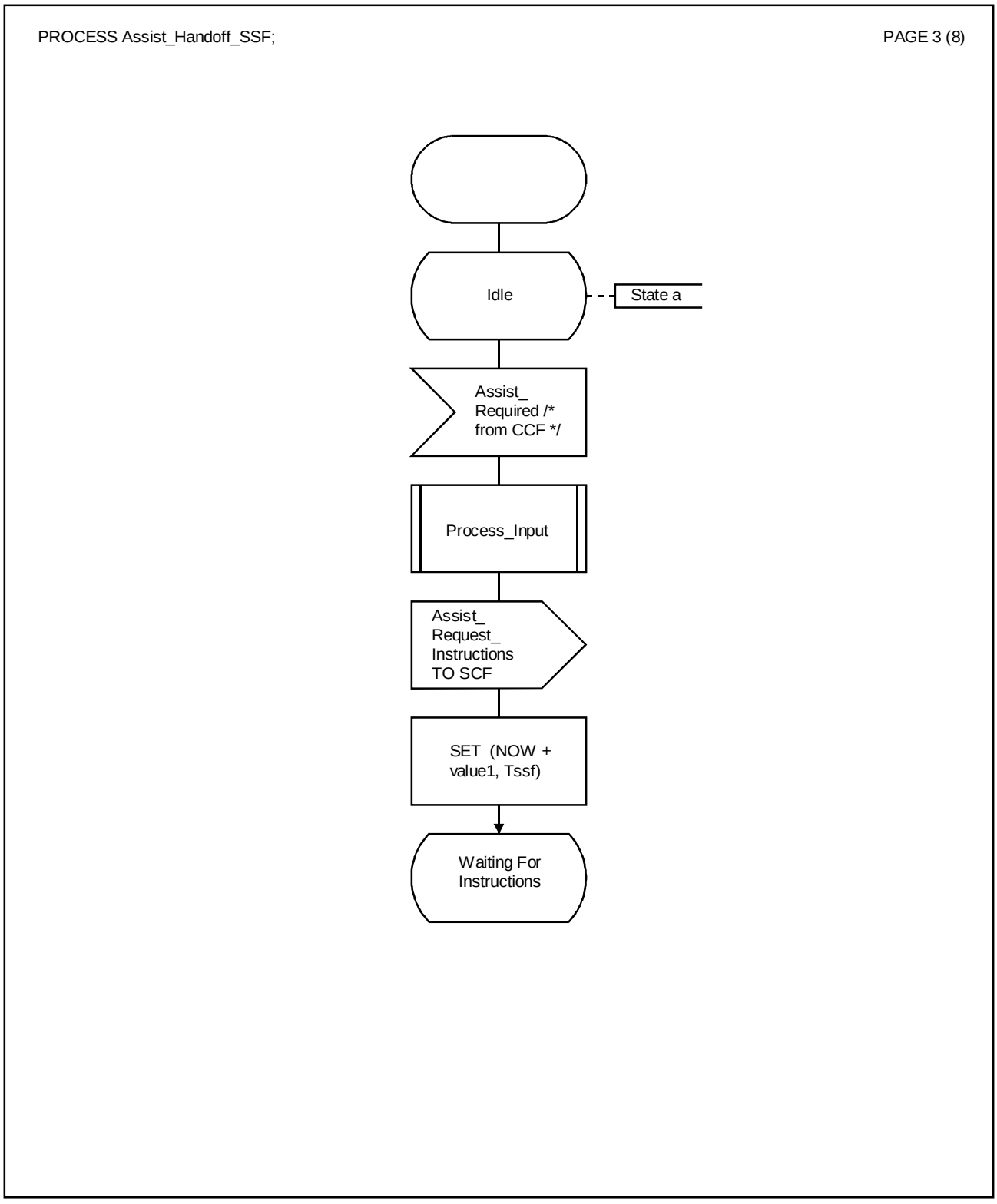


FIGURE A.2/Q.1218 (sheet 3 of 8)

SDL for assist/handoff SSF-FSM

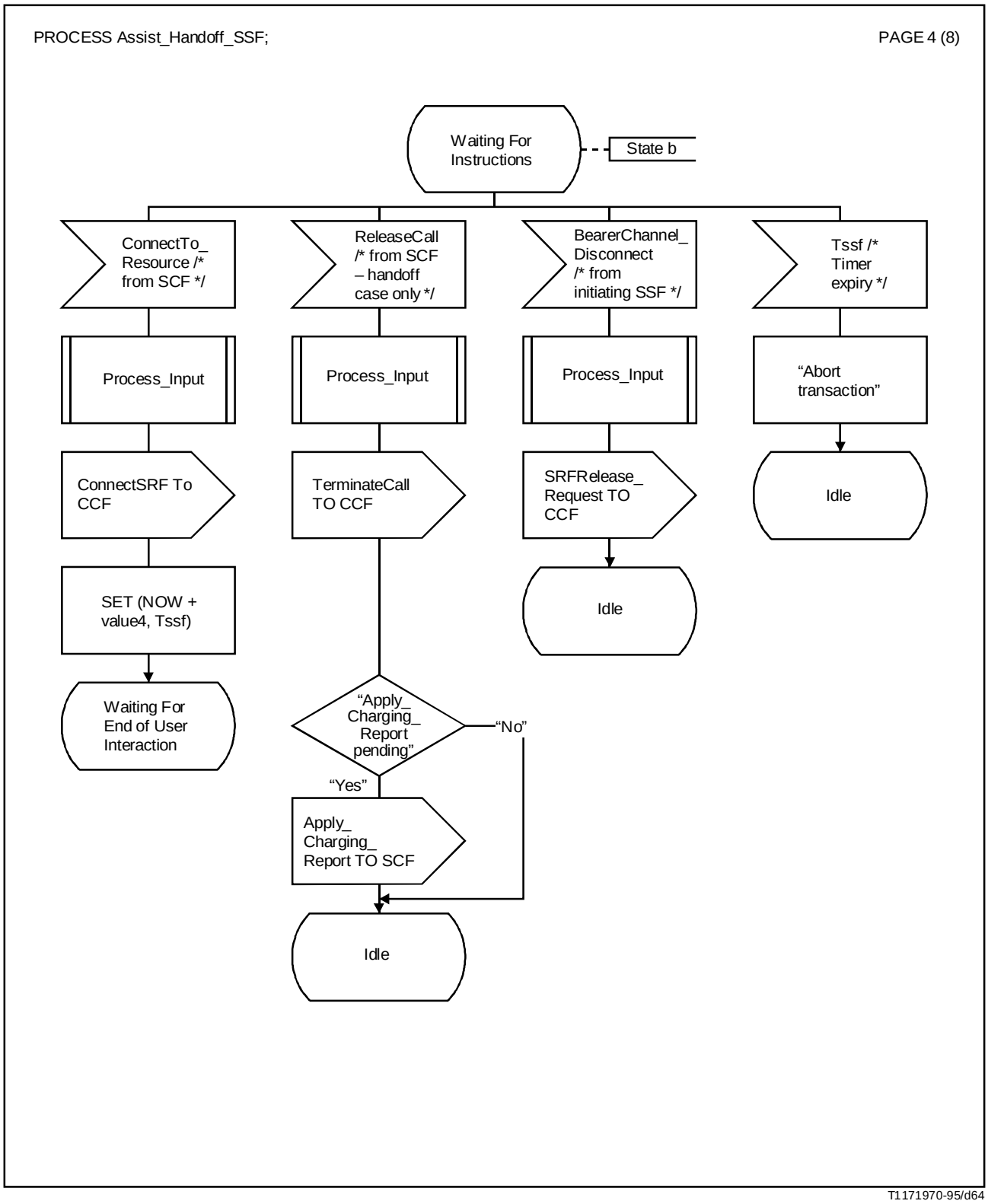
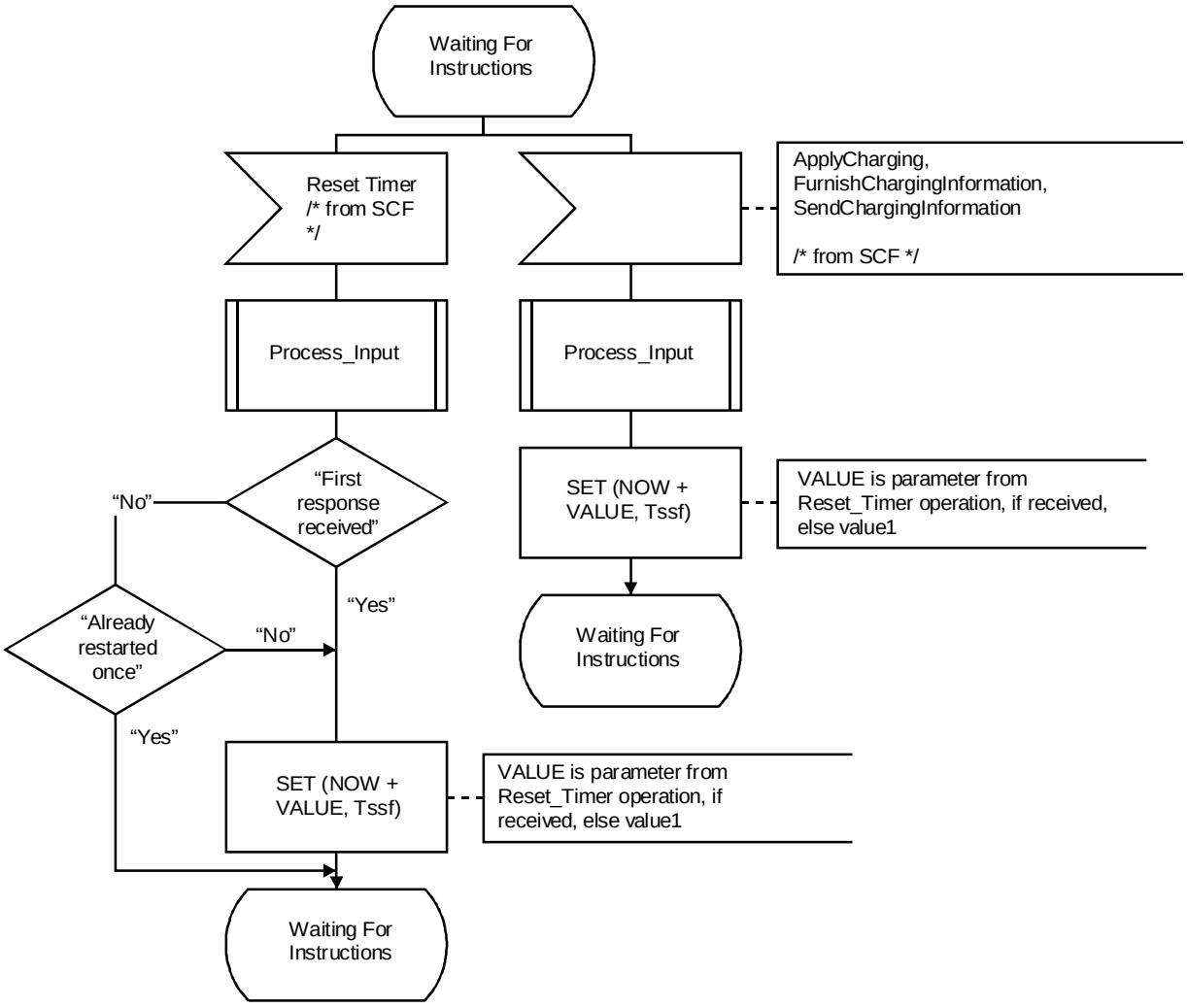


FIGURE A.2/Q.1218 (sheet 4 of 8)

SDL for assist/handoff SSF-FSM



T1171980-95/d65

FIGURE A.2/Q.1218 (sheet 5 of 8)

SDL for assist/handoff SSF-FSM

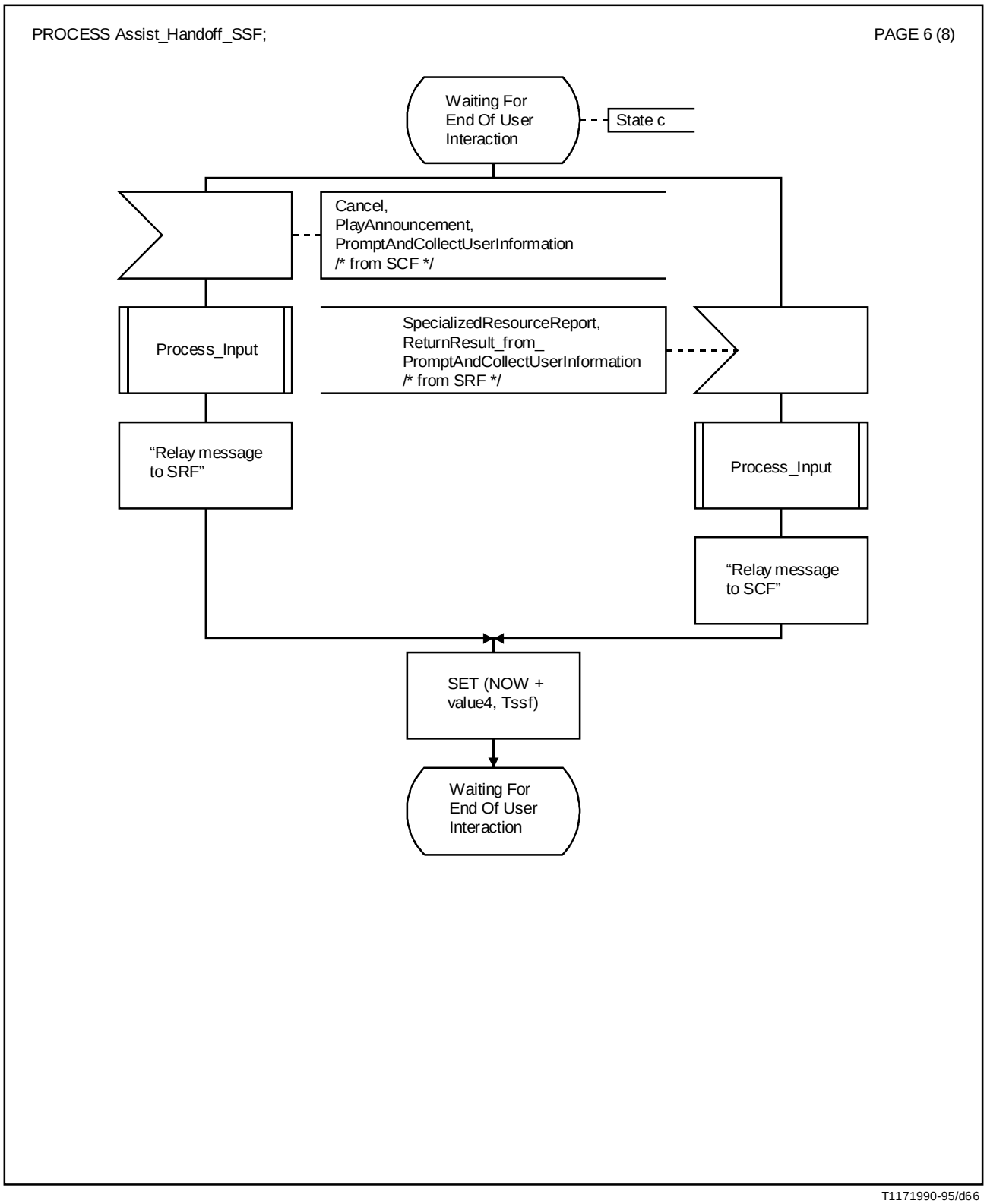


FIGURE A.2/Q.1218 (sheet 6 of 8)

SDL for assist/handoff SSF-FSM

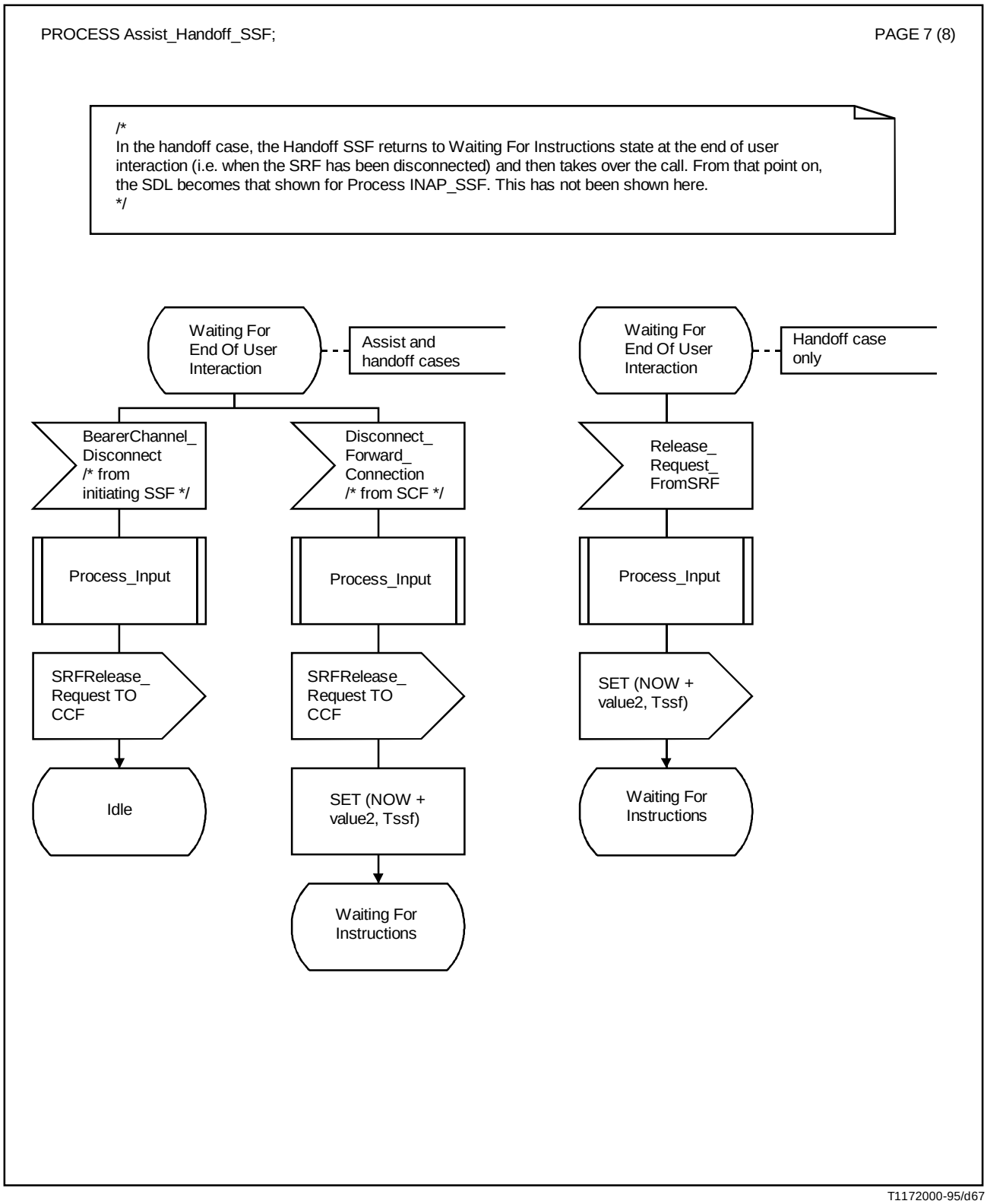


FIGURE A.2/Q.1218 (sheet 7 of 8)

SDL for assist/handoff SSF-FSM

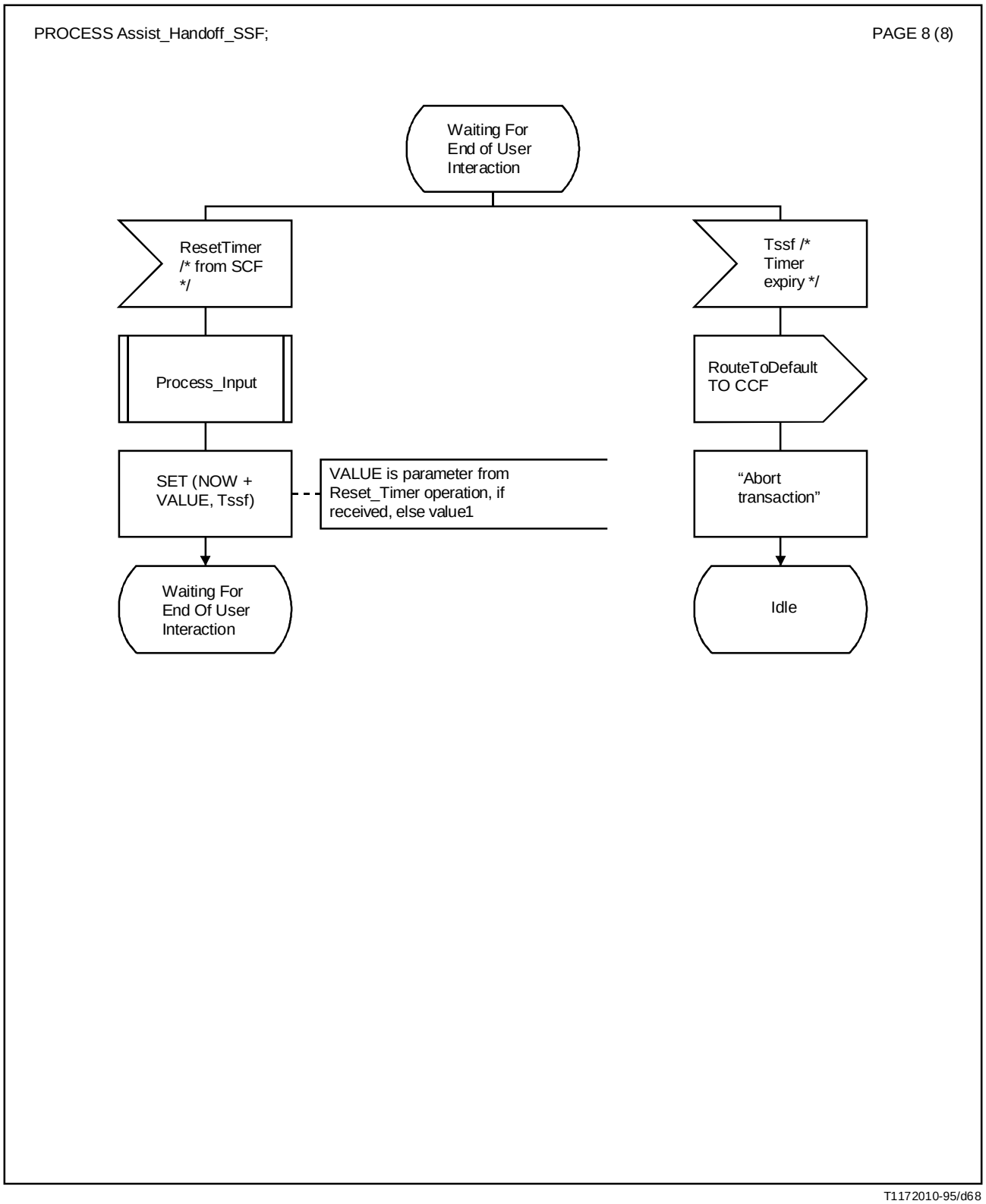


FIGURE A.2/Q.1218 (sheet 8 of 8)

SDL for assist/handoff SSF-FSM

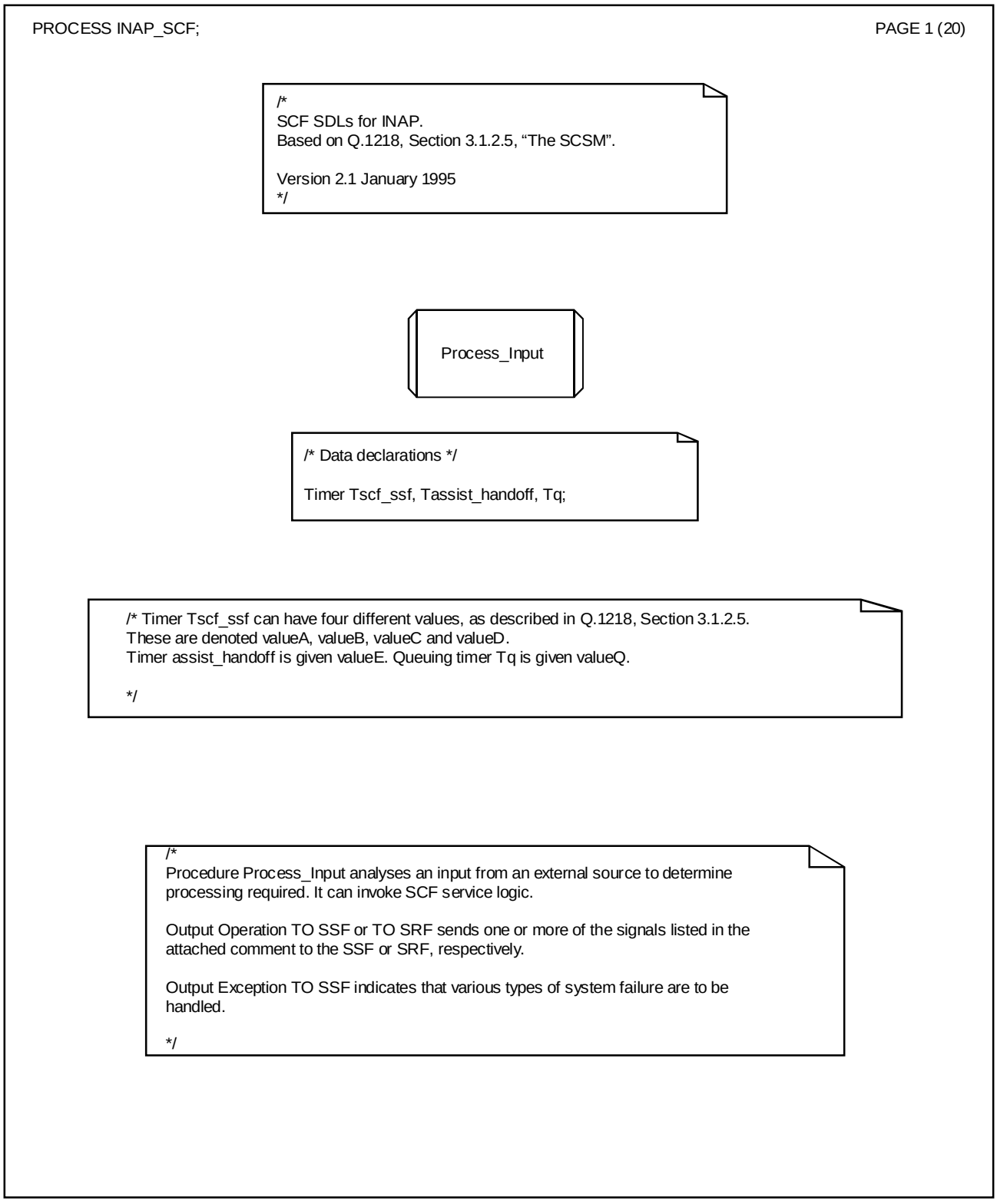


FIGURE A.3/Q.1218 (sheet 1 of 20)

SDL for SCF-FSM

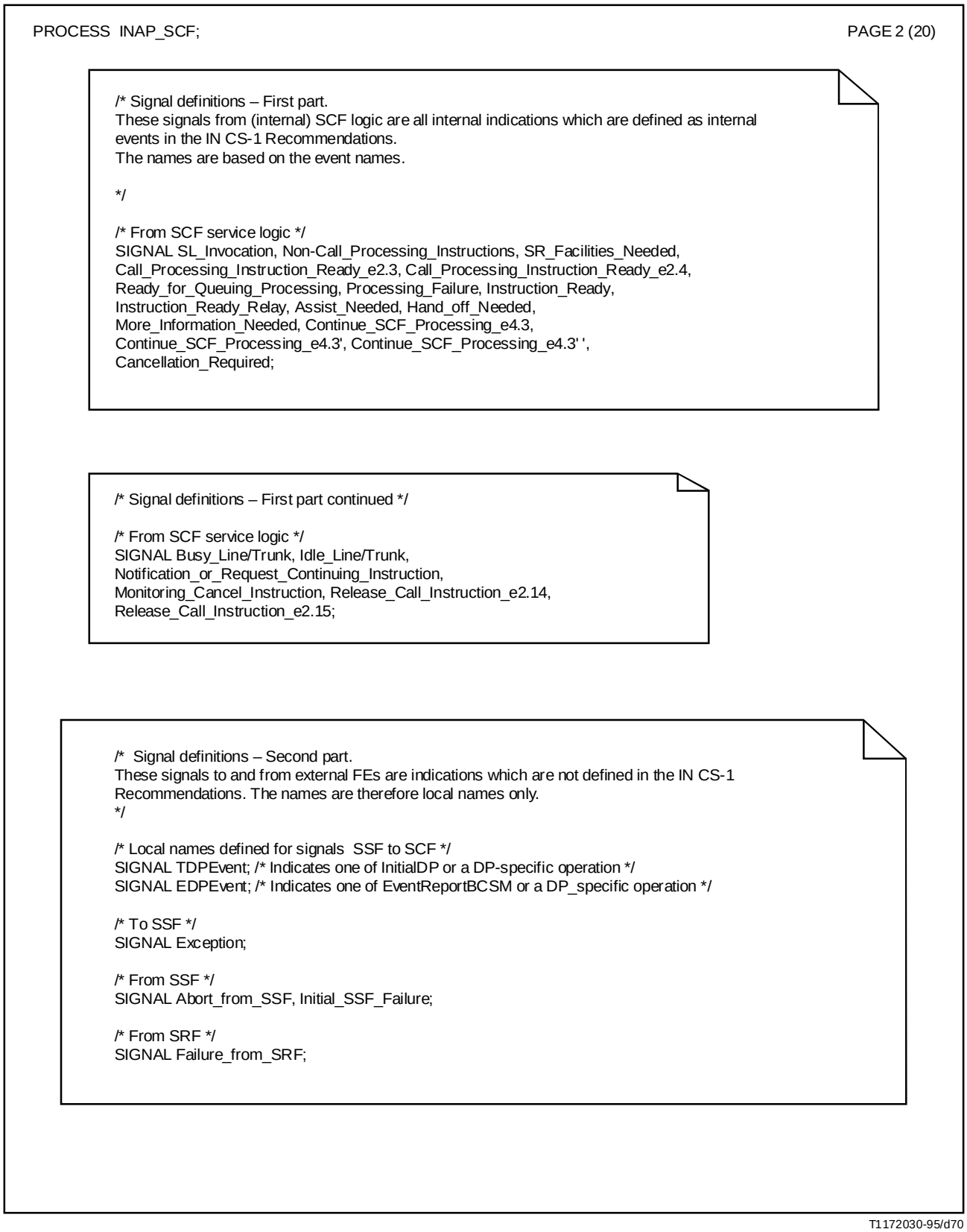


FIGURE A.3/Q.1218 (sheet 2 of 20)

SDL for SCF-FSM

```
/* Signal definitions – Third part. Defined in IN CS-1 Recommendations. */

/* From SSF or SRF */
SIGNAL AssistRequestInstructions;

/* From SSF – DP-specific operations */

SIGNAL AnalysedInformation, CollectedInformation, OAnswer,
OCalledPartyBusy, ODisconnect, OMidCall, ONoAnswer,
OriginationAttemptAuthorized, RouteSelectFailure, TAnswer,
TCalledPartyBusy, TDisconnect, TermAttemptAuthorized, TMidCall,
TNoAnswer;

/* Other operations from SSF */
SIGNAL InitialDP, EventReportBCSM, ApplyChargingReport,
CallInformationReport, EventNotificationCharging;
```

```
/* Signal definitions – Fourth part. Defined in IN CS-1 Recommendations. */

/* To SSF */
SIGNAL ResetTimer, ApplyCharging, CallInformationRequest,
Cancel, FurnishChargingInformation, RequestNotificationChargingEvent,
RequestReportBCSMEvent, SendChargingInformation,
AnalyseInformation, CollectInformation, Connect, Continue, ReleaseCall,
SelectFacility, SelectRoute, ConnectToResource,
EstablishTemporaryConnection;
```

```
/* Signal definitions – Fifth part. Operations defined in IN CS-1 Recommendations. */

/* To SRF */
SIGNAL PlayAnnouncement, PromptAndCollectUserInformation;

/* From SRF */
SIGNAL ReturnResult_from_PromptAndCollectUserInformation,
SpecializedResourceReport;
```

T1172040-95/d71

FIGURE A.3/Q.1218 (sheet 3 of 20)

SDL for SCF-FSM

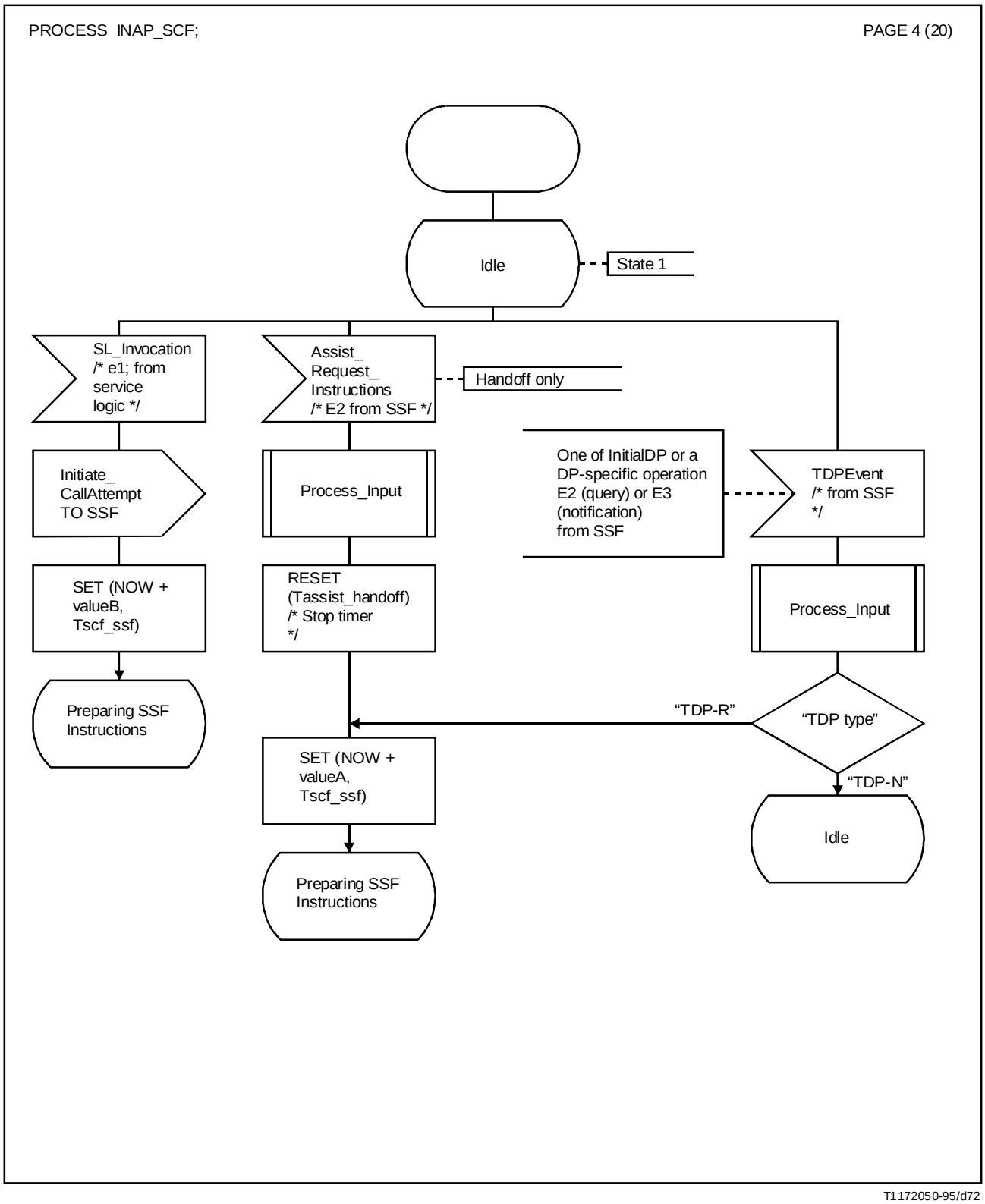


FIGURE A.3/Q.1218 (sheet 4 of 20)

SDL for SCF-FSM

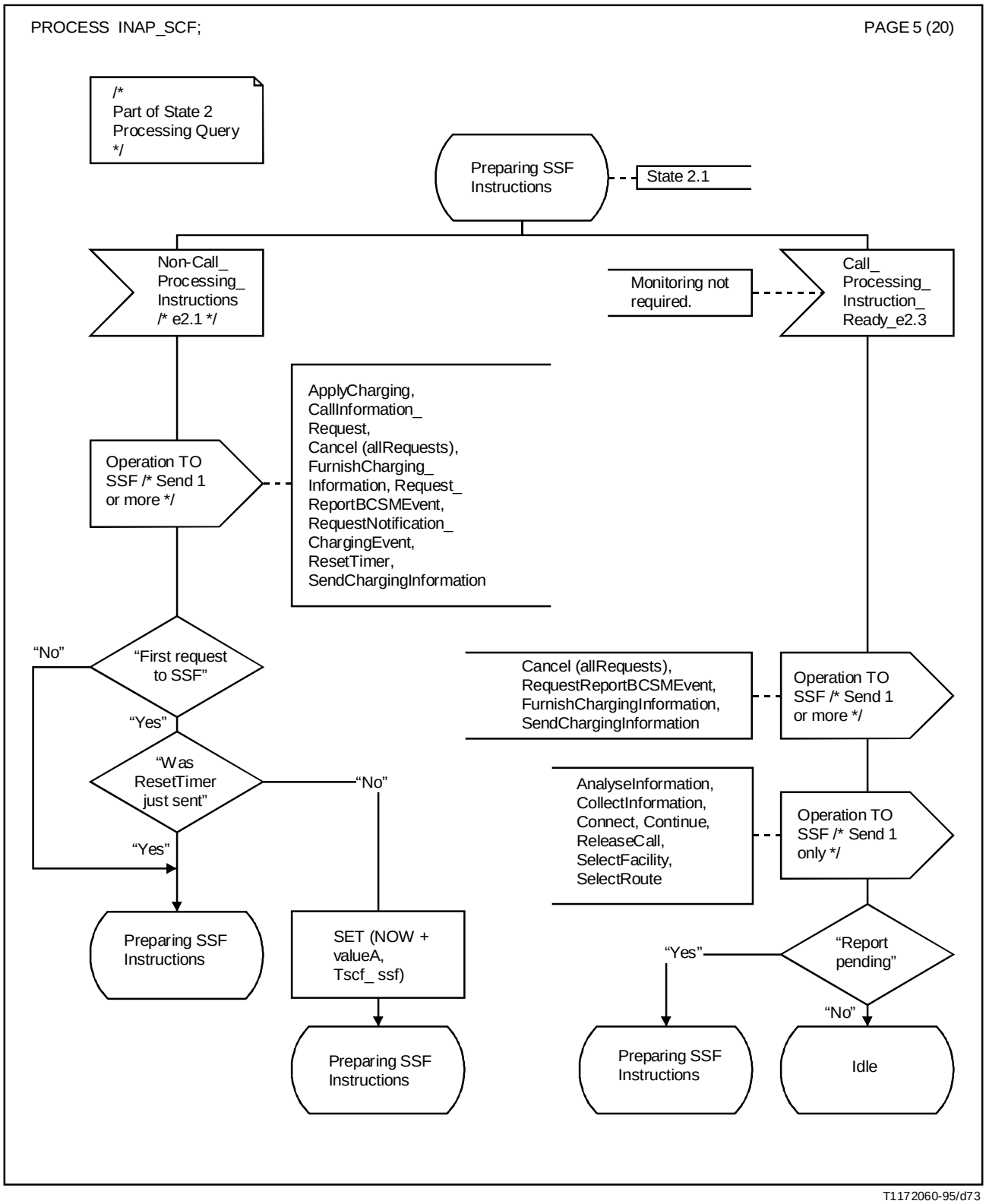


FIGURE A.3/Q.1218 (sheet 5 of 20)

SDL for SCF-FSM

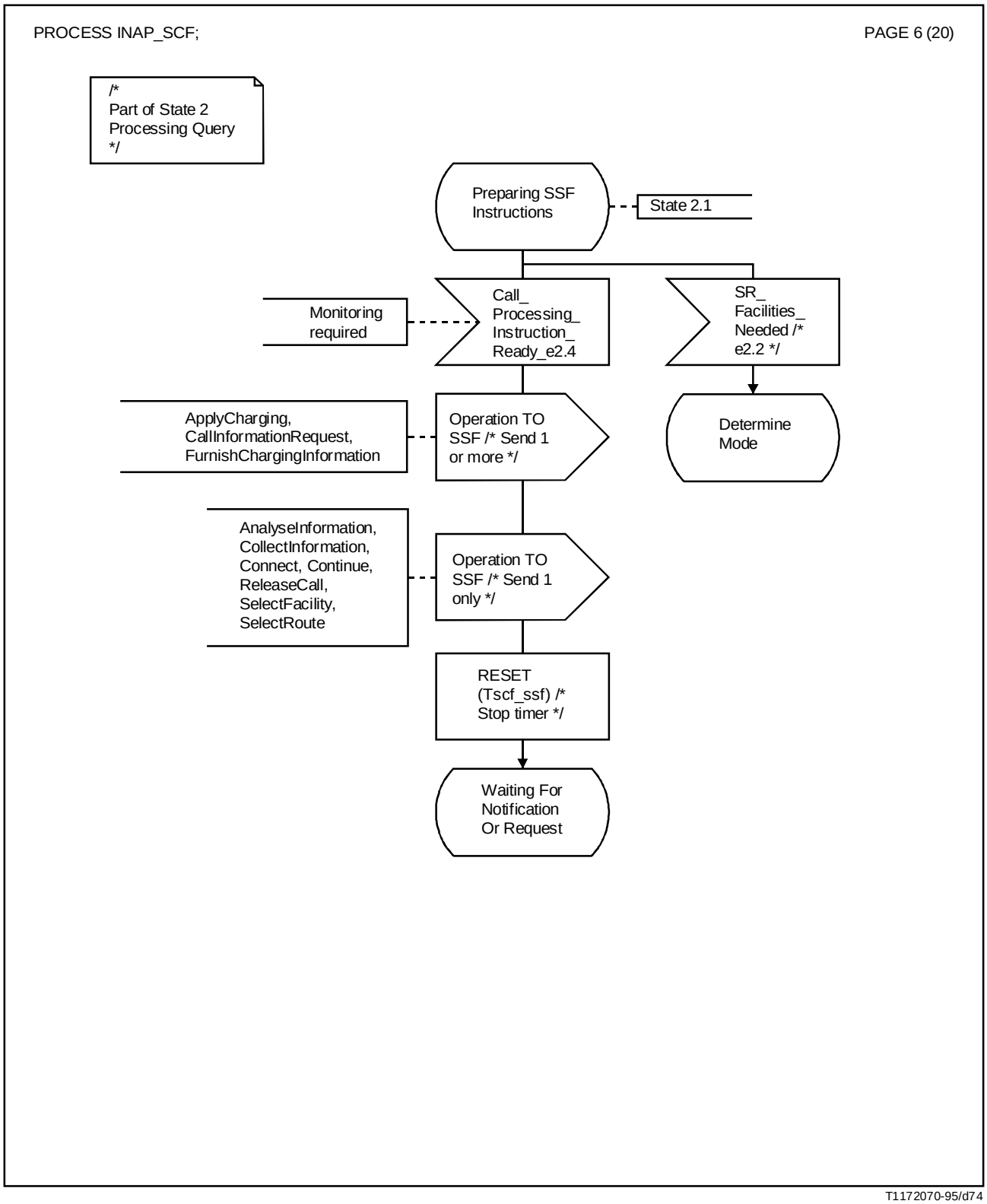


FIGURE A.3/Q.1218 (sheet 6 of 20)

SDL for SCF-FSM

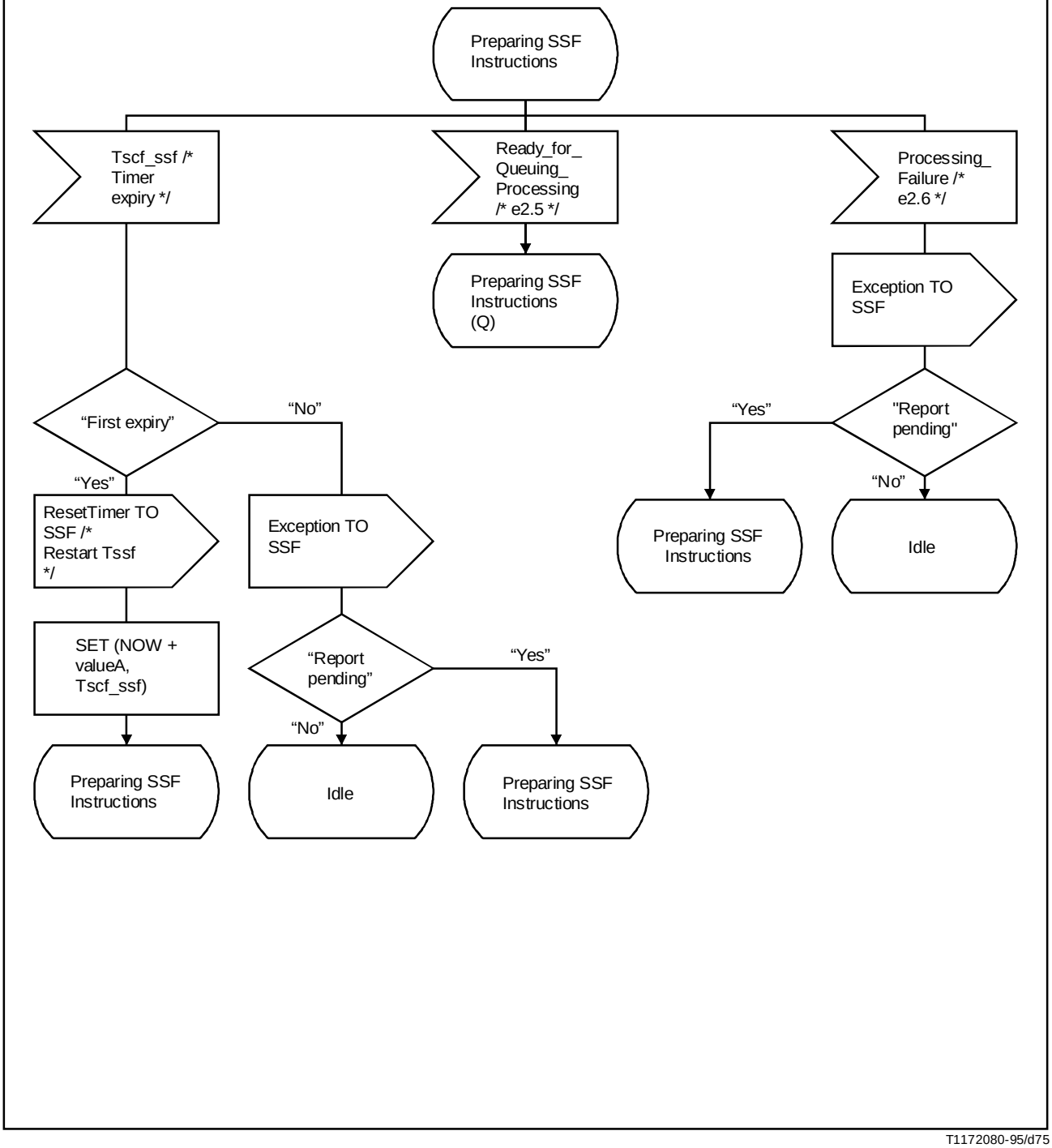


FIGURE A.3/Q.1218 (sheet 7 of 20)

SDL for SCF-FSM

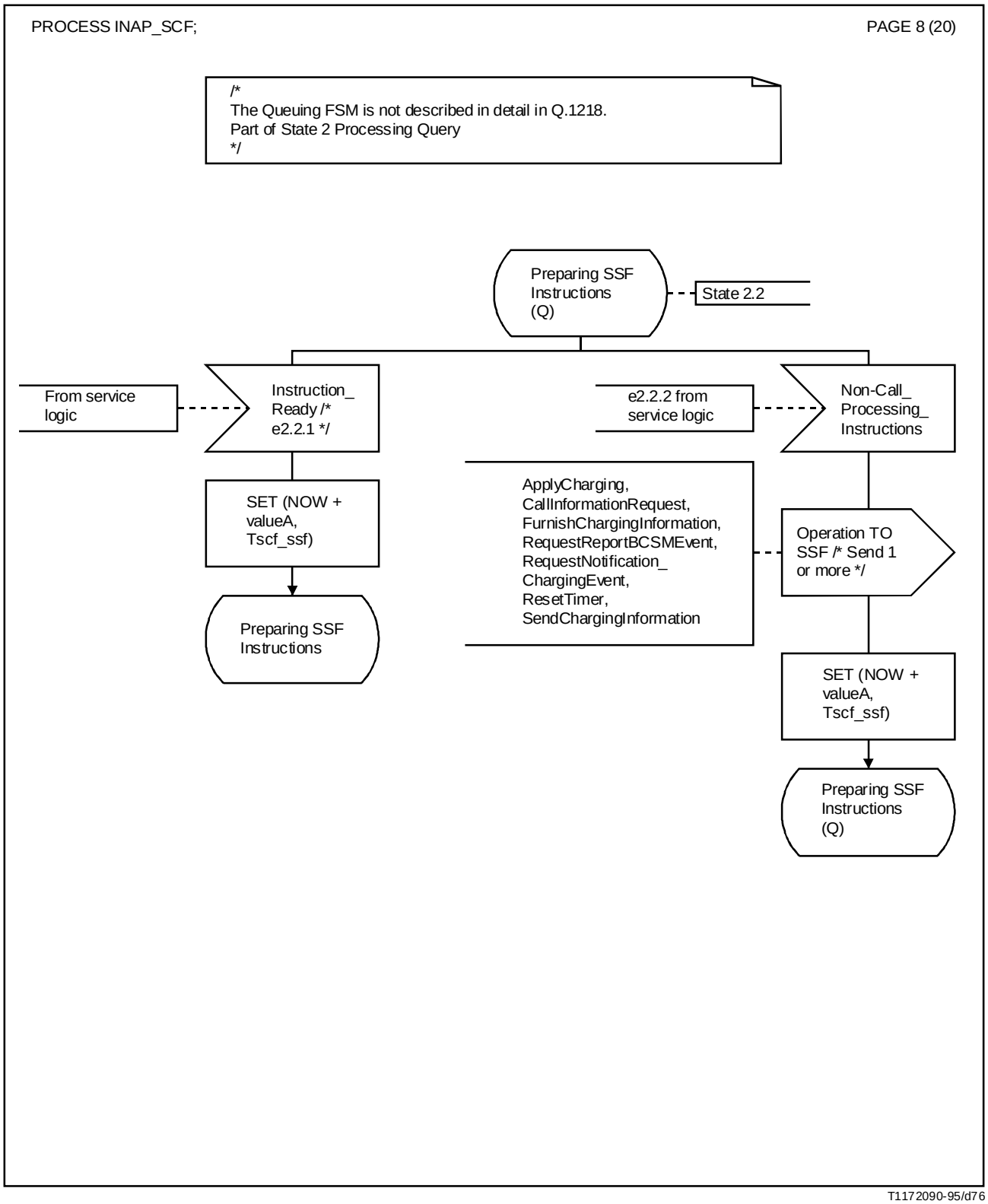
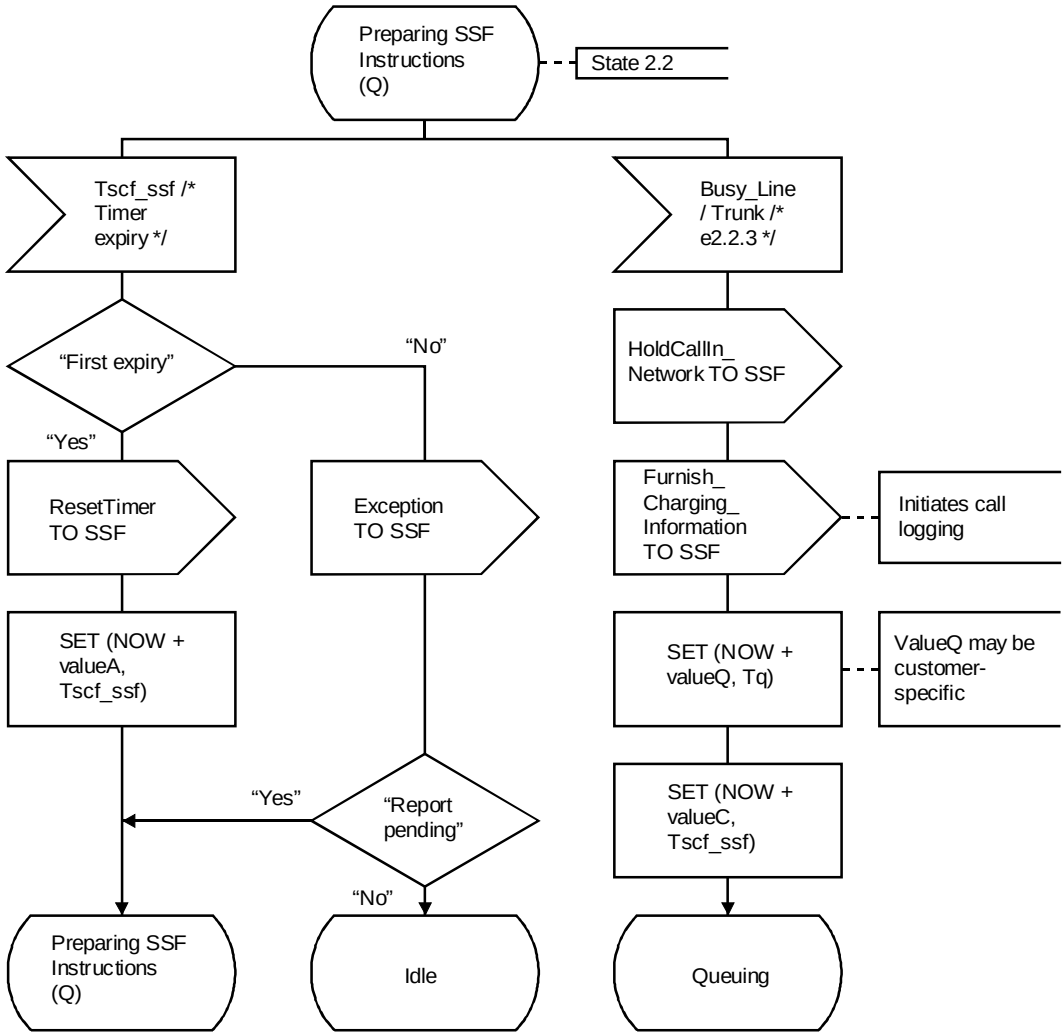


FIGURE A.3/Q.1218 (sheet 8 of 20)

SDL for SCF-FSM



T1172100-95/d77

FIGURE A.3/Q.1218 (sheet 9 of 20)

SDL for SCF-FSM

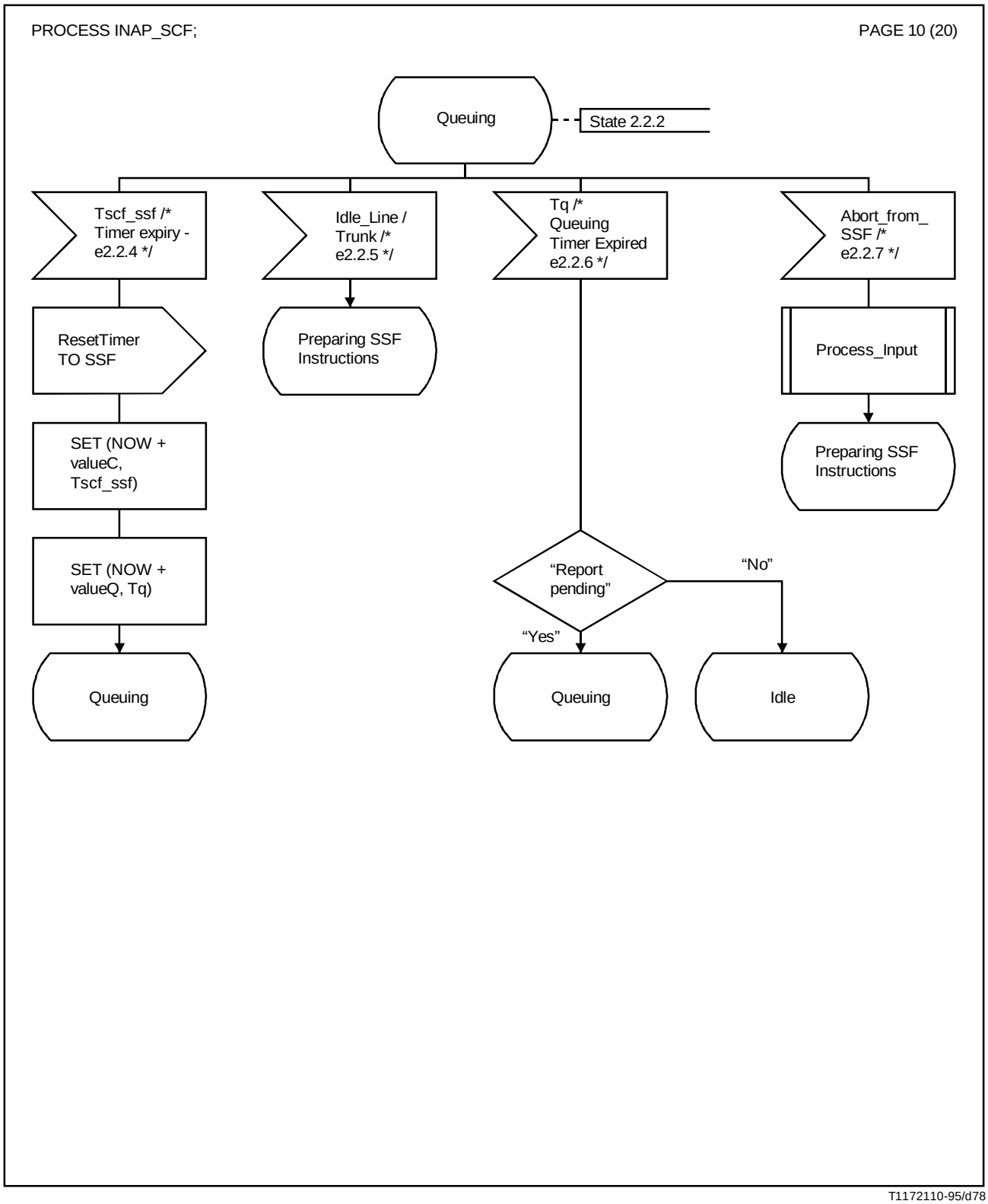


FIGURE A.3/Q.1218 (sheet 10 of 20)

SDL for SCF-FSM

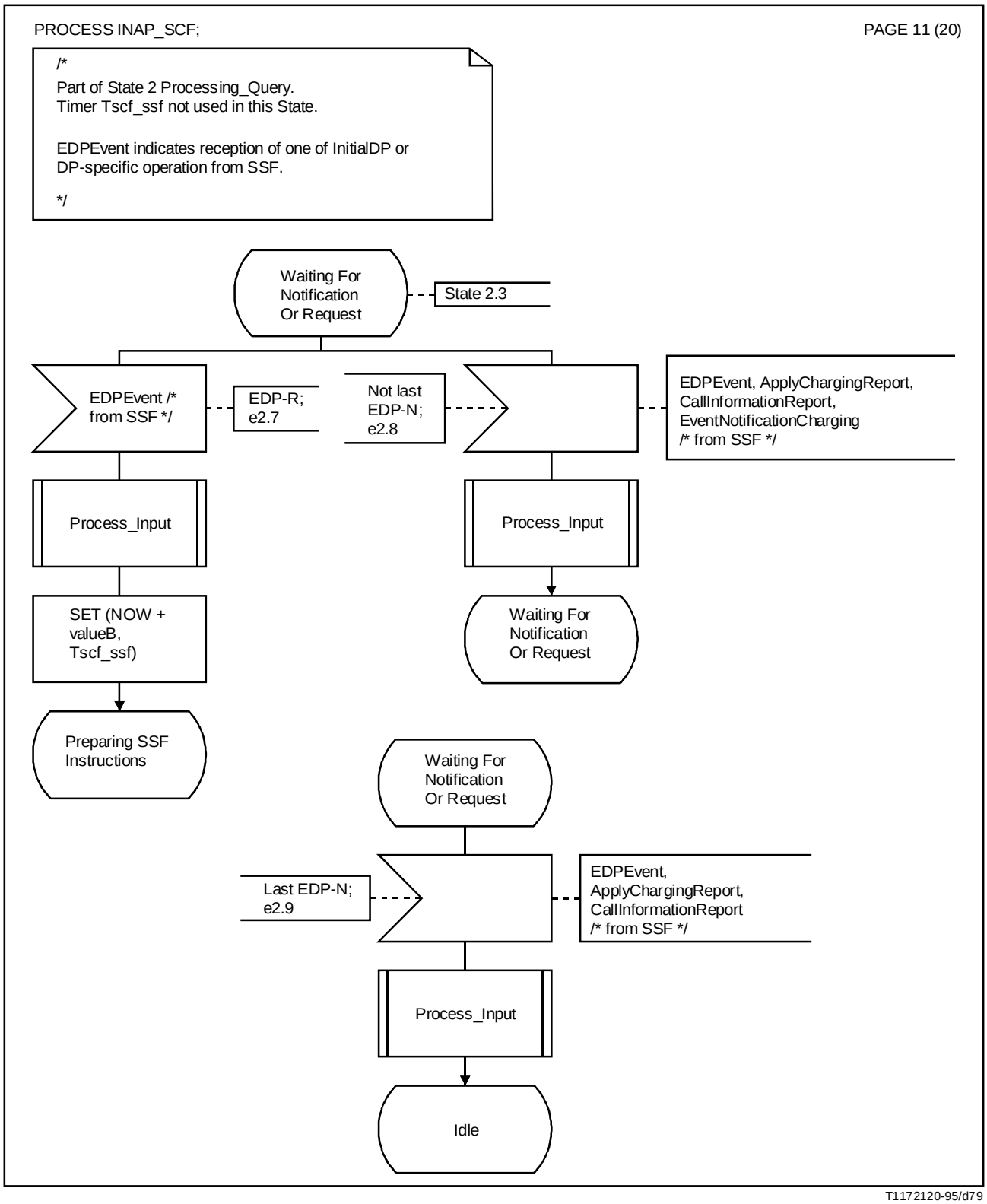


FIGURE A.3/Q.1218 (sheet 11 of 20)

SDL for SCF-FSM

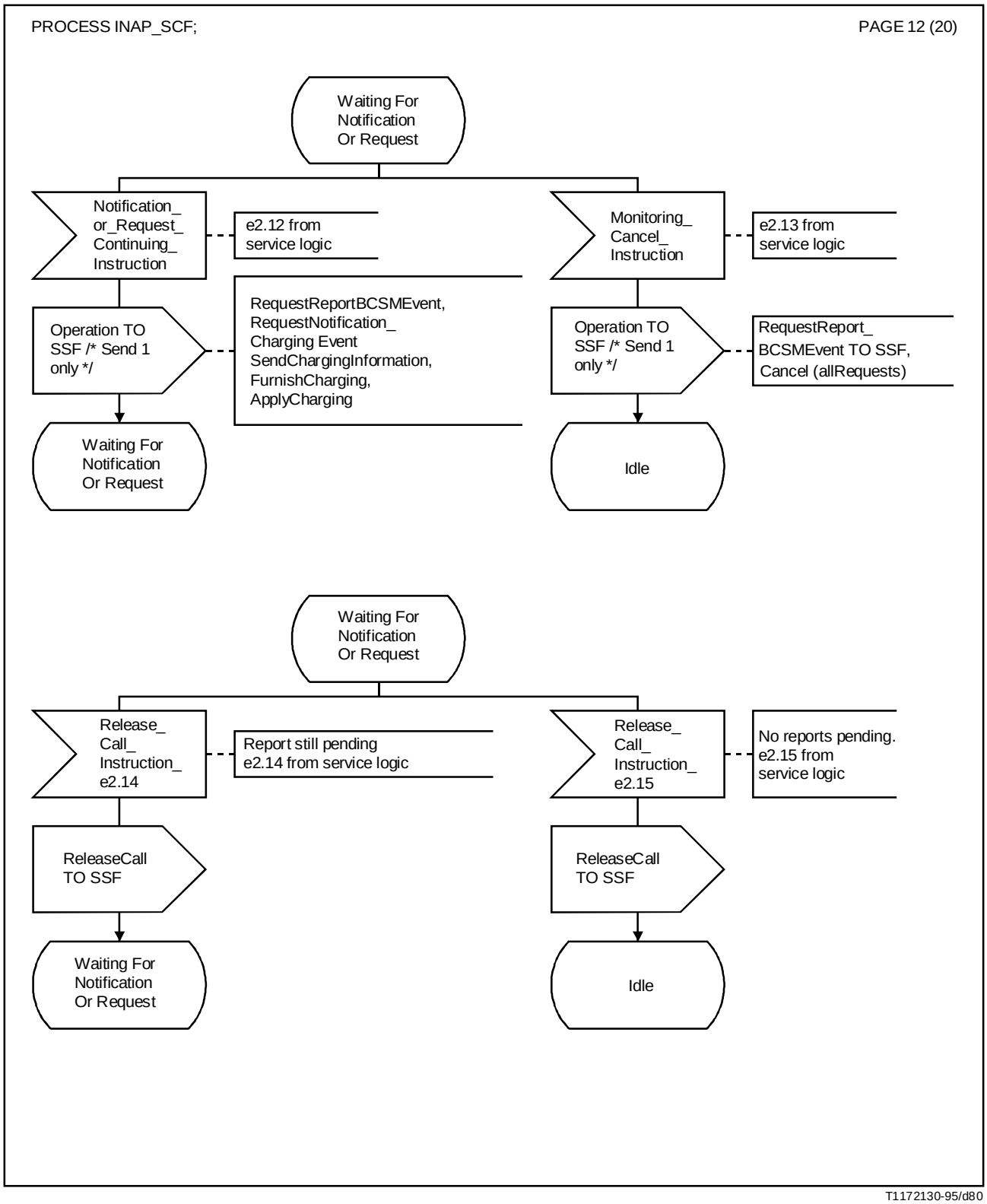


FIGURE A.3/Q.1218 (sheet 12 of 20)

SDL for SCF-FSM

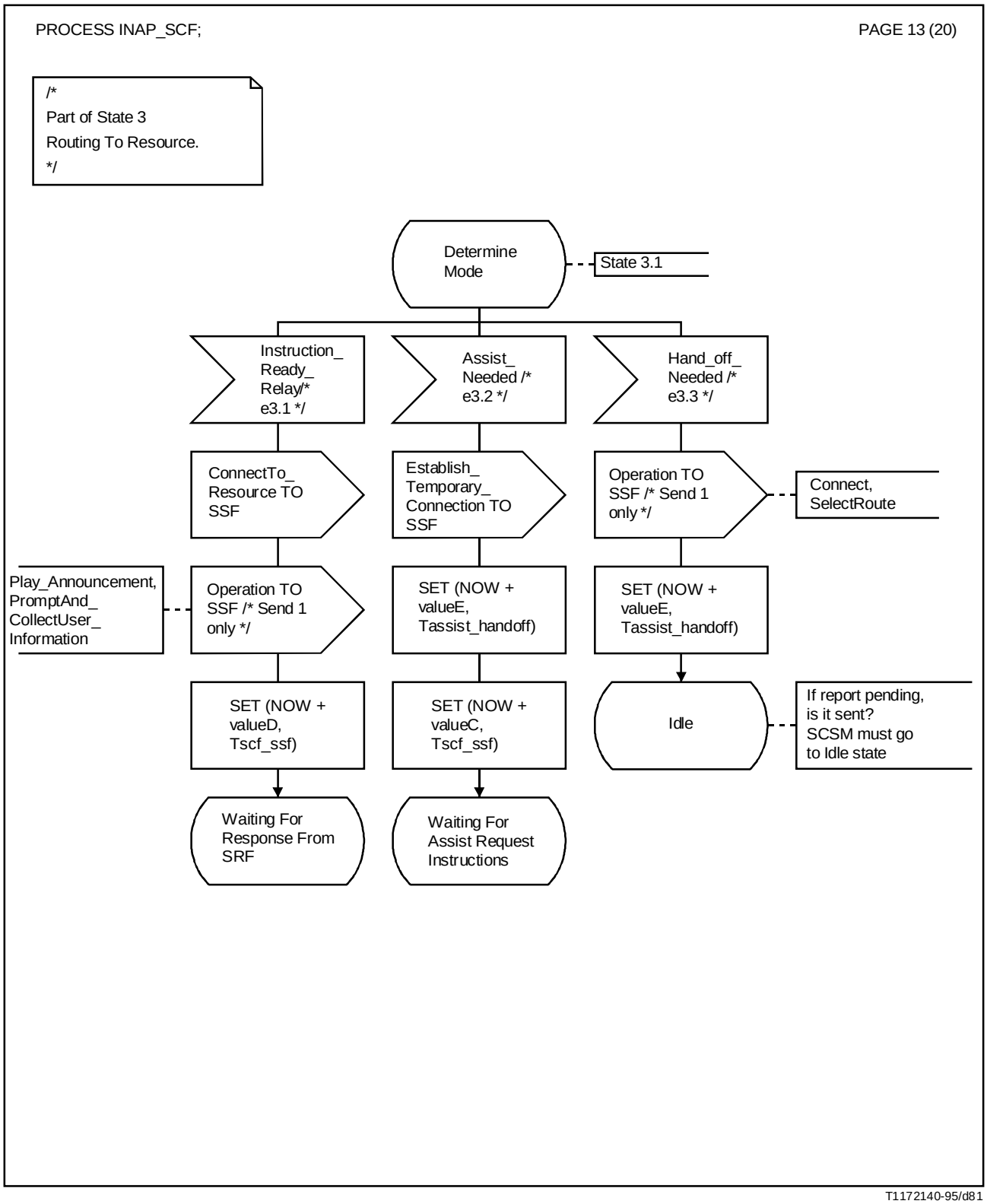


FIGURE A.3/Q.1218 (sheet 13 of 20)

SDL for SCF-FSM

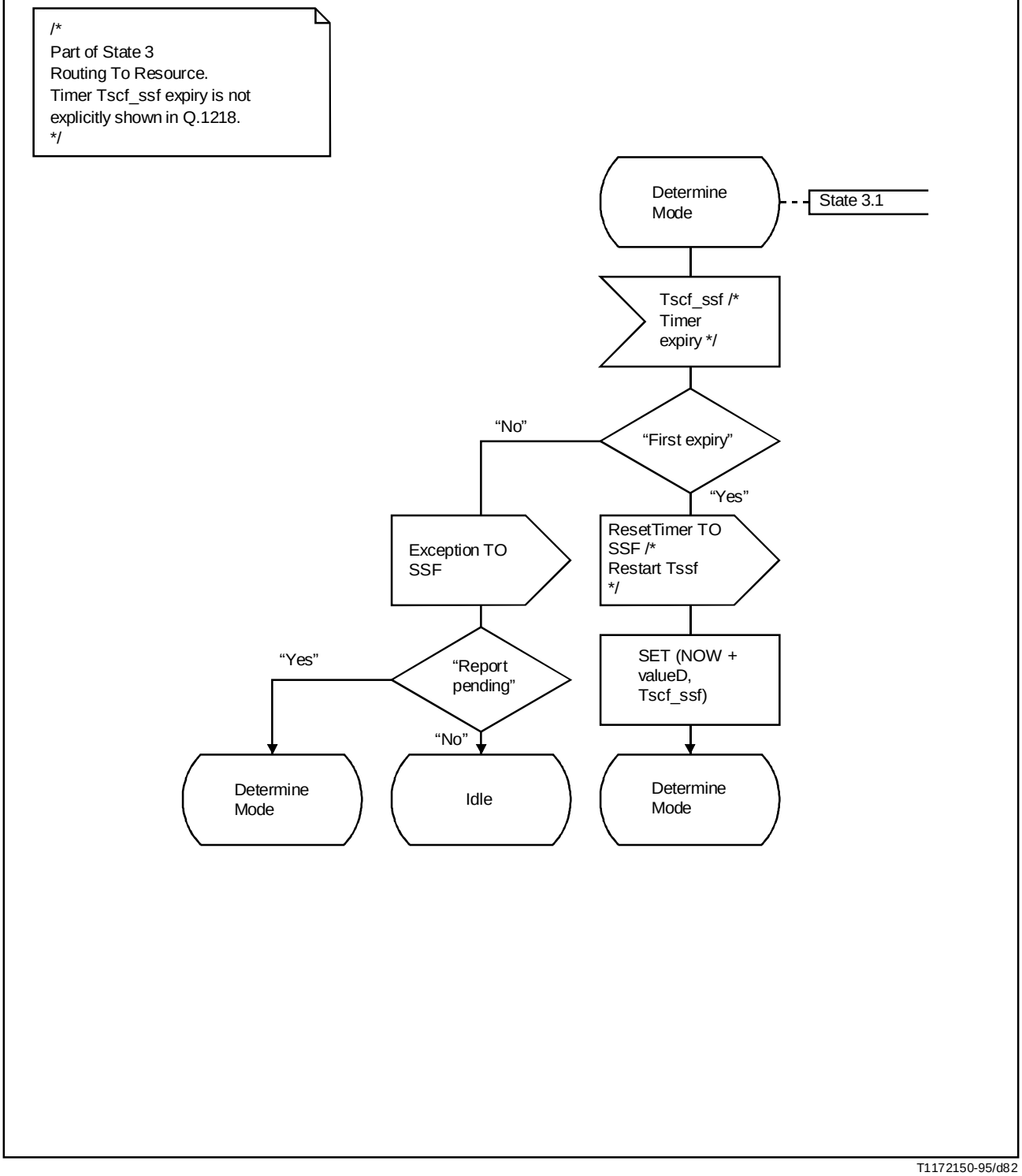


FIGURE A.3/Q.1218 (sheet 14 of 20)

SDL for SCF-FSM

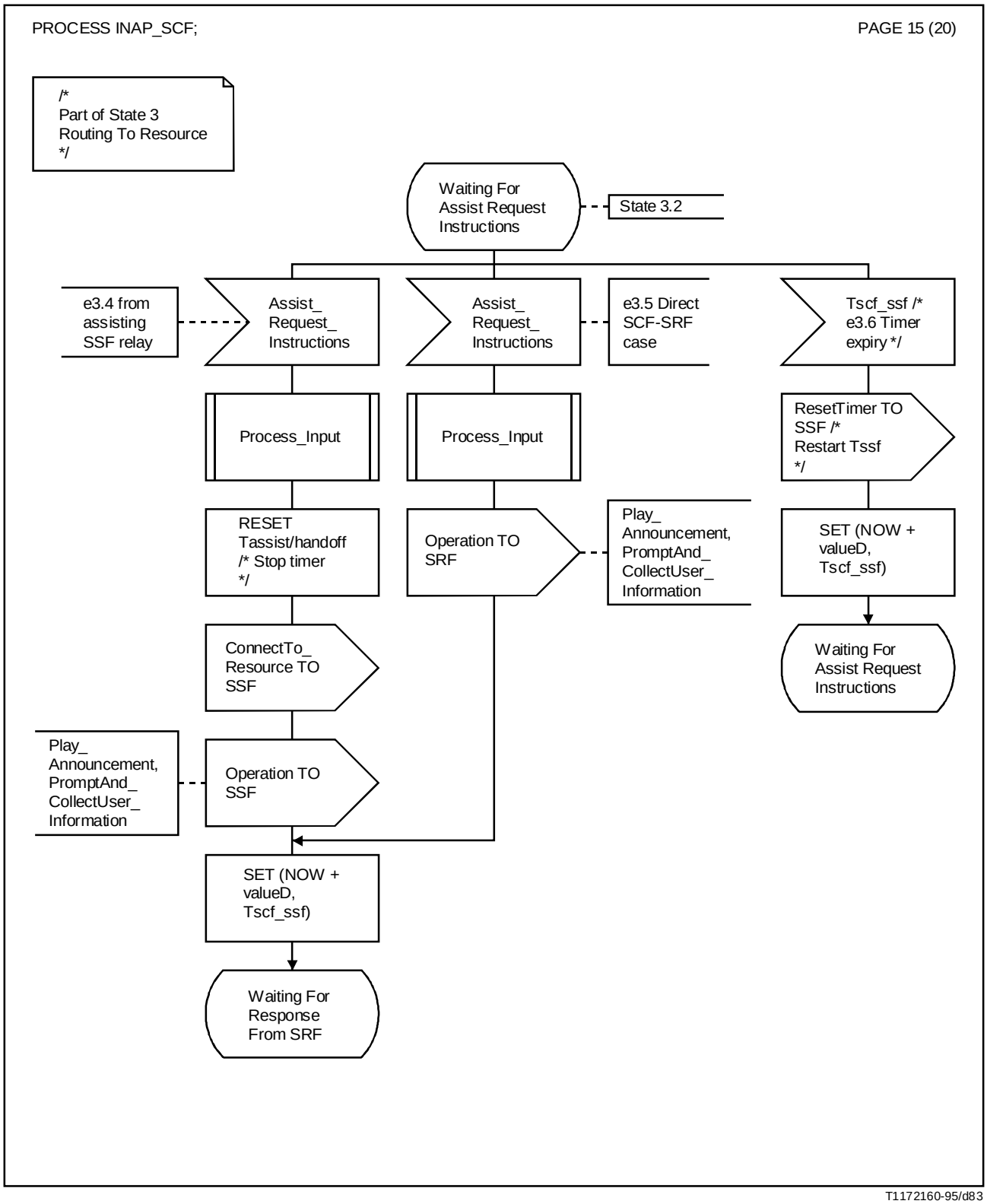


FIGURE A.3/Q.1218 (sheet 15 of 20)

SDL for SCF-FSM

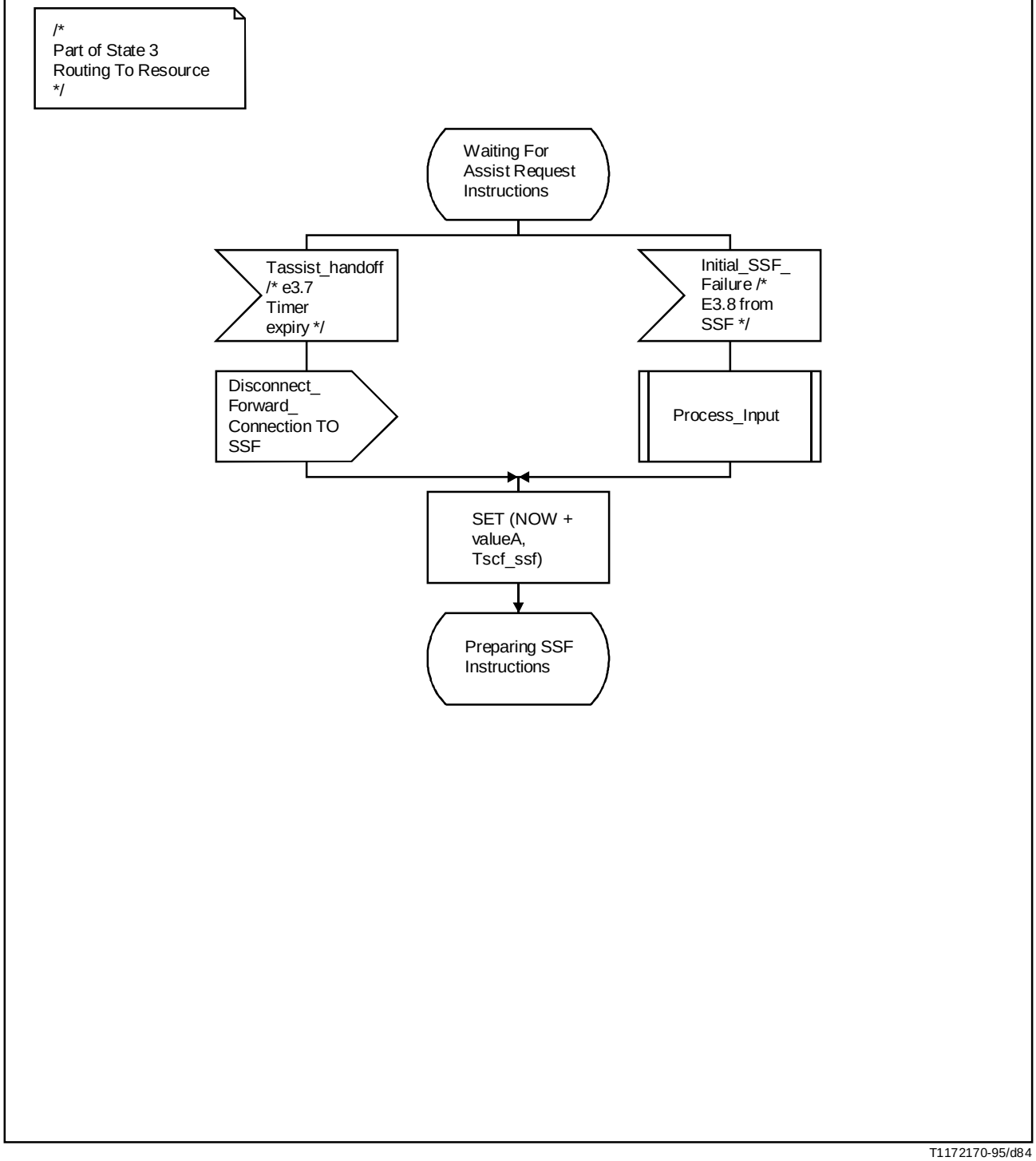


FIGURE A.3/Q.1218 (sheet 16 of 20)

SDL for SCF-FSM

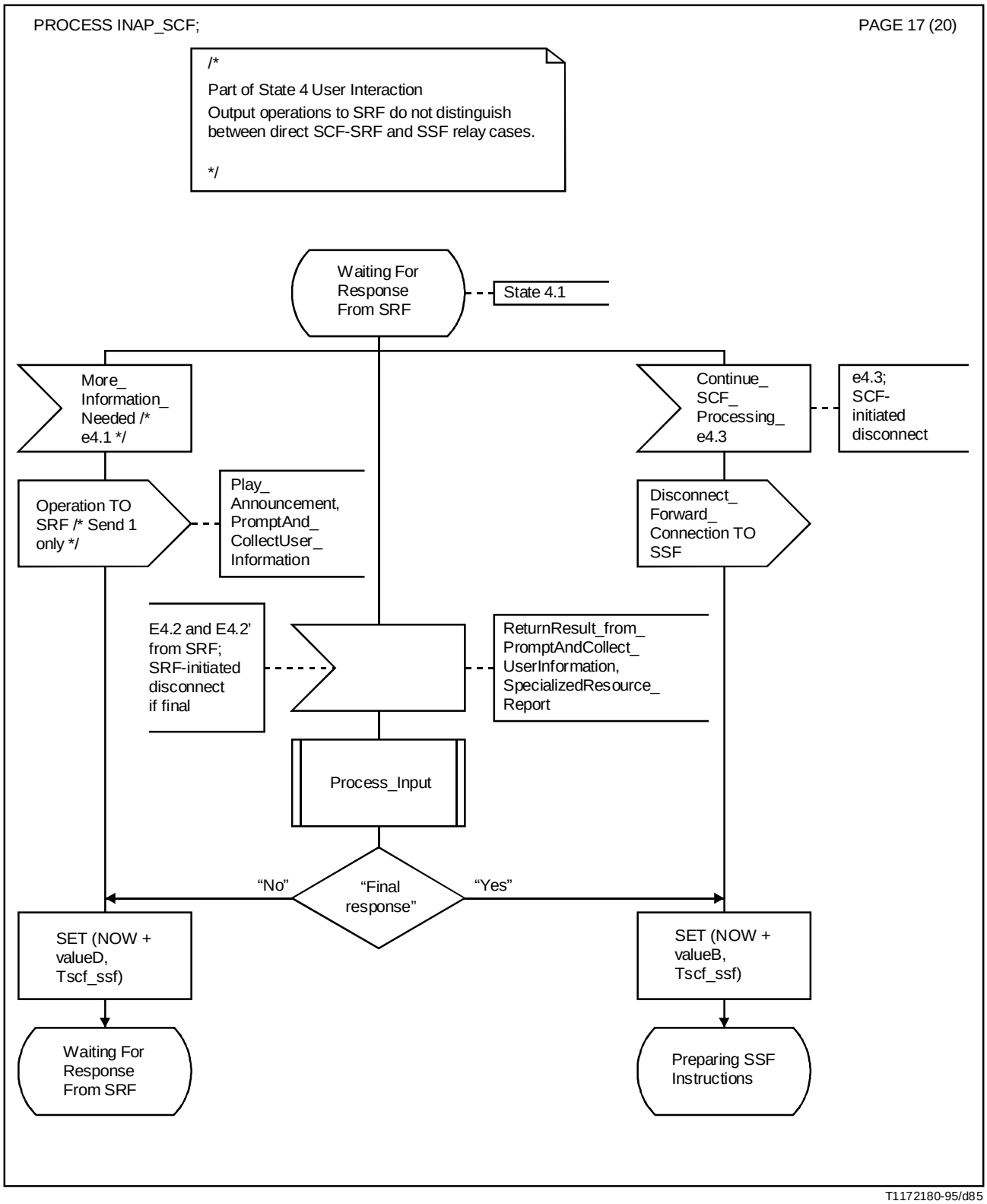


FIGURE A.3/Q.1218 (sheet 17 of 20)

SDL for SCF-FSM

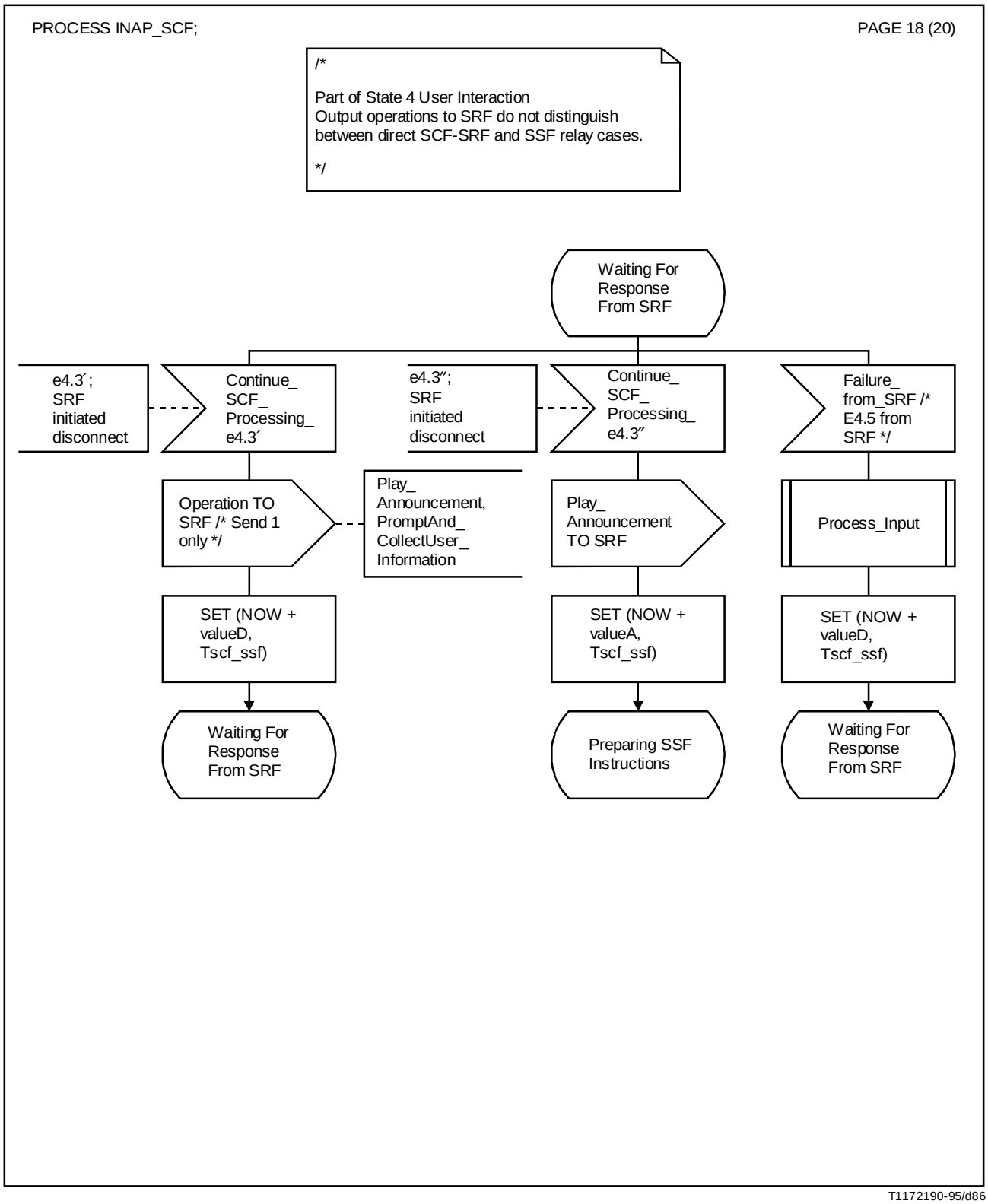


FIGURE A.3/Q.1218 (sheet 18 of 20)

SDL for SCF-FSM

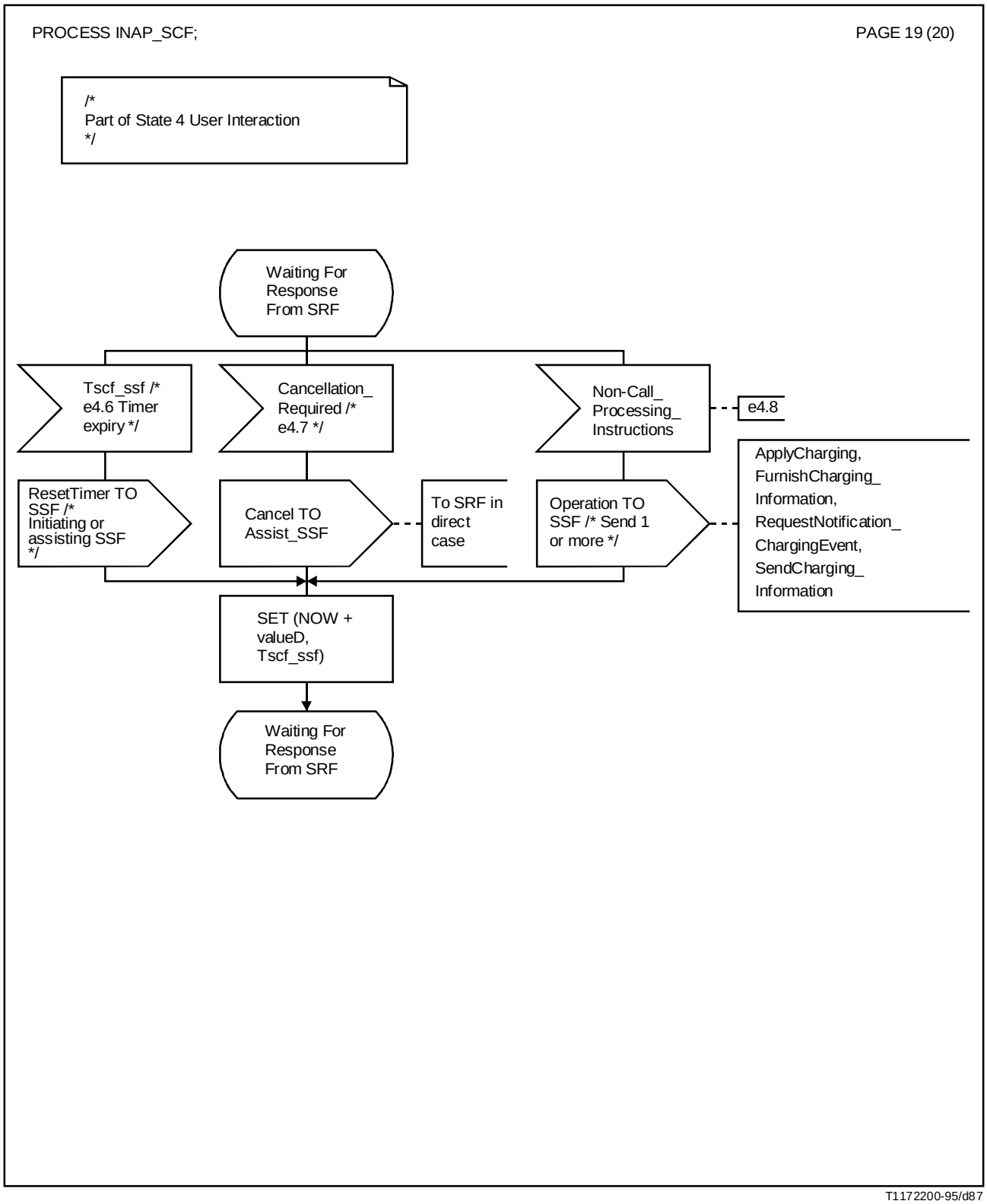


FIGURE A.3/Q.1218 (sheet 19 of 20)

SDL for SCF-FSM

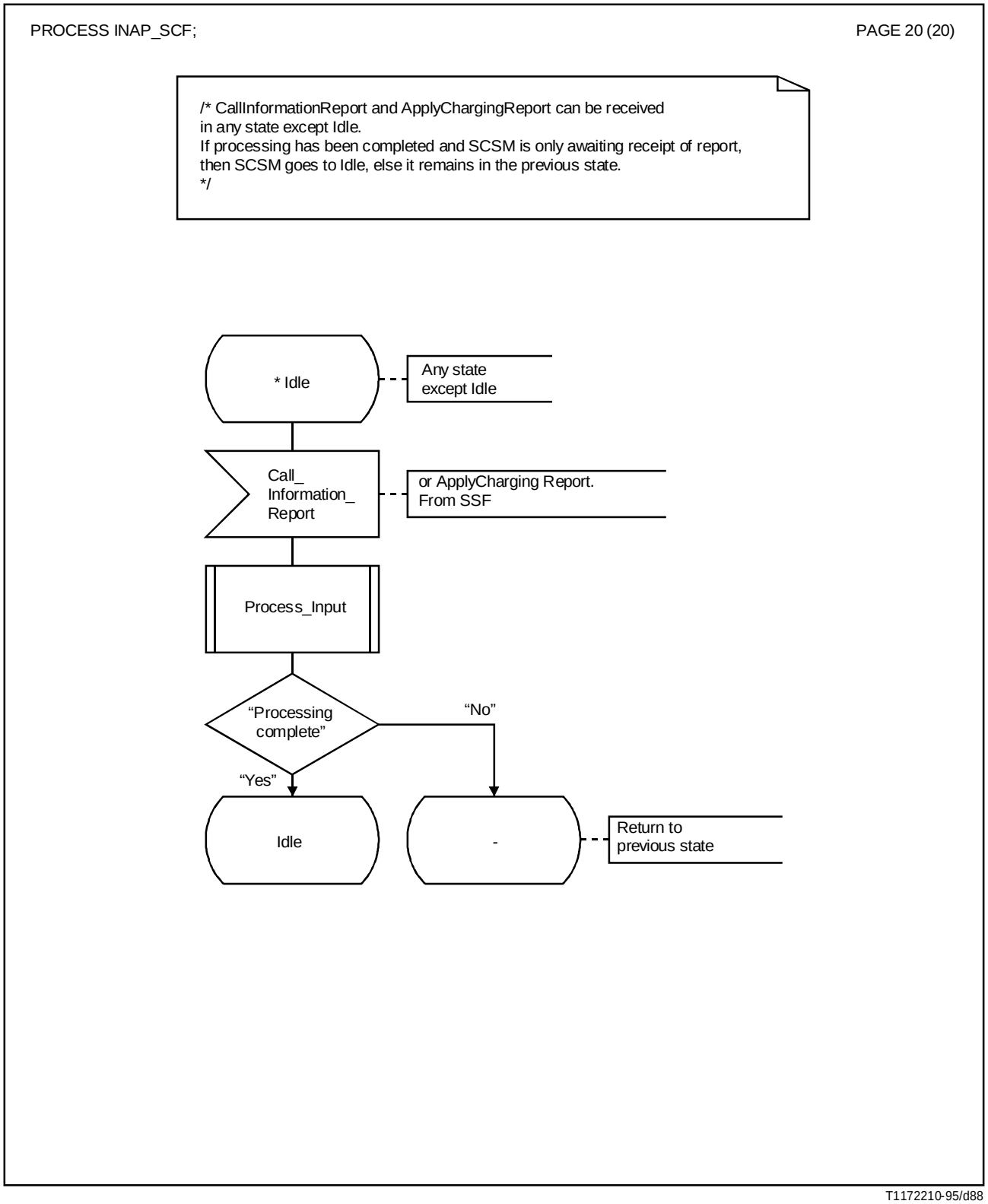


FIGURE A.3/Q.1218 (sheet 20 of 20)

SDL for SCF-FSM

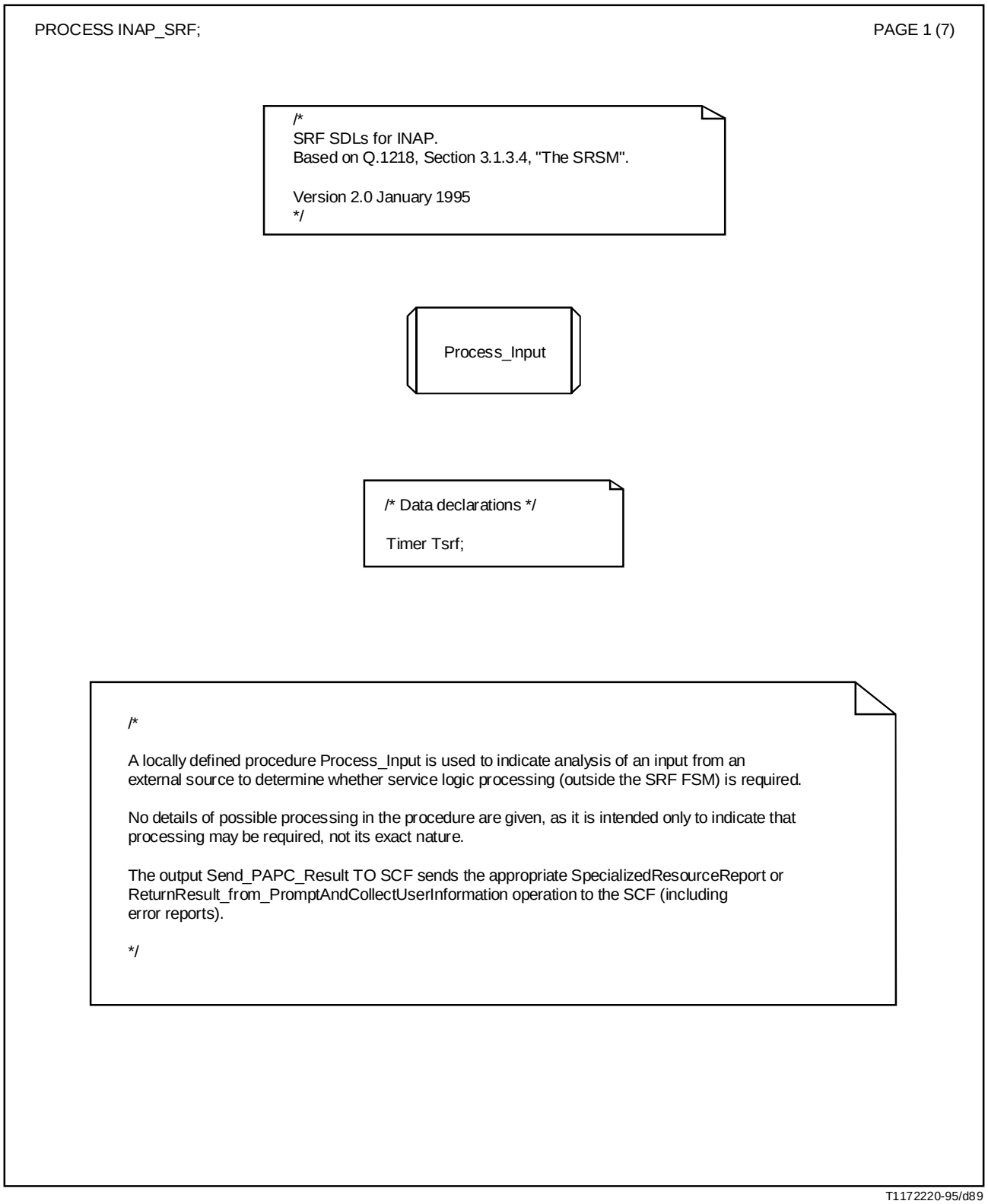


FIGURE A.4/Q.1218 (sheet 1 of 7)

SDL for SRF-FSM

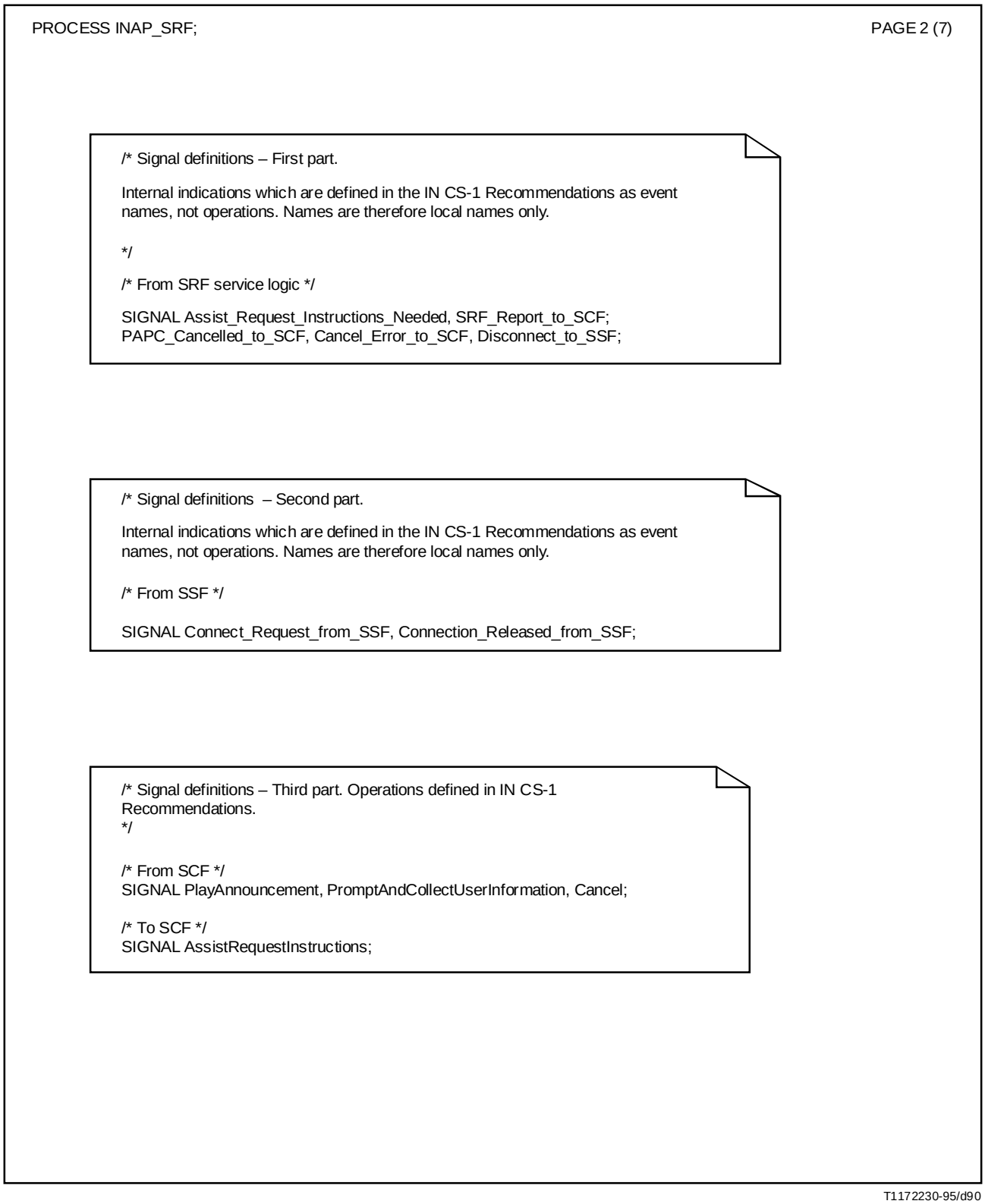
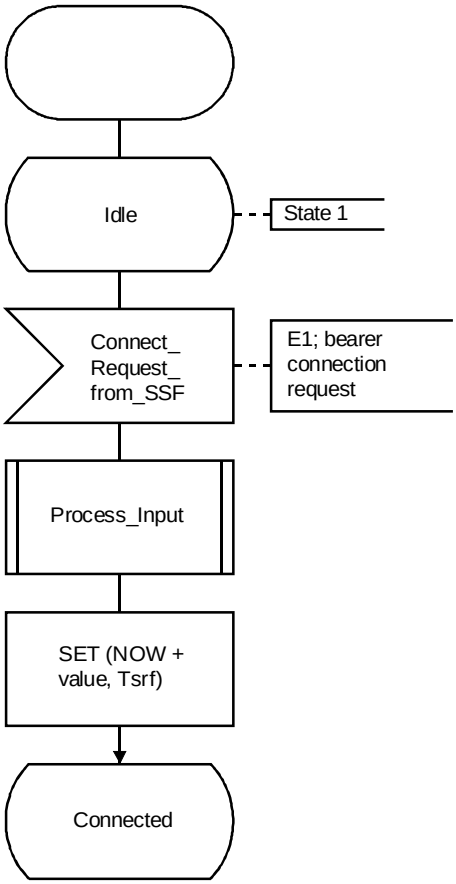


FIGURE A.4/Q.1218 (sheet 2 of 7)

SDL for SRF-FSM



T1172240-95/d91

FIGURE A.4/Q.1218 (sheet 3 of 7)

SDL for SRF-FSM

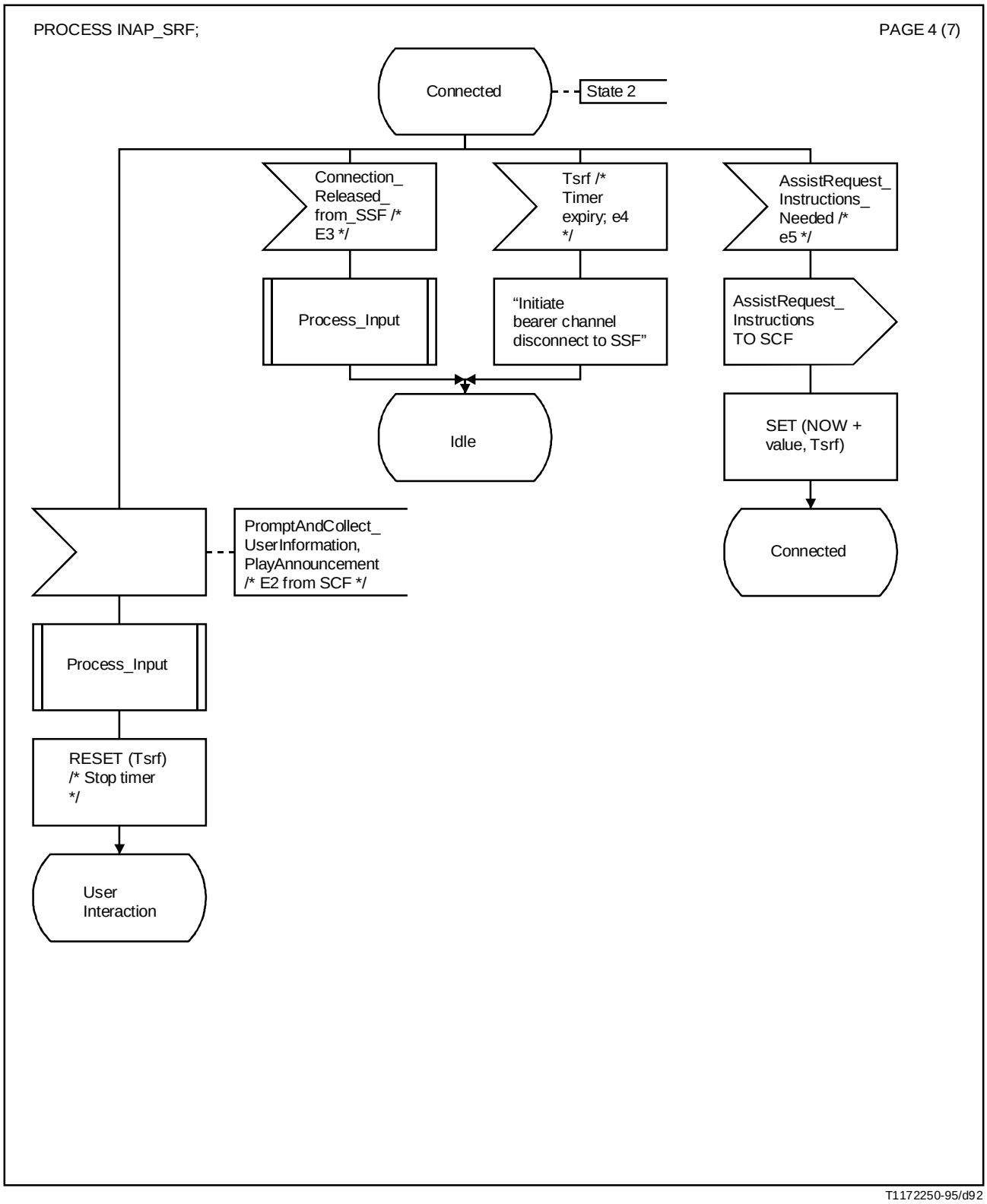


FIGURE A.4/Q.1218 (sheet 4 of 7)

SDL for SRF-FSM

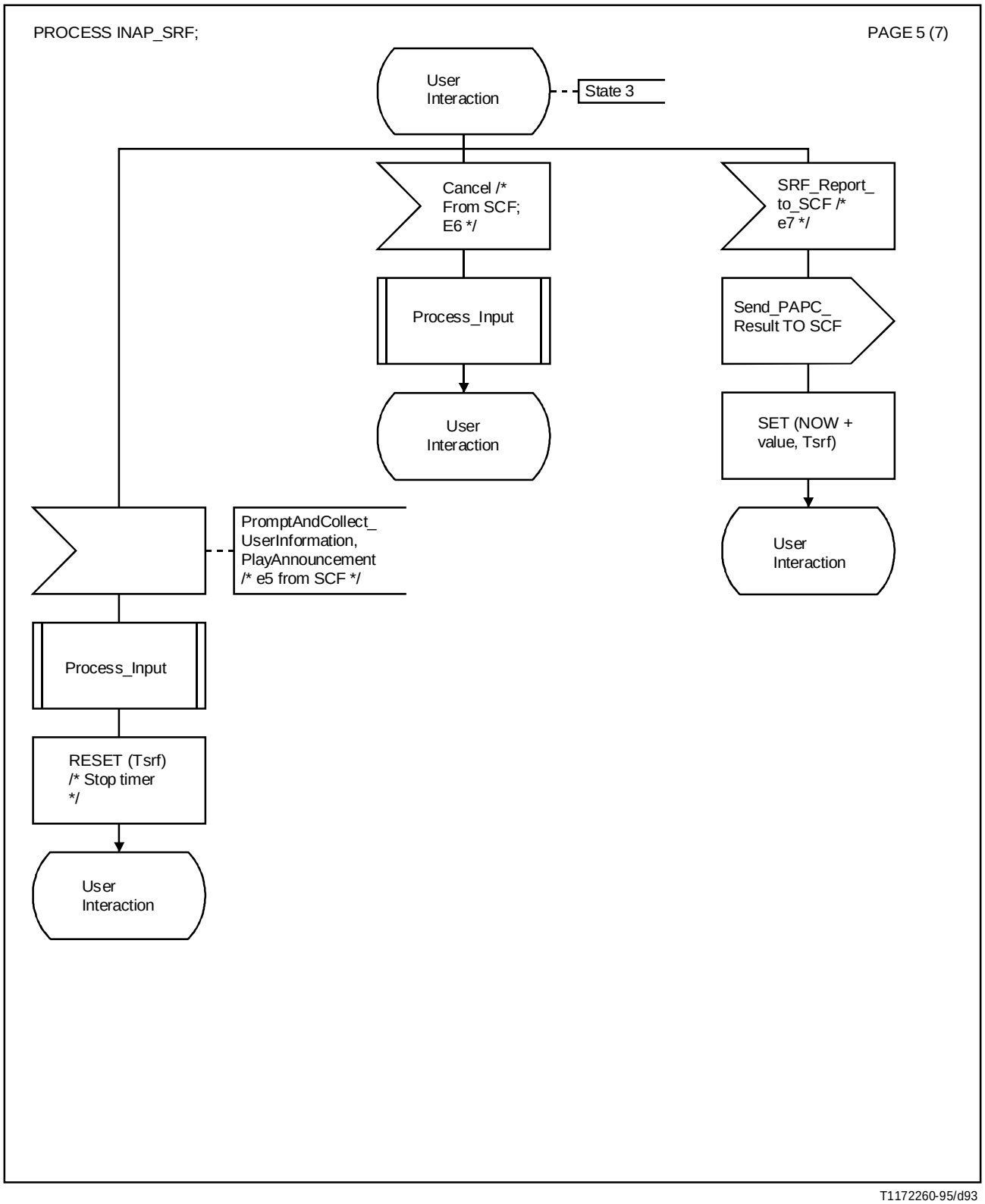


FIGURE A.4/Q.1218 (sheet 5 of 7)

SDL for SRF-FSM

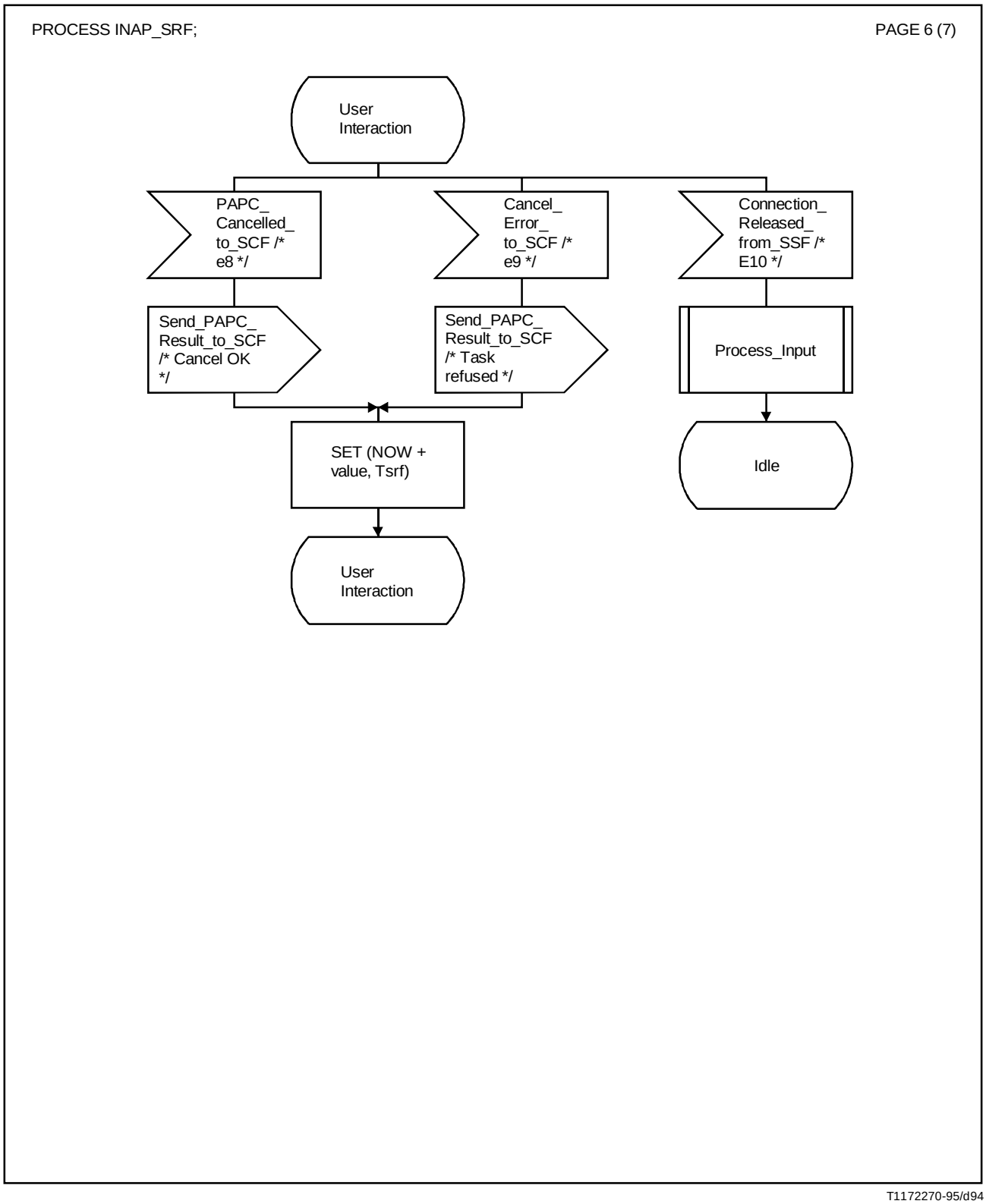
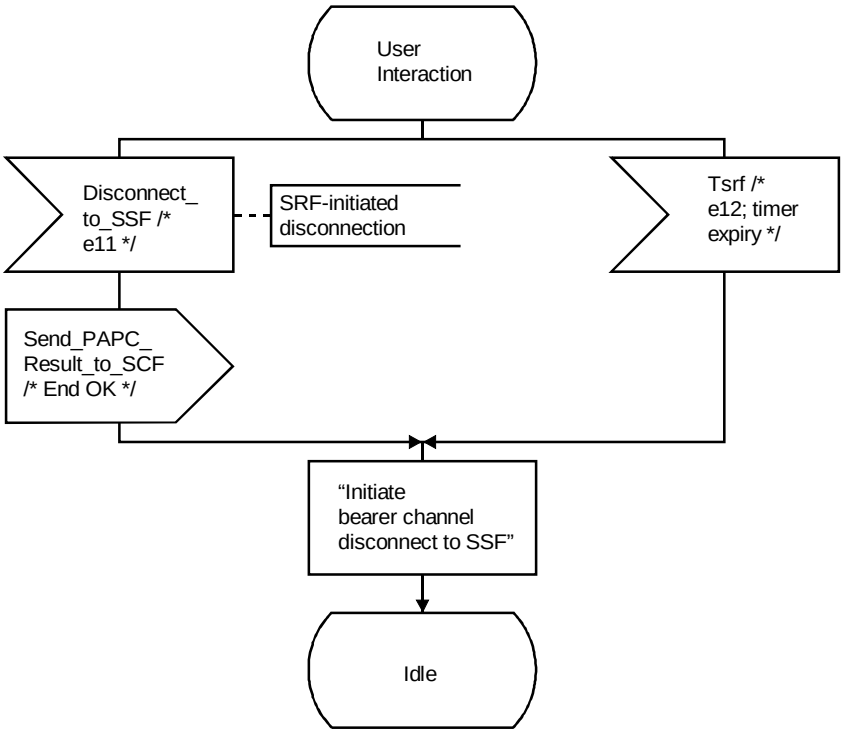


FIGURE A.4/Q.1218 (sheet 6 of 7)

SDL for SRF-FSM



T1172280-95/d95

FIGURE A.4/Q.1218 (sheet 7 of 7)

SDL for SRF-FSM

Annex B

Description of the SCSM (SDF-related states) and of the SDSM

(This annex forms an integral part of this Recommendation)

This description includes a text and an SDL description.

B.1 Description of the Process SCSM

When the Application process at the SCF side decides to create a binding between an SCF and the SDF on behalf of the end user, then a SCSM process instance shall be created by the application process. Two processes occur in the course of establishing a binding namely authentication and version negotiation. Authentication shall be concerned with the user and the directory. The standard shall provide the following methods of authentication techniques namely:

- the use of passwords; or
- the use of protected passwords.

When a SCSM process instance is created it shall be put in the "Idle" state.

B.1.1 State 1 – "Idle"

B.1.1.1 Normal procedures

When the SCSM process instance is in the state "Idle" and a need of the application process to interrogate an SDF exists, a Bind.inv primitive shall be sent by the SCF Application process. The bind argument shall be stored and a timer T_1 shall be started. This event shall cause a transition to the state "Wait_for_Subsequent_Requests" and other events are awaited. The timer T_1 is started in order to supervise the state "Wait_for_Subsequent_Requests" and will ensure that the SCSM process is deleted when the communication with the application process is interrupted. The Dialogue handling services of TC shall be used to support the DirectoryBind operation and to trigger the sending of the associated APDU to the peer SDF process instance.

B.1.1.2 Exceptional procedures

None identified.

B.1.2 State 2 – "Wait for subsequent requests"

B.1.2.1 Normal procedures

In this state, subsequent operations to be sent with the Bind operation (in the same message) to the SDF are expected. The following normal events are considered in this state:

- When a Op.inv primitive is received from the application process a component handling primitive TC-INVOKE req shall be sent to the TC and the process instance shall return to the "Wait_for_Subsequent_Requests" state. This operation shall be sent to the SDF in a message containing a Bind argument. The SCSM process instance shall wait for a Delimiter primitive response from the application process. The operations to be carried by the TC-INVOKE req primitive can be a search, modifyEntry, addEntry or removeEntry.
- When a Delimiter primitive is received from the application process the SCSM process instance shall stop the timer T_1, shall start timer T_2 and shall send a dialogue handling primitive TC-BEGIN req to TC. Once the TC-BEGIN req. primitive is received by the TC, a message containing the Bind argument and other operation arguments if any shall be sent to the SDF. This event shall cause the SCSM process instance to transit to the State 3 "Wait_for_Bind_result". The SCSM process instance shall wait for the response from the SDF. The timer T_2 is started in order to supervise the state "Wait_for_Bind_result" and will ensure that the SCSM process is deleted when the communication with the peer SDF application process is interrupted.

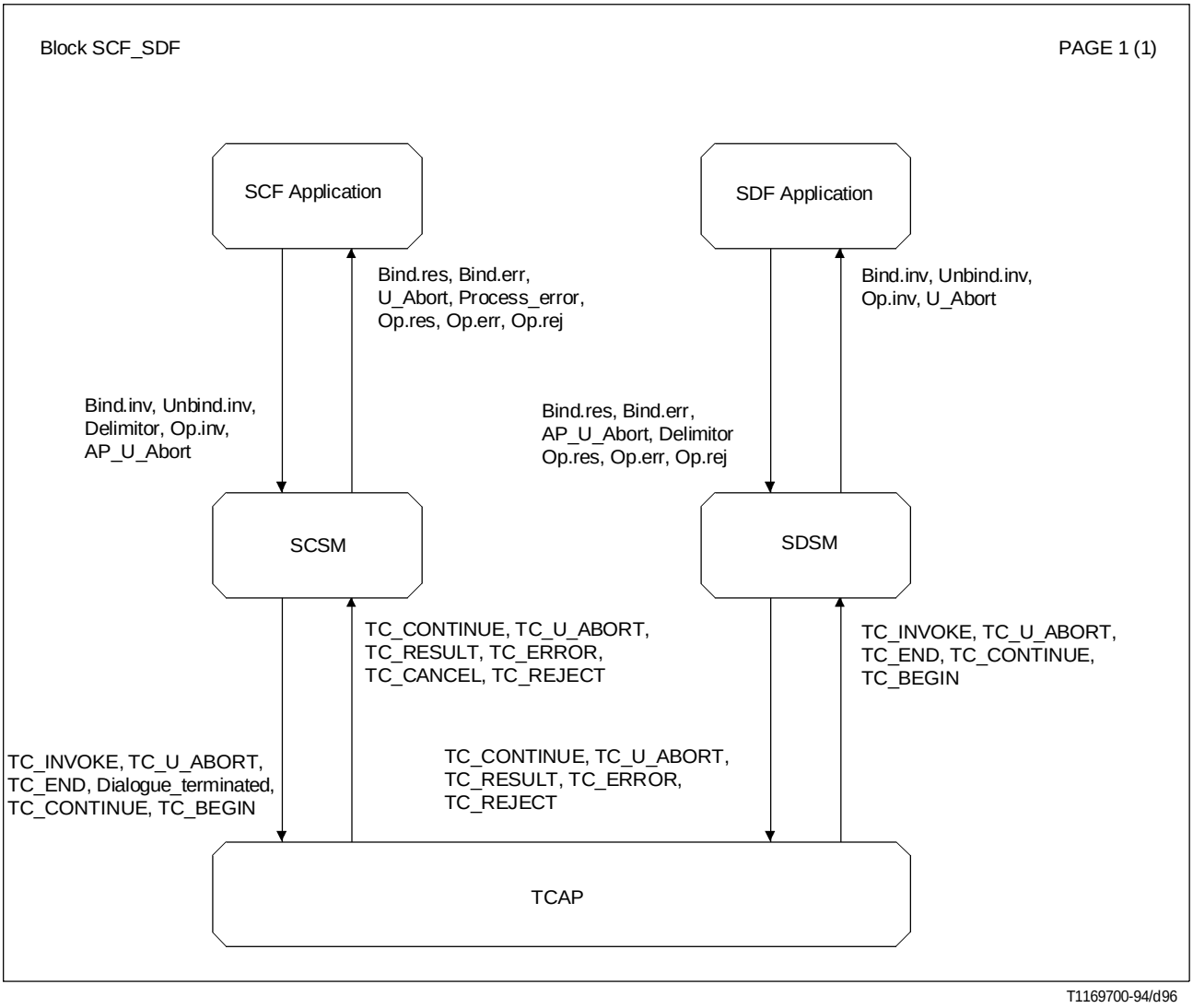


FIGURE B.1/Q.1218

SCF-SDF Model

B.1.2.2 Exceptional procedures

- When the timer T_1 expires and the SCSM process instance is in the "Wait_for_Subsequent_Request" the "Dialogue terminated" primitive shall be sent to the Component Coordination of the TC in order to discard the eventual directory invoke operation stored and awaiting transmission, a "Process_error" primitive shall be sent to the SCF application process and the SCSM process shall be terminated.

- When the SCF application process wants to abort the communication with the peer SDF it shall send a "AP_U_Abort" primitive to SCSM process instance, this process instance shall send a "Dialogue terminated" primitive to the Component Coordination of the TC in order to discard the eventual directory invoke operation stored and awaiting transmission and the SCSM process shall be terminated.

B.1.3 State 3 – "Wait_for_Bind_result"

B.1.3.1 Normal procedures

In this state, the SCF is waiting for the response from the SDF. The reception of the response in the TC-CONTINUE ind primitive from TC to the Bind operation previously issued to the SDF, shall stop the timer T_2, shall send a Bind.res primitive to the application process and shall cause a transition of the SCF to the state "SDF Bound".

B.1.3.2 Exceptional procedures

- When the timer T_2 expires and the SCSM process instance is in the "Wait_for_Bind_result", the "TC-U-ABORT" req primitive shall be sent to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SDF process, a "Process_error" primitive shall be sent to the SCF application process and the SCSM process shall be terminated.

- When the SCF application process wants to abort the communication with the peer SDF it shall send a "AP_U_Abort" primitive to SCSM process instance, this process instance shall send a "TC-U-ABORT" req primitive to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SDF process and the SCSM process shall be terminated.

- When a TC-U-ABORT ind. primitive is received from the TC having a directoryBind error operation component from the peer SDF process, this operation shall be passed in the "Bind.err" primitive to the application process and the SCSM process shall be terminated.

- When a TC-U-ABORT ind. primitive is received from the TC without having a directoryBind error operation component from the peer SDF process, a "U_Abort" primitive shall be passed to the application process and the SCSM process shall be terminated.

B.1.4 State 4 – "SDF Bound"

B.1.4.1 Normal procedures

In this state, the SCF has established an authenticated access to the SDF, and is waiting for requests to the SDF from the service logic or is waiting for responses to the operations previously issued to the SDF:

- When a Op.inv primitive is received from the application process, a component handling primitive TC-INVOKE req shall be sent to the TC and the process instance shall return to the "SDF_Bound" state. The SCSM process instance shall wait for a Delimiter primitive response from the application process. The operations to be carried by the TC-INVOKE req primitive can be a search, modifyEntry, addEntry or removeEntry.

- When a "Delimitor" primitive is received from the application process, then a TC-CONTINUE req primitive will be sent to the TC in order to forward the received operation(s) to the remote SDF application process instance.

- When a TC-RESULT ind primitive is received from the TC caused by the successful reception of a response to the operation previously issued to the SDF, the SCSM process instance shall send the operation return result to the application process and shall return to the same state.

- When a TC-U-ERROR ind primitive is received from the TC indicating that the invoked operation from the SCF application process has failed at the peer SDF side, the SCSM process shall send the operation return error to the application process and shall return to the same state.

- When the TC-L-REJECT ind primitive is received indicating that the TC Component sub-layer received an invalid operation, then the SCSM process instance shall send an operation reject to the application process and shall return to the same state.

- When the TC-R-REJECT ind primitive is received from the TC indicating that the operation was rejected by the remote Component sub-layer, then the SCSM process instance shall send an operation reject to the application process and shall return to the same state.
- When the TC-U-REJECT ind primitive is received from the TC indicating that the operation was rejected by the remote application process instance, then the SCSM process instance shall send an operation reject to the application process and shall return to the same state.
- When the TC-L-CANCEL ind primitive is received from the TC indicating that the operation invocation was locally terminated due to a time-out condition, then the SCSM process instance shall inform the application process and shall return to the same state.
- When an Unbind.inv primitive is received from the SCF application process caused by the need to terminate the authenticated access to the SDF, a TC-END req primitive shall be sent to the TC in order to end the dialogue with the remote SDF application process and the SCSM process instance shall be terminated.

B.1.4.2 Exceptional procedures

- When the SCF application process wants to abort the communication with the peer SDF, it shall send a "AP_U_Abort" primitive to SCSM process instance, this process instance shall send a "TC-U-ABORT" req primitive to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SDF process and the SCSM process shall be terminated.
- When a TC-U-ABORT ind. primitive is received from the TC, a "U_Abort" primitive shall be passed to the application process and the SCSM process shall be terminated.

B.2 Description of the Process SDSM

The SDF directory abstract service is defined in terms of a number of operations which the SDF can perform at the request of the SCF on behalf of the end user. Before invoking these operations, the SCF shall perform a bind operation at some access point. The SCF can then invoke operations, each invocation shall identify the particular operation to be performed and shall carry the arguments which define the actual request.

B.2.1 State 1 – "Idle"

B.2.1.1 Normal procedures

When the Dialogue handling services of TC sends a TC-BEGIN ind. containing the DirectoryBind operation, a SDSM process instance shall be created, a "Bind.inv" primitive shall be sent to the application process and timer T_3 shall be started. This event shall cause a transition to the state "Bind pending". The timer T_3 is started in order to supervise the state "Bind pending" and shall ensure that the SDSM process is deleted when the communication with the application process is interrupted.

B.2.1.2 Exceptional procedures

None identified.

B.2.2 State 2 – "Bind Pending"

B.2.2.1 Normal procedures

In this state, a Bind request has been received from the SCF. The SDF is performing the SCF access control procedures behind the Bind operation (e.g. access authentication). There may also be a case such that the Bind operation is a dummy one. Then, the access authentication is not required. Two events are considered in this state:

- When the TC-INVOKE ind. component handling primitive is received caused by the reception of operations before the result from the Bind operation is sent from the application process, a "Op.inv" primitive shall be sent to the application process and the SDSM process instance shall return to the "Bind pending" state.
- When the bind request succeeds, the SDF application process shall send a "Bind.res" primitive to the SDSM process instance containing the directoryBind return result. On receipt of the "Bind res" primitive the SDSM process instance shall stop timer T_3, store the information locally until a "Delimitor" primitive is received from the application process and shall transit to the "SCF bound" state.

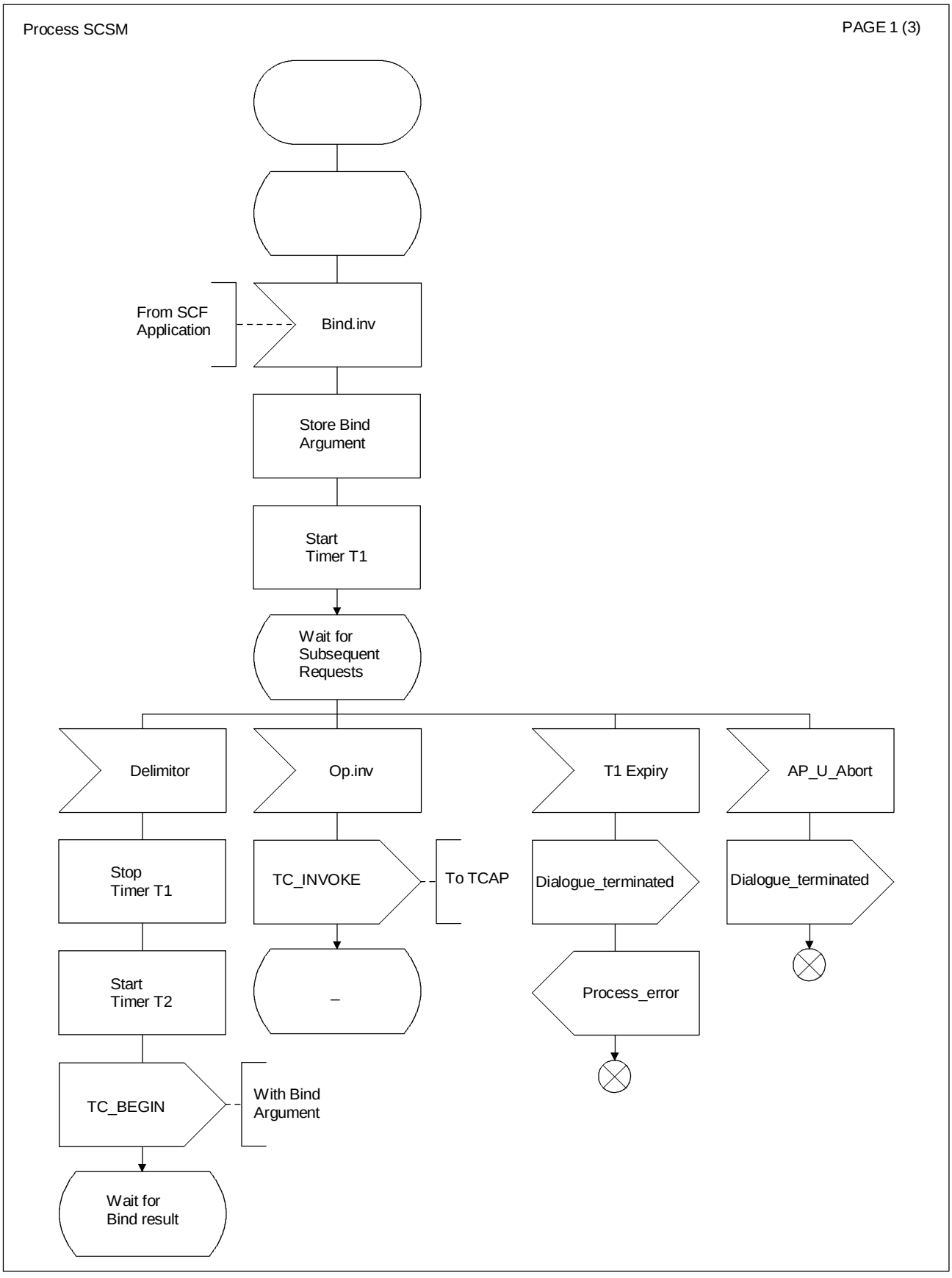


FIGURE B.2/Q.1218 (sheet 1 of 3)

SCSM

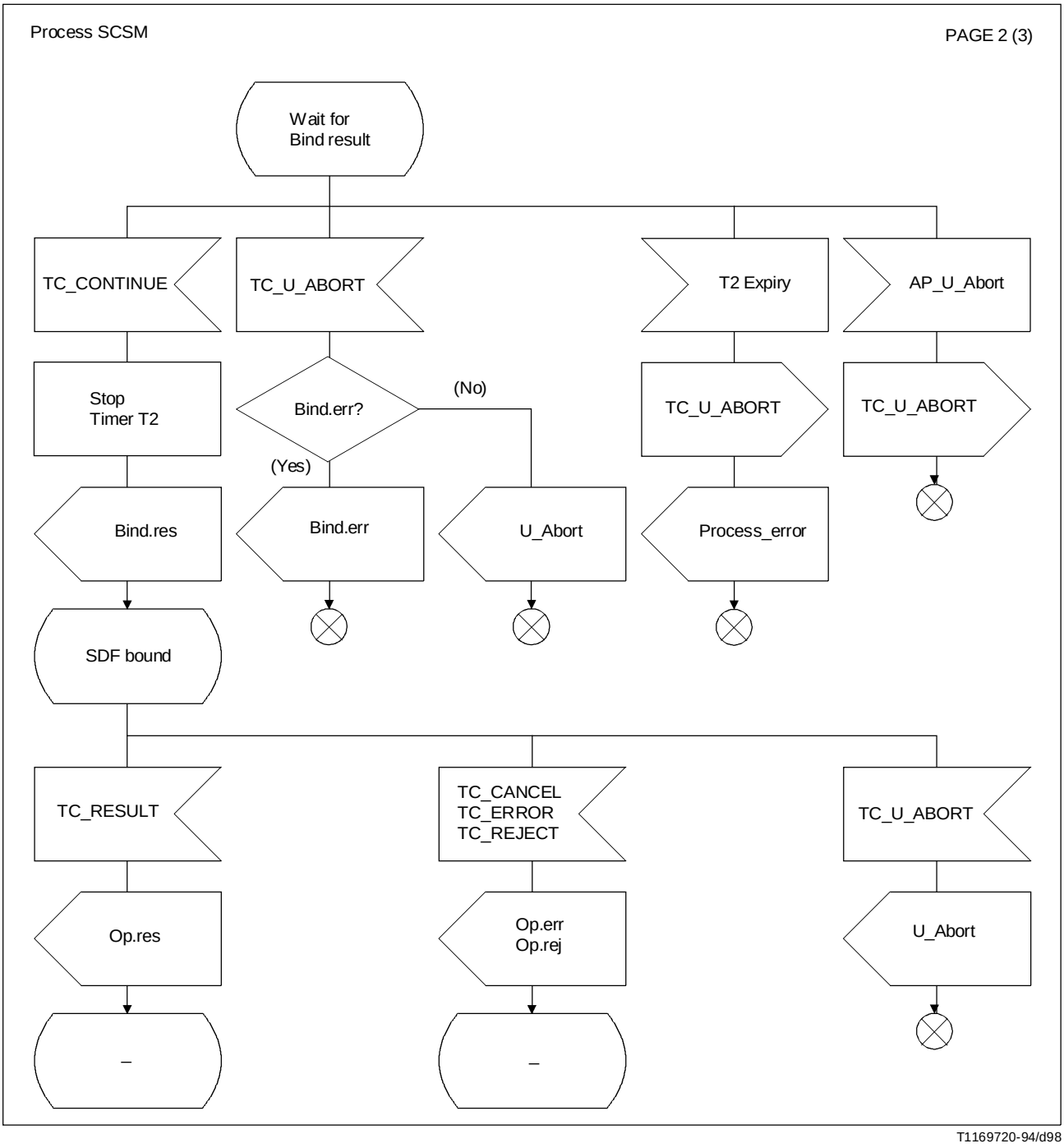


FIGURE B.2/Q.1218 (sheet 2 of 3)

SCSM

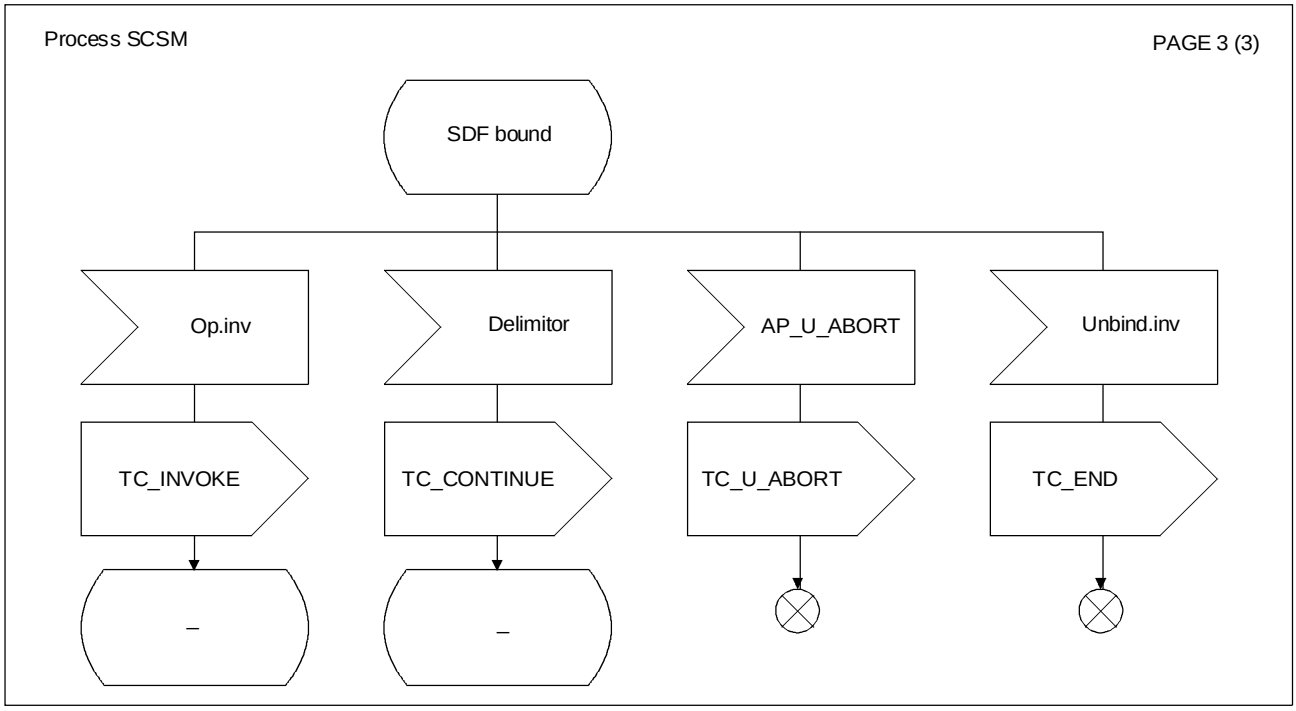


FIGURE B.2/Q.1218 (sheet 3 of 3)

SCSM

B.2.2.2 Exceptional procedures

- When the bind request fails, the application process shall send a "Bind.err" primitive to the SDSM process instance containing the directoryBind errors. The SDSM process instance shall send a "TC-U-ABORT" req primitive to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SCF process and the SDSM process shall be terminated.

- When the timer T_3 expires and the SDSM process instance is in the "Bind_pending" state, the "TC-U-ABORT" req primitive shall be sent to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SCF process, a "Process_error" primitive shall be sent to the SDF application process and the SDSM process shall be terminated.

- When the SDF application process wants to abort the communication with the peer SCF it shall send a "AP_U_Abort" primitive to SDSM process instance, this process instance shall send a "TC-U-ABORT" req primitive to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SCF process and the SDSM process shall be terminated.

B.2.3 State 3 – "SCF Bound"

B.2.3.1 Normal procedures

In this state, the access of the SCF to the SDF is authorized and operations coming from the SCF are accepted. Besides waiting for requests from the SCF, the SDF can also send in that state responses to previously issued operations. The following normal events are considered in this state:

- When a Op.res primitive is received from the application process indicating that an operation is successfully executed, a component handling primitive TC-RESULT req shall be sent to the TC and the process instance shall return to the "SCF bound" state. The SDSM process instance shall wait for a Delimiter primitive response from the application process to send a TC-CONTINUE req primitive to the TC Dialogue Handling entity so that the TC can forward this return result to the peer SCSM process instance. The return results to be carried by the TC-RESULT req primitive can be originated from previously received search, modifyEntry, addEntry or removeEntry operations.

- When a Op.err primitive is received from the application process indicating that an previously received operation cannot be successfully executed, a component handling primitive TC-U-ERROR req shall be sent to the TC and the process instance shall return to the "SDF bound" state. The SDSM process instance shall wait for a Delimiter primitive response from the application process to send a TC-CONTINUE req primitive to the TC Dialogue Handling entity so that the TC can forward this return error to the peer SCSM process instance. The return errors to be carried by the TC-U-ERROR req primitive can be originated from previously received search, modifyEntry, addEntry or removeEntry operations.

- When a Delimitor primitive is received from the application process, the SDSM process instance shall send a dialogue handling primitive TC-CONTINUE req to TC. Once the TC-CONTINUE req. primitive is received by the TC, a message containing the Bind return result, return error or reject argument previously stored, if not already sent, and other operation return result, return error or reject arguments if any shall be sent to the SCF. This event shall cause the SDSM process instance to return to the State "SCF bound".

- When a TC-END ind primitive is received from the Dialogue Handling of the TC caused by the reception of the Unbind operation from the SCF, the SDSM process instance shall send a "Unbind.inv" primitive to the application process and shall terminate the SDSM process instance. The SCF-SDF association shall be ended and all associated resources are released.

- When a TC-INVOKE ind component handling primitive is received from the TC a "Op.inv" primitive shall be sent to the application process and the SDSM process instance shall return to the "SCF bound" state.

B.2.3.2 Exceptional procedures

- When a Op.rej primitive is received from the application process indicating that an previously received operation is rejected, a component handling primitive TC-U-REJECT req shall be sent to the TC and the process instance shall return to the "SCF bound" state. The SDSM process instance shall wait for a Delimiter primitive response from the application process to send a TC-U-CONTINUE req primitive to the TC Dialogue Handling entity so that the TC can forward this reject to the peer SCSM process instance. The reject to be carried by the TC-U-REJECT req primitive can be originated from previously received search, modifyEntry, addEntry or removeEntry operations.

- When a TC-U-ABORT ind. primitive is received from the TC a "U_Abort" primitive shall be passed to the application process and the SDSM process shall be terminated.

- When the SDF application process wants to abort the communication with the peer SCF it shall send a "AP_U_Abort" primitive to SDSM process instance, this process instance shall send a "TC-U-ABORT" req primitive to the Dialogue Handling of the TC in order to abort the started dialogue with the peer SCF process and the SDSM process shall be terminated.

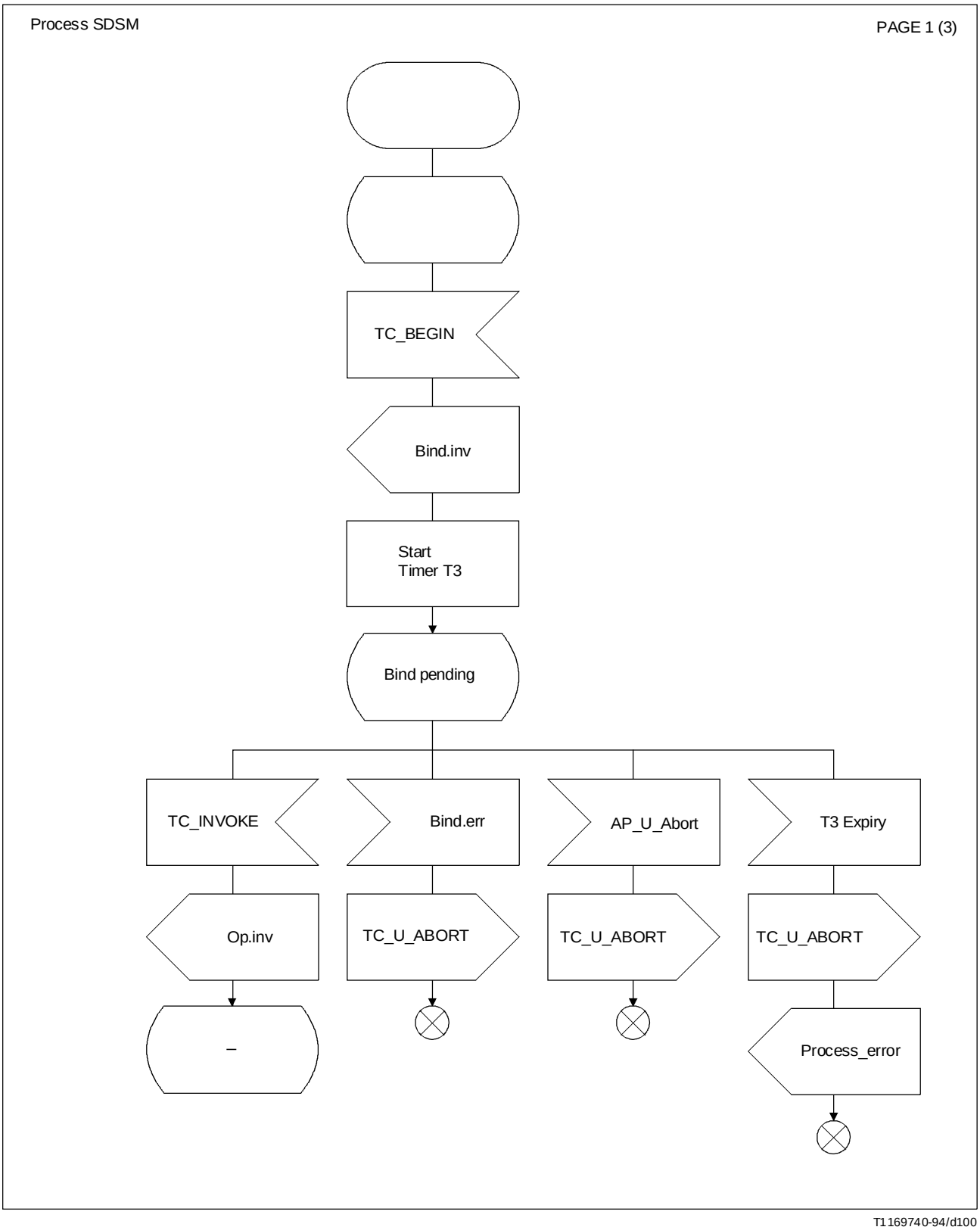


FIGURE B.3/Q.1218 (sheet 1 of 3)

SDSM

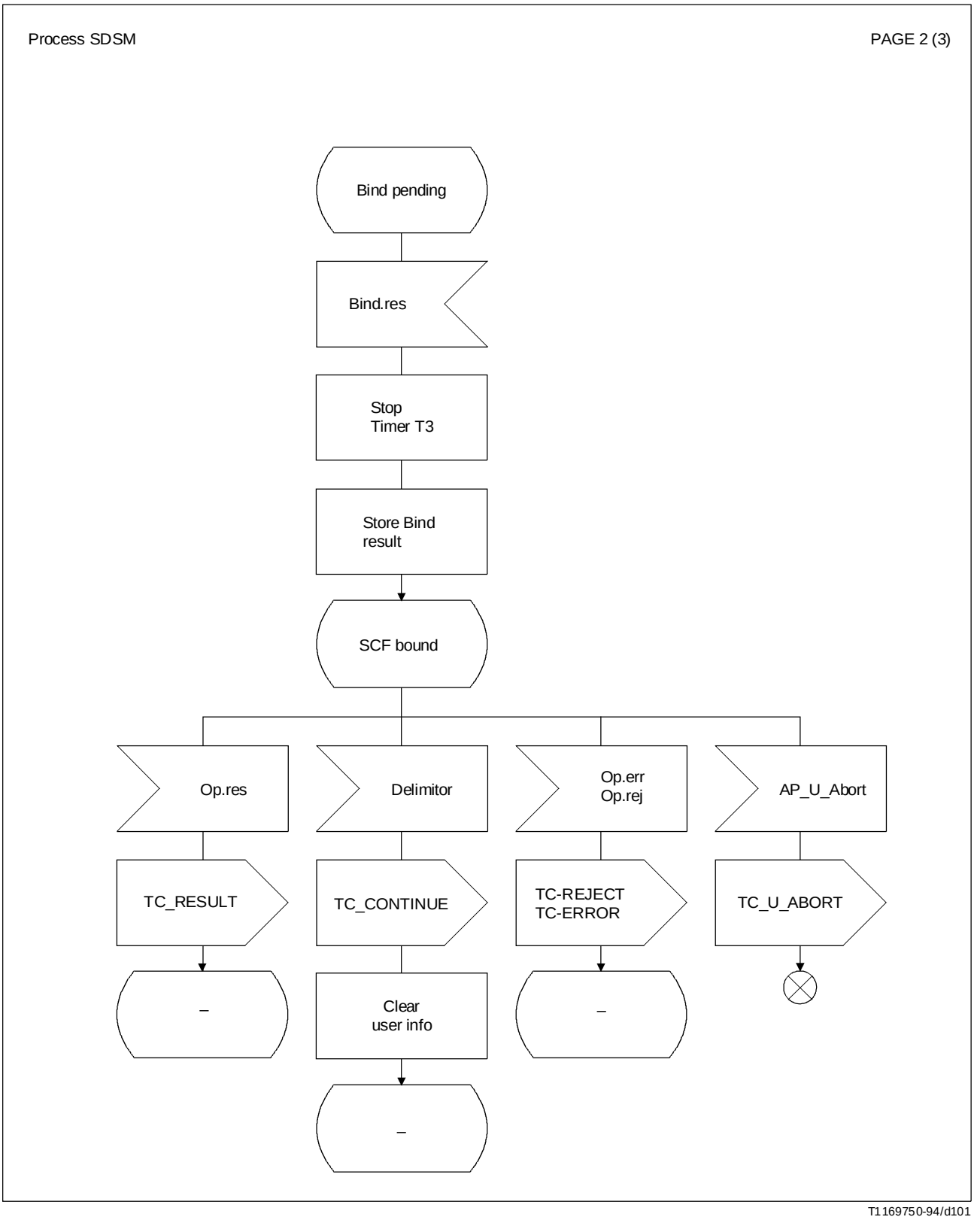


FIGURE B.3/Q.1218 (sheet 2 of 3)

SDSM

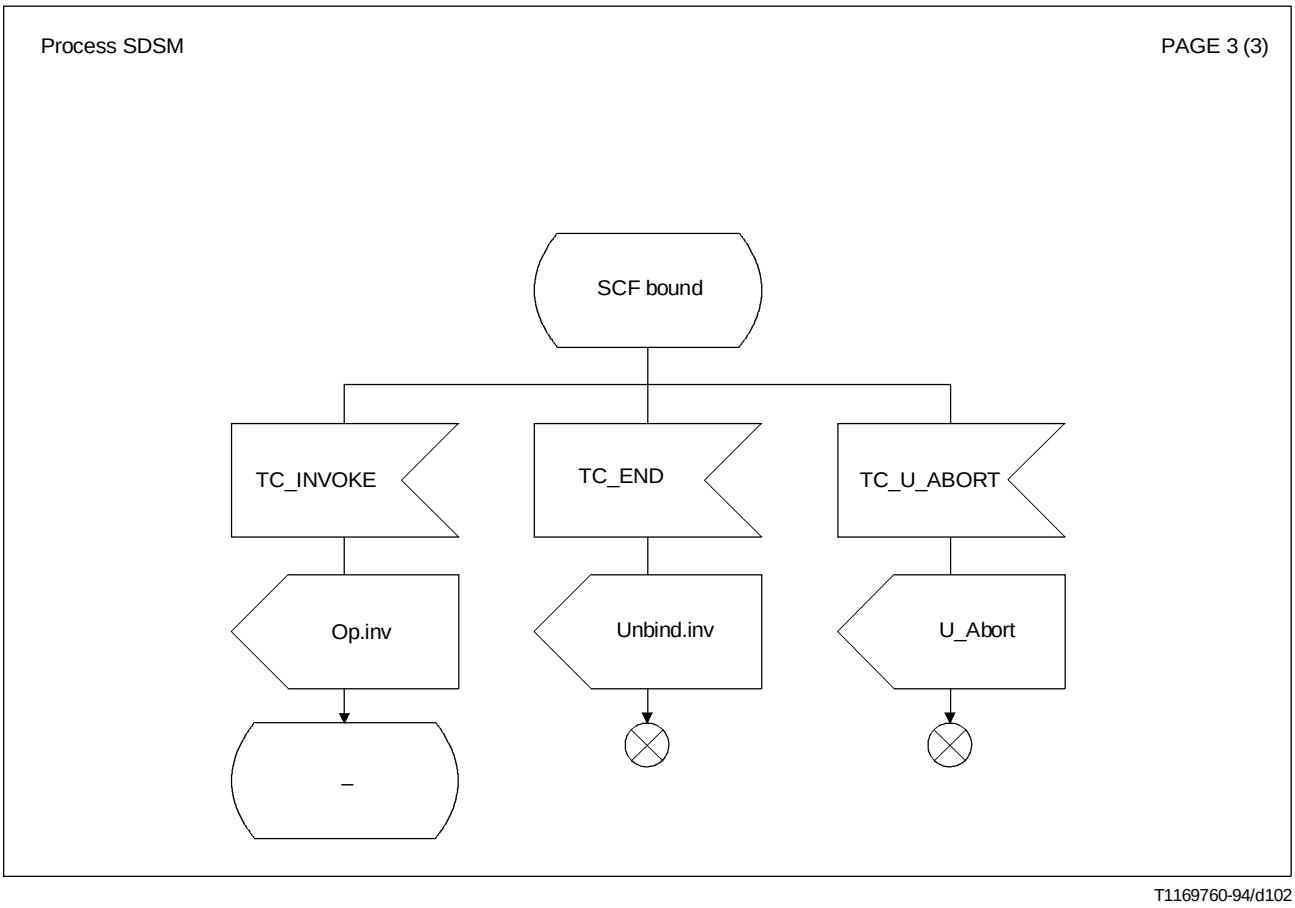


FIGURE B.3/Q.1218 (sheet 3 of 3)

SDSM

Appendix I

Service data modelling

(This appendix does not form an integral part of this Recommendation)

This appendix outlines the required steps and provides a set of guidelines and examples for the definition of a service data model and the mapping of it onto the information model defined in 2.2.2.3. These guidelines and examples provide a simple mechanism which addresses the items in I.1.1 and I.2.1.

Additional tools (e.g. attribute context specifications) may also provide solutions to data modelling, however this appendix does not address these tools.

I.1 Review of need for Service data modelling

The adoption of the X.501 information model as the data model for representing service data within the SDF, together with the use of X.511 operations for the interface between the SCF and SDF, has opened up the question of how IN Service data should be mapped onto this information model.

Inappropriate mapping of service data onto the X.500 information model used within the SDF can dramatically increase the amount of information which must be transferred across the SCF-SDF interface (in certain cases even exceeding the capacity of the TCAP link). In addition to higher interface traffic, the choice of mapping can also affect the amount of data processing which must be performed by the Functional Element Actions (FEA) within the SCF, to the point of making the SCF-FEA implementations dependent on the service data.

This subclause outlines a simple set of mapping rules which can be applied to service data to both minimize SCF-SDF interface traffic and to maintain service-independent handling of data.

I.1.1 General modelling issues

There are three general requirements to keep in mind when organizing the IN Service data into the X.500 information model. Each of these requirements should be given equal weight.

– **Req-1** – Minimize the traffic between the SCF and the SDF.

  The SCF-SDF interface will probably be one of the most heavily used interfaces within the IN environment. Each SLPI within the SCF may have to initiate multiple accesses to data stored in the SDF. It is crucial from a performance point of view that the data transfers be kept to a minimum.

– **Req-2** – Minimize the data processing to be done within the SCF.

  Data processing within the SCF is performed by the FEAs defined in Recommendation Q.1214 to handle the Information flows. These FEAs are assumed to be independent of the service for which they are invoked. If the data information being transferred is of the wrong granularity, then service dependent processing may be performed by the FEAs within the SCF. Care must be taken in the design of the X.500 information model for the IN service Data to maintain the service independency of the FEA.

– **Req-3** – Consider the effect of data distribution on the X.500 data management.

  For CS-1 systems, the SDF implementation may not be distributed. It should be noted that X.500 systems are inherently distributed data management systems. If the design of the structure of the DIT does not take the distribution of data into account, then the resulting Directory may be unmanageable in terms of the "knowledge" management required to access the Directory information across multiple Directory Service Agents (DSA).

  While it can be argued that a centralized Directory implementation need not take distribution into account, performance requirements may require the splitting of the Directory over a number of local DSAs, requiring distribution knowledge to be maintained within each DSA.

I.1.2 Service modelling – Defining the information required

Service modelling is defined in Recommendation Q.1203. It consists of the following steps:

1) identifying the service features which comprise the service;

2) identifying the data required to support the needed service feature;

3) identifying the shared data between the service features used by the service;

4) identify the operations and processing required on the service data;

5) identify the Access Control requirements for the service data.

These steps are not covered in this Recommendation. Only the additional steps required to map the Service Data onto the SDF Information Model defined in 2.2 are covered here.

I.2 Guidelines for building Service Data Models

Once the IN Service Data Model has been defined in terms of the service feature data, it must be mapped onto the X.500 information model. This can be done using the following steps:

1) selection of ATTRIBUTEs;
2) selection of OBJECT-CLASSs;
3) designing the Service DIT;
4) assignment of Access Control to ATTRIBUTEs and OBJECT-CLASSs.

I.2.1 Selection of ATTRIBUTEs

The selection of the ATTRIBUTEs is affected by the operations and processing required by the SCF. The following rules are provided to assist the selection process:

- **Guideline 1** – ATTRIBUTEs are the smallest addressable unit of IN Service Data which may be transferred across the SCF-SDF interface for processing by the SCF. Structured data types should be avoided in ATTRIBUTE definitions.
- **Guideline 2** – All the ATTRIBUTEs in the Directory which are to be retrieved by the IN service are by preference to be SINGLE-VALUED.
- **Guideline 3** – If Service Data contains multi-column tables of information (see Figure I.1), then each column[3), 4)] of the table should have an ATTRIBUTE definition.

  For example, in Figure I.1a) (Time of Day Routing table) three ATTRIBUTEs are defined (StartTime, StopTime and Destination). For Figure I.1b) (Abbreviated Number table) two ATTRIBUTEs are defined (AbbrevNo and Destination).

After all ATTRIBUTEs have been identified it may be possible to reduce the total number of ATTRIBUTEs defined due to identical definitions (e.g. Destination in Figure I.1).

Start Time	Stop Time	Destination
01:00	09:00	204 23456
09:01	15:00	204 12345
15:01	23:00	204 23456

a) Time of Day Routing table

Abbreviated No.	Destination
01	204 23456
02	204 12345
03	204 23456

b) Abbreviated Number table

FIGURE I.1/Q.1218

Multi-Column Tables in Service Data

3) This assumes that the tables are organized as in Figure I.1 with each row in the table representing a separate group of information. If this organization is reversed, with the columns containing the item groups then each row in the table should have an ATTRIBUTE definition.

4) This guideline is normally used in conjunction with Guideline 6 to determine entry items in the DIT.

I.2.1.1 ATTRIBUTE Type definitions

An ATTRIBUTE's type is defined by an ASN.1 data definition.

```
thisAttribute  ATTRIBUTE ::= {
        ..
        WITH SYNTAX thisAttributeASN1DataType,
        ..
        }
```

An ATTRIBUTE can be defined as a SUB-TYPE of another ATTRIBUTE which means that it must conform to the definition of the parent ATTRIBUTE.

```
thisAttribute  ATTRIBUTE ::= {
        ..
        SUBTYPE OF anotherAttribute,
        ..
        }
```

While this provides a convenient way to re-use attribute definitions it can have a side-effect which will impact on Req-1. This is because an X.500 Search operation for a specific AttributeType will return not only the required Attribute's values but also the Attribute Values of those Attributes which are a sub-type of the required Attribute.

It should be noted that Searching for a particular Attribute will not return any of the parent Attribute values.

An ATTRIBUTE can be defined as a Collective Attribute which implies that its values can be shared amongst a number of Directory Entries.

```
collectiveAttribute  ATTRIBUTE ::= {
        ..
        COLLECTIVE              TRUE,
        ..
        }
```

Examples of pre-defined attribute can be found in clause 2/X.520.

I.2.1.2 Defining Permitted Values

There are two methods which can be used to define permitted values for an attribute.

These are:

1) restrict the values to a specific set of values,
2) restrict the values to a range on numeric values.

The first case requires two additional attributes:

a) an attribute to hold the permitted values for the current attribute,

b) an Access Control attribute linking the two attributes together. This attribute must have a syntax of ACIItem and contain a ProtectedItem field containing the following:

```
ProtectedItem ::= SEQUENCE {
        ..
        restrictedBy    [10]    SET OF RestrictedValue OPTIONAL
        ..
        }
```

The RestrictedValue contents contain the Attribute Id's of the current Attribute and the Attribute containing the permited values. Since the RestrictedValue value is part of the Access Control attribute, the ability to change the RestrictedValue value implies the ability to modify all the Access control parameters for the current Entry contained within the Access Control Attribute.

The second case requires only one additional attribute:

c) an Access Control attribute holding the value range for the current attribute. This attribute must have a syntax of ACTItem type and contain a ProtectedItem field containing the following:

```
ProtectedItem ::= SEQUENCE {
        ..
        rangeOfValues   [7]     Filter OPTIONAL
        ..
        }
```

The Filter contents contain the value range specification the Attribute values. Note that to change one part of the value range requires replacing the entire Filter. Also since the Filter value is part of the Access Control attribute, the ability to change the Filter value implies the ability to modify all the Access control parameters for the current Entry contained within the Access Control attribute.

I.2.1.3 Defining Default Values

The default for an attribute type is set at execution time by specifying an Attribute value with a context whose Fallback field is set to TRUE.

Attribute values whose context fallback values are set to TRUE will only be returned if no other attribute values that match the criteria are present.

I.2.1.4 Defining MAX Attribute Values

Limiting the number of values which an Attribute may have, is done by specifying an Access Control Attribute containing a ProtectedItem field with the following syntax:

```
ProtectedItem ::= SEQUENCE {
        ..
        maxValueCount       [8]   SET OF MaxValueCount OPTIONAL
        ..
        }
```

The MaxValueCount contents contain the Attribute Id and the maximum numer of values which can be stored in the Attribute. Since the MaxValueCount value is part of the Access Control attribute, the ability to change the MaxValueCount value implies the ability to modify all the Access control parameters for the current Entry contained within the Access Control Attribute.

I.2.1.5 Defining Attribute Lifetimes

Attribute lifetimes are set by supplying a TemporalContext for the Attribute Value (at execution time).

I.2.1.6 Definition of MATCHING-RULES

MATCHING-RULES define how Attribute values are tested against external values for use in search Filters and Name Matching. An ATTRIBUTE definition has three different types of Matching Rules:

EQUALITY for testing (value = = attribute Value).

ORDERING for testing (value < attribute Value).

SUBSTRINGS for testing sub-strings matches.

```
thisAttribute  ATTRIBUTE ::= {
        ..
        EQUALITY MATCHING RULE equalityMatchingRuleName,
        ORDERING MATCHING RULE orderingMatchingRuleName,
        SUBSTRINGS MATCHING RULE substringsMatchingRuleName,
        ..
        }
```

Examples of predefined matching-rules can be found on clause 3/X.520.

I.2.1.7 Assigning the ATTRIBUTE Object Identifiers

Each ATTRIBUTE defined is assigned a unique Object Identifier. The form of this Object Identifier should be as follows:

```
-- IN object identifier prefix

in-oi       OBJECT IDENTIFIER ::= { ccitt recommendation q 1218 ... }
-- service attribute definitions

attributeX      OBJECT IDENTIFIER ::=       { in-oi ServiceID attributeType(4) AttributeID }
-- ServiceID defines the service/service feature to which the attribute belongs.
-- AttributeID defines the Attribute ID within the service: this is assigned by the Service
-- designer.
```

I.2.2 Selection of OBJECT-CLASSs

Once the ATTRIBUTEs used within the Service Data Model have been identified, then they can be grouped together into specific OBJECT-CLASS elements.

I.2.2.1 Defining the OBJECT-CLASS

An OBJECT-CLASS defines the specification of a Directory Entry. It consists of a grouping of ATTRIBUTEs into a single object class. Each instance of a Directory Entry within the DIT has a unique identity (see 6.2.2.2 for specification). The following rules can be used to assist the OBJECT-CLASS definition:

- **Guideline 4** – Service data values which can share a common unique identity should be grouped as ATTRIBUTEs within a Directory Entry (e.g. Service Data, User Data for a service).

- **Guideline 5** – Service data values which are part of a multi-column Table (see Guideline 3) should be grouped into a single OBJECT-CLASS definition.

An OBJECT-CLASS can be specified as being a SUB-CLASS of another OBJECT-CLASS. In this case the OBJECT-CLASS contains all of the attributes and properties of the parent OBJECT-CLASS.

This is similar to inheritance in C++.

I.2.2.2 Selecting the RDN (NAME-FORM)

Each OBJECT-CLASS can have a number of attributes designated as possible RDN attributes. This is done using the NAME-FORM specification. When the Directorey Entry is created one of these must be designated the active RDN (through the Distinguished Name defined in the AddEntry operation). After the object has been created, only the active RDN can be used in the name matching procedures for locating an object using the access operations (Search, ModifyEntry, RemoveEntry, ...).

Therefore if two objects of the same OBJECT-CLASS are created with the same ATTRIBUTE values but using different active RDNs, then they are considered as two distinct objects within the DIT. If the same RDN had been used for the second object then an object, creation error would have occurred.

I.2.2.3 Assigning the OBJECT-CLASS Object Identifiers

Each OBJECT-CLASS defined is assigned a unique Object Identifier. The form of this Object Identifier should be as follows:

-- *IN object identifier prefix*

in-oi OBJECT IDENTIFIER ::= { ccitt recommendation q 1218 ... }

-- *service object-class definitions*

objectClassX OBJECT IDENTIFIER ::= { in-oi ServiceID objectClass(6) ObjectClassID }

-- *ServiceID defines the service/service feature to which the attribute belongs.*

-- *ObjectClassID defines the Object Class ID within the service: this is assigned by the Service designer.*

I.2.3 Defining the DIT

This subclause describes how to build the DIT for an IN Service Data model.

I.2.3.1 Defining the Object Hierarchy (STRUCTURE-RULES)

The DIT defines a hierarchy of data objects. Each Distinguished name of an object consists of the concatenation of the RDN of the object with the RDN's of the objects in the DIT above the object. The hierarchy of objects within the DIT is defined using STRUCTURE-RULEs. Figure I.2 shows how a set of OBJECT-CLASS and STRUCTURE-RULEs form the DIT.

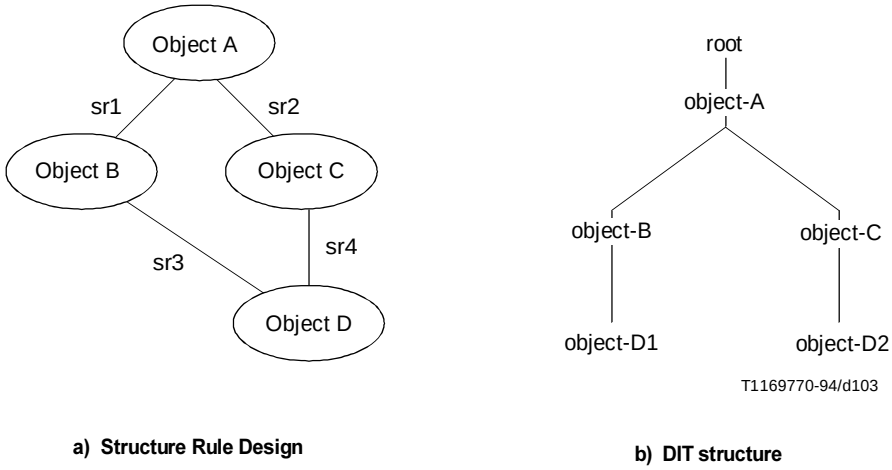


FIGURE I.2/Q.1218

Mapping of STRUCTURE-RULES onto DIT

The following set of rules can be used to build the DIT from the set of OBJECT-CLASS specifications:

- **Guideline 6** – Where Service Data values share a common structure, but may have multiple value sets (e.g. rows in a time-of-day translation table, user-specific data within a service) these are to be placed in Directory Entries which are subordinate to another Directory Entry (i.e. there is a STRUCTURE-RULE defining the relationship between the Directory Entries).

- **Guideline 7** – Where multiple sets of Data Entries exist under a specific Data Entry, it is recommended that each of the sets be placed as subordinates to a single data entry which is subordinate to the specific data entry (see Figure I.3). This will allow more efficient searching of the set data entries using the X.500 operations.

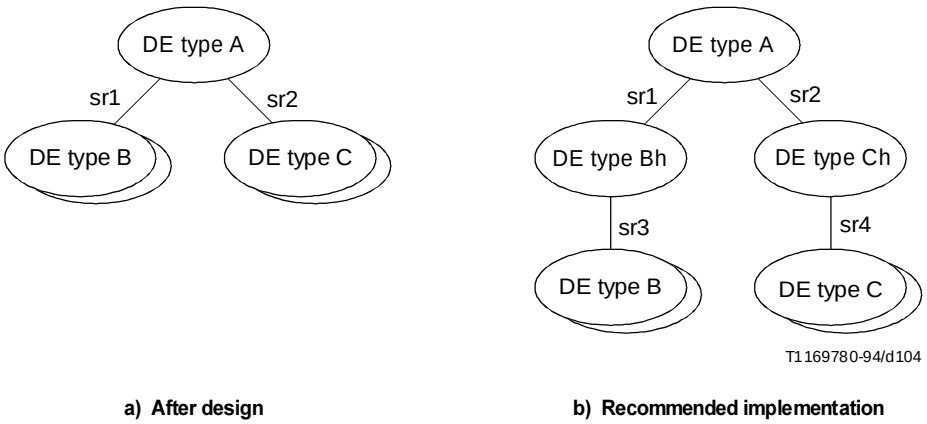


FIGURE I.3/Q.1218

Multiple Set recommended model structure

I.2.3.2 Defining alternative naming paths (Aliases)

Once an object class has been instanciated its Distinguished Name is fixed. If an alternative naming path to the object is required, then an Alias object must be defined within the DIT. This Alias object will have its own RDN attributes and will contain a pointer to the object being referenced.

An Alias object is an OBJECT-CLASS which is a subtype of the predefined alias OBJECT-CLASS.

I.2.3.3 Defining MAX number of subordinate Entries

Limiting the number of subordinate entries an Entry may have, is done by specifying an Access Control Attribute within the Entry containing a ProtectedItem field with the following syntax;

```
ProtectedItem ::= SEQUENCE {
        ..
        maxImmSub          [8]   INTEGER OPTIONAL,
        ..
        }
```

The maxImmSub value contains the maximum number of subordinate entries which may be stored below the current Entry. Note that this limit only applies to the current DSA. Since the maxImmSub value is part of the Access Control attribute, the ability to change the maxImmSub value implies the ability to modify all the Access Control parameters for the Entry contained in the Access Control Attribute.

I.3 Example Service (UPT Like Service)

This subclause implements an X.500 DIT for a simplified subset of a UPT service using the mapping rules.

I.3.1 Example Service description

The example service contains the following service features:

1) incall;
2) incall Registration;
3) incall Screening;
4) outcall;
5) outcall Screening by location.

These features are described briefly below together with the data information which would be available after the service design stage (ref. I.1.2). The information for each service feature is presented in table form with the following columns:

– Data: The name of the data item.

– Definition: The definition of the data item type (using ASN.1 style notation).

– Cardinality: This defines how different versions of the data item can occur (e.g. one for each user of the service).

– Feature Operation: This defines what operations (SCF-SDF interface) are used to manipulate the data item.

– Access Control: This defines what level of access is required by the different levels of service operators.

NOTE – This is not intended to be a complete service. Only sufficient detail is provided to show how mapping's onto the SDF Information Model should be performed.

I.3.1.1 Service Feature – Incall

Normal Incall operation performs the following routing algorithm:

1) if the user has a registered location active, then send call to registered location;
2) if active time-of-day location is active, then route to time-of-day location;
3) if the user has default location, then route to default location;
4) reject call with announcement "user is not available on system".

All defined locations must be one of a pre-defined set of locations for the user.

Service Feature Information

Data	Definition	Cardinality	Feature Operations	Security Access Control
Registered Locations	SET OF Location	One per service user	None	Service: R User management: R/W User: R

Time Of Day Location	SEQUENCE {
	startTime SEQUENCE {
	day DayOfWeek,
	time TimeOfDay },
	stopTime SEQUENCE {
	day DayOfWeek,
	time TimeOfDay }
	location Location }
		Many per service user	Search	Service: R
				User management: R/W
				User: R

Current Location	Location	One per user	Search	Service: R User: R/W
Current Location Expiry	TimeOfDay	One per user	None	Service: R User: R/W
Default Expiry	TimeOfDay	One per user	None	Service: R User management: R/W User: R
Default Location	Location	One per user	Search	Service: R User management: R/W User: R

I.3.1.2 Service Feature – Incall Registration

This service feature allows the user to register his current location. The current location can either be set to the current terminal (identified by CLI or a defined location). The user can optionally set an expiry time for the current location. If no expiry time is set, a default expiry time is used. The current location must be one of a predefined set of locations.

Service Feature Information

Data	Definition	Cardinality	Feature Operations	Security Access Control
Registered Location	SET OF Location	One per service user	None	Service: R User management: R/W User: R
Current Location	Location	One per user	modifyEntry	Service: R User: R/W
Current Location Expiry	TimeOfDay	One per user	modifyEntry	Service: R User: R/W
Default Expiry	TimeOfDay	One per user	None	Service: R User management: R/W User: R

I.3.1.3 Service Feature – Incall Screening

This service feature allows the user to reject an incoming call if it is from a specific location. There is a limit to the number of locations which can be used for screening.

Service Feature Information

Data	Definition	Cardinality	Feature Operations	Security Access Control
Incall Screen List	SET OF Location SIZE (1..MaxListSize)	One per service user	Search	Service: R User management: R/W User: R
MaxListSize	INTEGER	One per service user	None	Service: R Service management: R/W

I.3.1.4 Service Feature – Outcall

This service feature allows a user to make an outgoing call from a terminal location. The location must be one of the permitted locations for the user. The charge for the call is billed to the service user, not the owner of the terminal.

Service Feature Information

Data	Definition	Cardinality	Feature Operations	Security Access Control
Registered Locations	SET OF Location	One per service user	Search	Service: R User management: R/W User: R
User Id	ServiceNumber	One per service user	match/bind	Service management: R/W Service: R User management: R User: R
User Password	Password	One per service user	match/bind	Service: R User management: R/W User: R

I.3.1.5 Service Feature – Outcall screening by location

This service feature provides for preventing outgoing calls to specific locations from proceeding. The list of prescribed locations is different for each location the user can use.

Service Feature Information

Data	Definition	Cardinality	Feature Operations	Security Access Control

Outcall
Screen List
	SET OF {
	SEQUENCE {
	stopList SET OF Location,
	location Location }}
		One per service user	Search	User management: R/W
				User: R

I.3.1.6 General Service Information

The service features have the following relationships for interaction:

- Incall screening is done before Incall;
- Outcall is done before Outcall Screening by location.

Additional Service Information

The following additional service parameters are provided:

Data	Definition	Cardinality	Feature Operations	Security Access Control
Service Key	ServiceId	One per service	match/bind	Service management: R/W Service: R User management: R User: R
User Name	String	One per service user	match	User management: R/W User: R

I.3.2 Example Service data modelling

Now that the service data information has been defined, the mapping of the data onto the information model using the steps defined in subclause I.2 can begin.

I.3.2.1 Selection of Attributes

The first step is to determine the ATTRIBUTE definitions required for the Service Data. For the purposes of defining attribute identifiers, the following object identifier is used:

exampleAttribute OBJECT IDENTIFIER ::= { in-oi exampleService(1) attributeType(4) }

Data – Registered Locations

The Registered Location stores the Locations at which a user may register to receive incoming calls.

Registered Locations	SET OF Location	One per service user	None	Service: R User management: R/W User: R

For the purposes of this example the definition of a Location can be considered to be equivalent to a telephone number. This allows the use of SUBTYPE definitions for the service attributes which use Location type definitions from a pre-defined attribute in Recommendation X.520 (telephoneNumber). The definition of the Registered Location attribute therefore becomes:

```
regLocations ATTRIBUTE ::= {
        SUBTYPE OF telephoneNumber,
        COLLECTIVE TRUE,
        ID          in-at-regLocations }
```

The attribute is defined as a Collective Attribute since its values will be shared among a number of Entries in the DIT (refer to I.3.3.2)

Data – Time Of Day Location

The Time Of Day Location is actually a translation table consisting of 5 columns (startTime.day, startTime.time, stopTime.day, stopTime.time, location).

Time Of Day Location	SEQUENCE {
	startTime SEQUENCE {
	day DayOfWeek,
	time TimeOfDay },
	stopTime SEQUENCE {
	day DayOfWeek,
	time TimeOfDay }
	location Location }
		Many per service user	Search	Service: R
				User management: R/W
				User: R

Using Guideline 3, 5 different ATTRIBUTEs are defined, one for each column:

```
startDay  ATTRIBUTE ::= {
        WITH SYNTAX          DaysOfWeek                    -- from proposed X.520 Temporal Context
        SINGLE VALUE         TRUE
        ID                   in-at-startDay }

startTime  ATTRIBUTE ::= {
        WITH SYNTAX                     DayTime                    -- from proposed X.520 Temporal Context
        EQUALITY MATCHING RULE          numericStringMatch         -- from X.520
        ORDERING MATCHING RULE          numericStringOrderingMatch -- from X.520
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch -- from X.520
        SINGLE VALUE                    TRUE
        ID                              in-at-startTime }

stopDay  ATTRIBUTE ::= {
        WITH SYNTAX          DaysOfWeek                    -- from proposed X.520 Temporal Context
        SINGLE VALUE         TRUE
        ID                   in-at-stopDay }

stopTime  ATTRIBUTE ::= {
        WITH SYNTAX                     DayTime                    -- from proposed X.520 Temporal Context
        EQUALITY MATCHING RULE          numericStringMatch         -- from X.520
        ORDERING MATCHING RULE          numericStringOrderingMatch -- from X.520
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch -- from X.520
        SINGLE VALUE                    TRUE
        ID                              in-at-stopTime }

location  ATTRIBUTE ::= {
        SUBTYPE OF           telephoneNumber
        SINGLE VALUE         TRUE
        ID                   in-at-location }
```

This Attribute requires additional set-up to restricts it's values to those contained in the Attribute regLocations. This will be done later with an Access Control Attribute in the entry containing both regLocations and location.

Data – Current Location

The Current Location defines the registered location for a user. It has a specific lifetime.

Current Location	Location	One per user	Search	Service: R User: R/W

The definition of the Current Location attribute therefore becomes:

```
currentLocation  ATTRIBUTE ::= {
        SUBTYPE OF           telephoneNumber
        SINGLE VALUE         TRUE
        ID                   in-at-cur-location }
```

This Attrribute requires additional set-up to restrict its values to those contained in the Attribute regLocations. This will be done later with an Access Control Attribute in the entry containing both regLocations and currentLocation.

The lifetime for this attribute is specified by providing the value with a temporal context with the endTime field set to the value of either Current Location Expiry or Default Expiry. The selection of which endTime to use is the responsability of the function setting the attribute value[5)].

Data – Current Location Expiry

The Current Location Expiry data contains the expiration time for the current location data.

Current Location Expiry	TimeOfDay	One per user	None	Service: User:	R R/W

Since the Current Expiry data is only used for setting the lifetime of the currentLocation Atribute, this data is not stored as a separate Attribute within the Directory. Instead, it is present as a TemporalContext value attached to the value of currentLocation. Note that this is an optionally supplied value from the User. If it is not supplied, then the value of Default Expiry is used instead. This selection is managed by the SCF-FEA responsible for managing the Attribute.

Data – Default Expiry

The Default Expiry data contains the default expiry time for the current location expiry data.

Default Expiry	TimeOfDay	One per user	None	Service: User management: User:	R R/W R

Since the Default Expiry data is only used as a default value for setting the lifetime of the currentLocation Atribute, this data is not stored as a separate Attribute within the Directory. Instead, it is present as a TemporalContext value attached to the value of currentLocation. Note that this value is only used if the User does not supply a value for Current Location Expiry. This selection is managed by the SCF-FEA responsible for managing the Attribute.

Data – Default Location

The Default Location defines the location to use if all other routing options fail.

Default Location	Location	One per user	Search	Service: User management: User:	R R/W R

The definition of the Default Location attribute therefore becomes:

```
defaultLocation  ATTRIBUTE ::= {
        SUBTYPE OF      telephoneNumber
        SINGLE VALUE    TRUE
        ID              in-at-def-location }
```

5) While this may appear to require Service dependent information in the SCF-FEA responsible for managing the Attribute, the lifetime concept is sufficiently general (i.e. can be applied to a large number of attributes independently) that a standard mechanism can be developed to handle it in a service independent manner.

This Attribute requires additional set-up to restricts it's values to those contained in the Attribute **regLocations**. This will be done later with an Access Control Attribute in the entry containing both **regLocations** and **defaultLocation**.

Data – Incall Screen List

The Incall Screen List defines a list of Locations, of specified maximum length, from which calls to the user are to be rejected.

Incall Screen List	SET OF Location SIZE 1..MaxListSize	One per service user	Search	Service: R User management: R/W User: R

The Incall Screen List can be considered in two ways, as a table with a single column or as an attribute with multiple values. In this case, since the operation only tests to see if specific values are present and does not retrieve the value, the multi-valued attribute version can be safely used.

The definition of the Incall Screen List attribute therefore becomes:

```
incallScreenList  ATTRIBUTE ::= {
        SUBTYPE OF      telephoneNumber,
        ID              in-at-incall-screen }
```

This attribute requires additional set-up to restrict the number of values which can be stored in the Attribute. This will be done later with an Access Control Attribute in the entry containing IncallScreenList.

Data – MaxListSize

The MaxListSize data holds the maximum length of the Incall Screen List.

MaxListSize	INTEGER	One per service user	None	Service: R Service management: R/W

Since the MaxListSize data is only used as a value for the maximum number of values of the **IncallScreenList** Attribute, this data is not stored as a separate Attribute within the Directory. Instead, it is present as a Protected Item of type maxValueCount in an Access Control Attribute in the entry containing **IncallScreenList**.

Data – User Id

The User_Id data defines the number used to identify the user of a service. When coupled to a service prefix (e.g. 014) it forms a Location.

User Id	ServiceNumber	One per service user	match/bind	Service management: R/W Service: R User management: R User: R

The definition of the User Id attribute therefore becomes:

```
userId  ATTRIBUTE ::= {
        WITH SYNTAX                     NumericString
        EQUALITY MATCHING RULE          numericStringMatch              -- from X.520
        ORDERING MATCHING RULE          numericStringOrderingMatch      -- from X.520
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch    -- from X.520
        SINGLE VALUE                    TRUE
        ID                              in-at-user-id }
```

Data – User Password

The User Password holds the PIN number of a user. It is used to validate registration and Outcall information.

User Password	Password	One per service user	match/bind	Service: R User management: R/W User: R

Since this attribute is used in the Bind operation, the User Password must use the userPassword attribute definition defined in 6.3/X.509.

Data – Outcall Screen List

The Outcall Screen List provides a list of barred locations for each registered location the user has. It is a table with two columns (regLocation, stopList).

Outcall Screen List	SET OF {
	SEQUENCE {
	stopList SET OF Location,
	location Location }}
		One per service user	Search	User management: R/W
				User: R

Two attributes are defined for the Outcall Screen List:

outcallScreenLocation ATTRIBUTE ::= {
SUBTYPE OF telephoneNumber
SINGLE VALUE TRUE
ID in-at-outcall-location }

This Attribute requires additional set-up to restrict its values to those contained in the Attribute **regLocations**. This will be done later with an Access Control Attribute in the entry containing both **regLocations** and **outcallScreenLocation**.

outcallScreenStopList ATTRIBUTE ::= {
SUBTYPE OF telephoneNumber
ID in-at-outcall-stoplist }

Data – Service Key

The Service Key data identifies the Service being used.

Service Key	ServiceId	One per service	match/bind	Service management: R/W Service: R User management: R User: R

The definition of the Service Key attribute therefore becomes:

serviceKey ATTRIBUTE ::= {
WITH SYNTAX Object Identifier
SINGLE VALUE TRUE
ID in-at-service-key }

Data – User Name

The User Name data hold the name of the User of the Service.

User Name	String	One per service user	match	User management: R/W User: R

The definition of the User Name attribute therefore becomes:

```
userName  ATTRIBUTE ::= {
          SUBTYPE OF name,                          -- from X.520
          ID              in-at-user-name }
```

I.3.2.2 Selection of Object Classes

Once the ATTRIBUTEs have been selected, they can be organized into OBJECT-CLASSs. Using Guideline 5, the following attributes can be grouped together (attributes in [] denotes referenced attributes):

- startDay, startTime, stopDay, stopTime, location, [regLocations];
- outcallScreenLocation, outcallScreenStopList, [regLocations].

The remaining attributes must be grouped using Guideline-4. Since attributes which share the same cardinality can be said to share the same identity, the following groupings are used:

- serviceKey;
- currentLocation, defaultLocation, incallScreenList, userId, userPassword, userName, [regLocations].

Attributes can be mandatory or optional within an object-class. From the description of the service logic it can be determined that the currentLocation, defaultLocation attributes are optional. Also, any attribute which is grouped as part of a table must be a mandatory attribute. Other attributes should be made mandatory as well.

The OBJECT-CLASS definitions required are:

```
serviceClass  OBJECT-CLASS      ::=     {
          MUST CONTAIN          serviceKey
          ID                    in-oc-service }

serviceUserClass  OBJECT-CLASS  ::=     {
          MUST CONTAIN          userId, userPassword, userName, incallScreenList, reqLocations,
          MAY CONTAIN           currentLocation, defaultLocation
          ID                    in-o-service-user }

timeOfDayLocation  OBJECT-CLASS ::=     {
          MUST CONTAIN          startDay, startTime, stopDay, stopTime, location, reqLocations
          ID                    in-oc-time-location}

outcallScreenList  OBJECT-CLASS ::=     {
          MUST CONTAIN          outcallScreenLocation, outcallScreenStopList, reqLocations
          ID                    in-oc-outcall-screen}
```

The definition of the Access Control Attributes used to handle Attribute Access Control has been deferred until the DIT design is complete (see I.3.2.4).

Once the object classes have been defined, the naming attributes for the objects must be assigned. This requires the identification of one of the mandatory attributes which has a unique value for each occurrence of the object.

Examining the objects defined, the following naming attributes can be used:

serviceClass	**serviceKey**
serviceUserClass	**userId**
timeOfDayLocation	**? (Nothing useable)**
outcallScreenList	**outcallScreenLocation**

The only object class which has a naming problem is the timeOfDayLocation object. There are two ways around this problem. The first is to define a unique attribute (for example tableRowIndex) to allow naming. This does not affect performance of operations and can be useful in the management operation which are not being considered here.

The second solution is to combine the StartDay, StartTime attributes into a composite attribute, the values of which would then be unique. This method would involve redefining MATCHING-RULEs and some extra processing.

For simplicity the first solution is chosen. The required name forms, with new attribute and revised object-class, are:

tableRowIndex ATTRIBUTE ::= {
WITH SYNTAX Integer
SINGLE VALUE TRUE
ID in-at-maxListValues }

timeOfDayLocation OBJECT-CLASS ::= {
MUST CONTAIN tableRowIndex, startDay, startTime, stopDay, stopTime, location, regLocations
ID in-oc-time-location }

serviceName NAME-FORM ::= {
NAMES serviceClass
WITH ATTRIBUTES serviceKey
ID in-nf-service-name }

serviceUserName NAME-FORM ::= {
NAMES serviceUserClass
WITH ATTRIBUTES userId
ID in-nf-service-user-name }

timeOfDayLocationName NAME-FORM ::= {
NAMES timeOfDayLocation
WITH ATTRIBUTES tableRowIndex
ID in-nf-tod-location-name }

outcallScreenName NAME-FORM ::= {
NAMES outcallScreenList
WITH ATTRIBUTES outcallScreenLocation
ID in-nf-outcall-screen-name }

I.3.2.3 Designing the Service DIT

At this point the design of the DIT can begin. Looking at the cardinality information, the following OBJECT-CLASS relationships can be derived (see Figure I.4):

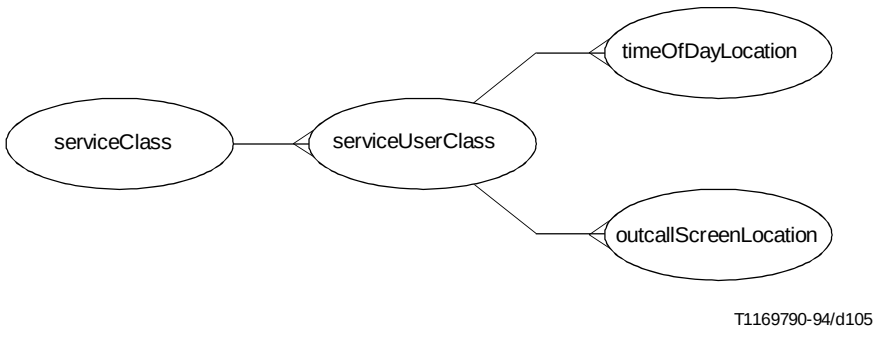


FIGURE I.4/Q.1218

Object-Class Relationships

Since serviceUserClass has two entry sets below it, Guideline 7 should be invoked which results in adding an extra object level, as shown in Figure I.5:

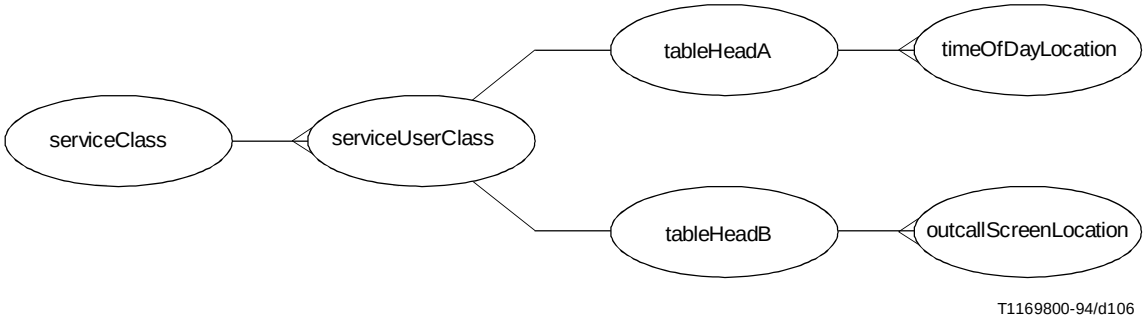


FIGURE I.5/Q.1218

Object-Class Relationships after Guideline 7

This requires additional object class and attribute definitions. These are:

tableHeadIdentifier ATTRIBUTE ::= {
WITH SYNTAX **Integer**
SINGLE VALUE TRUE
ID **oi-gen-table-head }**

tableHead OBJECT-CLASS ::= {
MUST CONTAIN **tableHeadIdentifier**
ID **in-oc-tableHead }**

tableHeadA OBJECT-CLASS ::= {
MUST CONTAIN **tableHeadIdentifier**
ID **in-oc-tableHeadA }**

tableHeadA NAME-FORM ::= {
NAMES **tableHeadA**
WITH ATTRIBUTEs **tableHeadIdentifier**
ID **in-nf-tableHeadNameA }**

tableHeadB OBJECT-CLASS ::= {
MUST CONTAIN **tableHeadIdentifier**
ID **in-oc-tableHeadB }**

tableHeadNameB NAME-FORM ::= {
NAMES **tableHeadB**
WITH ATTRIBUTEs **tableHeadIdentifier**
ID **in-nf-tableHeadNameB }**

Therefore the structure rules required to define the DIT structure are:

sr-root STRUCTURE-RULE ::= {
NAME FORM **serviceClassName**
ID **sr0 }**

sr-user STRUCTURE-RULE ::= {
NAME FORM serviceClassName
SUPERIOR RULES sr0
ID sr1 }

sr-tableA STRUCTURE-RULE ::= {
NAME FORM tableHeadNameA
SUPERIOR RULES sr1
ID sr2 }

sr-tod-location STRUCTURE-RULE ::= {
NAME FORM timeOfDayLocationName
SUPERIOR RULES sr2
ID sr3 }

sr-tableB STRUCTURE-RULE ::= {
NAME FORM tableHeadNameB
SUPERIOR RULES sr1
ID sr4 }

sr-outcall-screen STRUCTURE-RULE ::= {
NAME FORM outcallScreenListName
SUPERIOR RULES sr4
ID sr5 }

The creation of an Entry of OBJECT-CLASS subEntry holds the specification of the sub-tree to which the collective attributes apply. For the case of the regLocations collective attribute the subEntry is subordinate to an Entry of the OBJECT-CLASS serviceUserClass.

To modify the regLocation Attribute requires the Distinguished Name of the subEntry to be supplied (see Figure I.6).

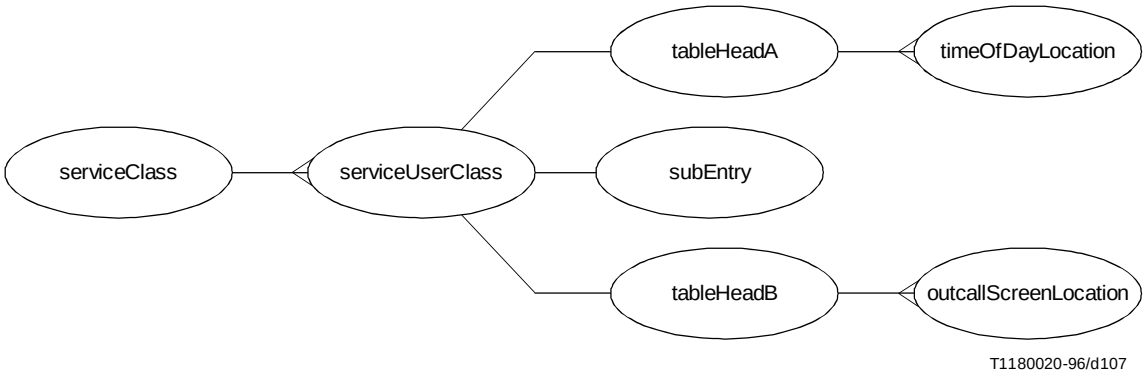


FIGURE I.6/Q.1218

Directory Tree showing sub-entry location

I.3.2.4 Assigning Access control

Note that the DIT is complete, the next required step is to define the Access Control Attribute values for each of the Entries in the DIT. An Entry's Access Control Attribute (entryACI, see 16.5.2/X.501) contains values of type ACIItem which control how the user may access the Attribute values. The following table defines the Access Control required for each of the attributes defined within the DIT.

Object	Attribute	UserClasses defined by X.500 Management					Special Protected Items
		User-X	AllUsers	User Mgt	Service	Service Mgt	
serviceClass	serviceKey	+R+F	+F	+R+F	+R+F	+R+A+D+F	
serviceUserClass	userId	+R+F	+F	+R+F	+R+F	+R+A+D+F	
	userPassword	+R+F+C		+R+A+D+F	+R+F	+R+F	
	userName	+R+F+C		+R+A+D+F	+R+F	+R+F	
	incallScreenList	+R+F		+R+A+D+F(*)	+R+F	+R+F	maxValueCount
	currentLocation	+R+A+D+F(*)		+R+F	+R+F	+R+F	restrictedBy
	defaultLocation	+R+A+D+F(*)		+R+F	+R+F	+R+F	restrictedBy
tableHeadA	tableHead Identifier	+R+F		+R+F	+R+F	+R+A+D+F	
timeOfDay Location	tableRowIndex	+R+F		+R+A+D+F	+R+F	+R+F	
	startDay	+R+F		+R+A+D+F	+R+F	+R+F	
	startTime	+R+F		+R+A+D+F	+R+F	+R+F	
	stopDay	+R+F		+R+A+D+F	+R+F	+R+F	
	stopTime	+R+F		+R+A+D+F	+R+F	+R+F	
	location	+R+F		+R+A+D+F(*)	+R+F	+R+F	restrictedBy
tableHeadB	tableHead_ Identifier	+R+F		+R+F	+R+F	+R+A+D+F	
outcallScreenList	outcallScreen_ Location	+R+F		+R+A+D+F(*)	+R+F	+R+F	restrictedBy
	outcallScreen_ StopList	+R+F		+R+A+D+F	+R+F	+R+F	
serviceUser_ Subentry	regLocations	+R+F		+R+A+D+F	+R+F	+R+F	

+	Grant permission	A	Add attribute to entry	M	Modify attribute
–	Deny permission	R	Read access	C	Compare attribute
*	Special protected item	D	Remove access	F	Filter match attribute

Each table column under UserClasses denotes a potential ACIItem value for the entryACI attribute. In practice the values can be combined in a single ACIItem which combines a set of protected items which have the same access permissions for a set of UserClasses. Each ACIItem has a unique identifier field, which can be used to apply Access Control to individual values of the Access Control Attribute itself. This allows different "Users" to modify different Access Control values (which may contain permitted value or size limits) without being able to change other values of the Access Control attribute.

Where permissions are marked with a (*) this indicates that a special Protected Item of the type specified in the last column (Special Protected Items) must be included in the ACIItem to handle permitted value or value count limit controls.

I.4 Defining Contexts

This subclause defines a set of Context types which may be found useful across of range on IN services.

I.4.1 Numerical Index Attribute Value Context

The Numerical Index Attribute Value Context associates a numerical index with an attribute value.

```
numericalIndexAVC       ATTRIBUTE-VALUE-CONTEXT ::= {
        SYNTAX          INTEGER
        id              id-avc-numericalIndex }
```

An example of use of the numerical index attribute value context is an "abbreviated to expanded number" conversion table, where the index (the context value) is the abbreviated number and the attribute value the expanded number.

Note that a similar String Index Attribute Value Context type could also be defined.

Appendix II

Aspects of the intelligent network interface identified as "For Further Study" (FFS) relative to CS-1

(This appendix does not form an integral part of this Recommendation)

II.1 General

II.1.1 General consideration

This appendix includes call party handling and other issues which were considered to be incomplete when developing the Q.1218 intelligent network interface Recommendation CS-1. Although the material in this appendix is built upon CS-1, the procedures for these capabilities may be undefined and FFS relative to CS-1. The material is included in this appendix to provide some technical basis for future work.

II.1.2 Relationship to other appendices of the Q.1200-Series Recommendations

This appendix only applies to Q.1218 intelligent network interface Recommendation CS-1. Each of the Q.1200-Series Recommendations includes a specific appendix, if needed.

II.1.3 Format of Document

This introduction provides an explanation of the purpose and scope of the appendix:

- Subclause II.2 itemizes the operations.
- Subclause II.3 itemizes the parameters.
- Subclause II.4 is an ASN.1 module of the operations and parameters.
- Subclause II.5 includes procedures for the operations.

II.2 Operations

The operations listed in this subclause are in addition to the operations itemized in clause 2.

II.2.1 Consideration applicable to all operations in this appendix

The following operations or aspects of the operations are FFS relative to CS-1. These operations rely on CS-1 capabilities for which the corresponding procedures are undefined. Therefore, they are included in this appendix for completeness.

The defined defaults may be incomplete (e.g. Leg ID) and may be dependent on the point in call (PIC).

II.2.2 Add party operation

II.2.2.1 Consideration

The difference between this operation and the attach operation needs to be clarified.

II.2.2.2 Description

SCF → SSF

This operation is used to perform the call processing actions to add all call party connections from one call to another call, then clear the first call (e.g. to create a conference call). From the perspective of the controlling party, this operation effectively bridges two calls.

II.2.3 Attach operation

II.2.3.1 Consideration

The difference between this operation and the add party operation needs to be clarified.

II.2.3.2 Description

SCF → SSF

This operation enables the SCF to request the SSF to include a leg in the current relationship instance. The leg is transferred from another relationship instance, from which it was removed using the detach operation. Notice that detach may also be executed after attach using the same absolute identifier.

II.2.4 Change parties operation

II.2.4.1 Description

SCF → SSF

This operation is used to perform the call processing actions to change a particular party connection from one call to another call. From the perspective of the particular call party, this operation effectively places the first call on hold and retrieves the associated call from hold.

II.2.5 Detach operation

II.2.5.1 Consideration

The difference between this operation and the release call party operation needs to be clarified.

II.2.5.2 Description

SCF → SSF

This operation enables the SCF to request the SSF to remove a leg from one relationship instance and to assign it an absolute (i.e. single network-wide) identifier (Correlation identifier), so that it can be transferred to another relationship instance, to which the leg was/will be attached by means of the attach operation using the same absolute identifier.

II.2.6 Hold call party connection operation

II.2.6.1 Description

SCF → SSF

This operation is used during the active phase of a call between two or more parties to put one party connection on hold.

II.2.7 **Initiate call attempt operation** (for case of more than 1 party)

II.2.7.1 Consideration

This information flow is included in the main body of Recommendation Q.1214, for the case of creating a call to one call party. The IF is listed in this appendix for the case of creating a call to more than one party, in the same call, which for CS-1 is FFS.

II.2.7.2 Description

SCF → SSF

This operation is used to request the SSF to create a new call to one or more parties using address information provided by the SCF (e.g. predefined conference call, previous prompt and collect information). Any errors associated with performing this operation are returned.

II.2.8 Reconnect operation

II.2.8.1 Description

SCF → SSF

This operation is used to resume a held party to a call (inverse of hold call party connection).

II.2.9 Release call party connection operation

II.2.9.1 Consideration

The difference between this operation and the detach operation needs to be clarified.

II.2.9.2 Description

SCF → SSF

This operation is used to release a call party connection during a call between two or more parties.

II.3 Parameters

The parameters listed in this subclause are additional parameters for the operations itemized in clause 2.

II.3.1 Considerations applicable to all parameters in this appendix

The following parameters are FFS relative to CS-1. These parameters rely on CS-1 capabilities for which the corresponding procedures are undefined. Therefore, they are included in this appendix for completeness.

II.3.2 Leg Id created parameter (from Analyse information operation)

II.3.2.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

II.3.3 Leg Id created parameter (from connect operation)

II.3.3.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

II.3.4 Leg Id created parameter (from initiate call attempt operation)

II.3.4.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

II.3.5 Leg Id created parameter (from select facility operation)

II.3.5.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

II.3.6 Leg Id created parameter (from select route operation)

II.3.6.1 Description

DEFAULT bPARTY

Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use with a choice of unilateral ID assignment or bilateral ID assignment.

II.3.7 Leg 1 parameter (from initial DP operation)

II.3.7.1 Description

DEFAULT {aPARTY, pending}

Indicates call party information, as defined by a Leg object. This includes a LegID to reference each call party, and a LegStatus to indicate whether the call party is connected or not.

II.3.8 Leg 2 Parameter (from initial DP operation)

II.3.8.1 Description

OPTIONAL

Indicates call party information, as defined by a Leg Object. This includes a LegID to reference each call party, and a LegStatus to indicate whether the call party is connected or not.

II.3.9 Call Id Parameter

II.3.9.1 Description

Indicates an identifier to reference an instance of a Call accessible to the SCF. Refer to 4.2.2.1/Q.1214 for a description of call segment.

II.4 ASN.1 module of operations and parameters

The following modules describe the additional operations discussed in this appendix. The modules do not describe the modifications to the operations in clause 2 to include the leg parameters discussed in subclause II.3.

II.4.1 Abstract syntax of the IN CS-1 application protocol – Appendix

This subclause specifies additional abstract syntax for the IN CS-1 application protocol using Abstract Syntax Notation One (ASN.1), defined in Recommendation X.208.

The material in this appendix is based on the appendix material in Recommendation Q.1214.

Operation types

```
IN-CS-1-Operations-appendix { ccitt recommendation q 1218 modules(0) cs-1-operations-app(4) version1(0) }
-- This module contains additional type definitions for IN CS-1 operations.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS
        OPERATION,
        ERROR

FROM TCAPMessages { ccitt recommendation q 773 modules(0) messages(1) version2(2) };
-- TYPE DEFINITION FOR ADDITIONAL    IN CS-1    OPERATIONS
-- SCF-SSF operations

AddParty ::= OPERATION
        ARGUMENT
            AddPartyArg
        RESULT
            CallPartyHandlingResultsArg
        ERRORS {
            DataAlreadyExists,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SCF → SSF
-- This operation is used to perform the call processing actions to add all call party connections
-- from one Call to another Call, then clear the first Call (e.g. to create a conference call).
-- From the perspective of the controlling party, this operation effectively bridges two Calls.

Attach ::= OPERATION
        ARGUMENT
            AttachArg
        ERRORS {
            DataAlreadyExists,
            LegIDAlreadyAssigned,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            TooLate,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
-- SCF → SSF
-- This operation is used to attach two Calls.

ChangeParties ::= OPERATION
        ARGUMENT
            ChangePartiesArg
        RESULT
            CallPartyHandlingResultsArg
        ERRORS {
            DataAlreadyExists,
            MissingParameter,
            SystemFailure,
            TaskRefused,
            UnexpectedComponentSequence,
            UnexpectedDataValue,
            UnexpectedParameter
            }
```

-- *SCF → SSF*
-- *This operation is used to perform the call processing actions to change a particular party*
-- *connection from one Call to another Call. From the perspective of the particular call party, this*
-- *operation effectively places the first Call on hold and retrieves the associated Call from hold.*

Detach ::= OPERATION
ARGUMENT
DetachArg
ERRORS {
DataAlreadyExists,
MissingParameter,
SystemFailure,
TaskRefused,
TooLate,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter,
UnknownLegID
}

-- *SCF → SSF*
-- *This operation is used to detach two Calls.*

HoldCallPartyConnection ::= OPERATION
ARGUMENT
HoldCallPartyConnectionArg
RESULT
CallPartyHandlingResultsArg
ERRORS {
DataUnavailable,
MissingParameter,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *SCF → SSF*
-- *This operation is used during the active phase of a call between two or more parties to put one*
-- *party connection on hold.*

Reconnect ::= OPERATION
ARGUMENT
ReconnectArg
RESULT
CallPartyHandlingResultsArg
ERRORS {
DataAlreadyExists,
MissingParameter,
SystemFailure,
TaskRefused,
UnexpectedComponentSequence,
UnexpectedDataValue,
UnexpectedParameter
}

-- *SCF → SSF*
-- *This operation is used to resume a held party to a call (inverse of HoldCallPartyConnection).*

ReleaseCallPartyConnection ::= OPERATION
ARGUMENT
ReleaseCallPartyConnectionArg
RESULT
CallPartyHandlingResultsArg
ERRORS {
DataAlreadyExists,
MissingParameter,
SystemFailure,

```
        TaskRefused,
        UnexpectedComponentSequence,
        UnexpectedDataValue,
        UnexpectedParameter
        }
-- SCF → SSF
-- This operation is used to release a call party connection during a call between two or more parties.

END

IN-CS-1-Errors-appendix { ccitt recommendation q 1218 modules(0) cs-1-errors-app(5) version1(0) }
-- This module contains additional type definitions for the IN CS-1 errors.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

        ERROR
FROM TCAPMessages { ccitt recommendation q 773 modules(0) messages(1) version2(2) };
-- TYPE DEFINITION FOR  IN CS-1  ERRORS

LegIDAlreadyAssigned ::= ERROR
-- Indicates that a legID has already been assigned with the requested value.

TooLate ::= ERROR
-- Indicates that the operation could not be performed in a timely manner.

UnknownLegID ::= ERROR
-- Indicates that the legID does not exist.

END
```

Data types

```
IN-CS-1-DataTypes-appendix { ccitt recommendation q 1218 modules(0) cs-1-datatypes-app(6) version1(0) }
-- This module contains additional type definitions for the IN CS-1 data types.

DEFINITIONS IMPLICIT TAGS ::=

BEGIN
-- TYPE DEFINITION FOR ADDITIONAL  IN CS-1  DATA TYPES
-- Argument Data Types

AddPartyArg ::= SEQUENCE {
        originalCallID          [0]  CallID          OPTIONAL,
        destinationCallID       [1]  CallID          OPTIONAL
        }
-- OPTIONAL denotes network operator specific use.

AttachArg ::= SEQUENCE {
        newLegID                [0]  LegID           OPTIONAL,
        correlationidentifier   [1]  CorrelationID   OPTIONAL
        }
-- OPTIONAL denotes network operator specific use.

CallPartyHandlingResultsArg ::= SEQUENCE OF LegInformation

ChangePartiesArg ::= SEQUENCE {
        callID                  [0]  CallID          OPTIONAL,
        targetCallID            [1]  CallID,
        legToBeConnectedID      [2]  LegID
        }
-- OPTIONAL denotes network operator specific use.

DetachArg ::= SEQUENCE {
        legToBeDetached         [0]  LegID           OPTIONAL,
        correlationidentifier   [1]  CorrelationID   OPTIONAL
        }
-- OPTIONAL denotes network operator specific use.
```

```
HoldCallPartyConnectionArg ::= SEQUENCE {
        callID                  [0]  CallID                 OPTIONAL,
        legID                   [1]  LegID
        }
-- OPTIONAL denotes network operator specific use.

ReconnectArg ::= SEQUENCE {
        callID                  [0]  CallID                 OPTIONAL,
        heldLegID               [1]  LegID
        }
-- OPTIONAL denotes network operator specific use.

ReleaseCallPartyConnectionArg ::= SEQUENCE {
        legToBeReleased         [0]  LegID,
        callID                  [1]  CallID                 OPTIONAL,
        releaseCause            [2]  Cause                  OPTIONAL
        }
-- OPTIONAL denotes network operator specific use. Common Data Types

CallID ::= INTEGER
-- Indicates an identifier to reference an instance of a Call accessible to the SCF. Refer to
-- 4.2.2.1/Q.1214 for a description of Call Segment.

Cause ::= OCTET STRING
-- Indicates the cause for interface related information. Refer to the Q.763 Cause parameter
-- for encoding.

LegInformation ::= SEQUENCE {
        legID                   [0]  LegID,
        legStatus               [1]  LegStatus
        }
-- Indicates call party information, as defined by a Leg object. This includes a LegID to reference
-- each call party, and a LegStatus to indicate whether the call party is connected or not.

LegStatus ::= ENUMERATED {
        connected(0),
        unconnected(1),
        pending(2),
        interacting(3)          -- user connected to a resource
        }
-- Indicates the state of the call party.

END
```

Application Protocol (Operation and Error Codes)

```
IN-CS1-Codes-appendix { ccitt recommendation q 1218 modules(0) cs-1-codes-app(7) version1(0) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN
-- OPERATION AND ERROR CODE DEFINITION
-- Code point values are for further study. The operations are grouped by the identified ASEs.
-- Call party handling ASE

addParty                        AddParty ::= ffs

changeParties                   ChangeParties ::= ffs

holdCallPartyConnection         HoldCallPartyConnection ::= ffs

reconnect                       Reconnect ::= ffs

releaseCallPartyConnection      ReleaseCallPartyConnection ::= ffs
-- Attach ASE

attach                          Attach ::= ffs

detach                          Detach ::= ffs

END
```

II.5 Procedures

PRINCIPLE is as follows:

The SSF procedures are illustrated by means of Finite State Machines (FSMs).

The (call level) finite state machine, presently described in this Recommendation, consists of several states, which represent different states of the call.

In some of the call states, another level of finite state machine(s) exists, which will have to be described, and which represents the state of a connection to one of the parties. This FSM is named "leg level FSM".

Such a "leg level FSM" is "born out" or "dies out" of some of the call level FSM transitions. Any "event" (e.g. an operation received) causes a state transition of all relevant FSMs.

The Leg states used are the possible values of the parameter “LegStatus”, are the following:

- "IDLE": connection to the party does not exist;
- "PENDING": party not connected, and under creation process;
- "UNCONNECTED": party not connected, but in a stable state;
- "CONNECTED": party connected to another party;
- "INTERACTING": party connected to an SRF.

Appendix III
Expanded ASN.1 coding

(This appendix does not form an integral part of this Recommendation)

This appendix contains the expanded ASN.1 source of the SCF/SSF/SRF interface.

For ASN.1 type definitions see 2.1.

NOTES

1 "??" as a range value means a network specific value.

2 The operations are arranged alphabetically.

```
activateServiceFiltering OPERATION
   ARGUMENT
      SEQUENCE {
         filteredCallTreatment [0] IMPLICIT SEQUENCE {
            sFBillingChargingCharacteristics [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            informationToSend [1] CHOICE {
               inbandInfo [0] IMPLICIT SEQUENCE {
                  messageID [0] CHOICE {
                     elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                     text     [1] IMPLICIT SEQUENCE {
                        messageContent [0] IMPLICIT IA5String (SIZE (??..??)),
                        attributes [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
```

```
                elementaryMessageIDs [29] IMPLICIT SEQUENCE SIZE (1..??) OF
                   INTEGER (0..2147483647),
                variableMessage [30] IMPLICIT SEQUENCE {
                   elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                   variableParts [1] IMPLICIT SEQUENCE SIZE (1..5) OF
                      CHOICE {
                         integer   [0] IMPLICIT INTEGER (0..2147483647),
                         number    [1] IMPLICIT OCTET STRING (SIZE (??..??)),
                         time      [2] IMPLICIT OCTET STRING (SIZE (2)),
                         date      [3] IMPLICIT OCTET STRING (SIZE (3)),
                         price     [4] IMPLICIT OCTET STRING (SIZE (4))}}},
             numberOfRepetitions [1] IMPLICIT INTEGER (1..127) OPTIONAL,
             duration   [2] IMPLICIT INTEGER (0..32767) OPTIONAL,
             interval   [3] IMPLICIT INTEGER (0..32767) OPTIONAL},
             tone       [1] IMPLICIT SEQUENCE {
             toneID     [0] IMPLICIT INTEGER (0..2147483647),
             duration   [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
          displayInformation [2] IMPLICIT IA5String (SIZE (??..??))} OPTIONAL,
       maximumNumberOfCounters [2] IMPLICIT INTEGER (1..100) OPTIONAL,
       releaseCause [3] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL},
    filteringCharacteristics [1] CHOICE {
       interval [0] IMPLICIT INTEGER (-1..32000),
       numberOfCalls [1] IMPLICIT INTEGER (0..2147483647)},
    filteringTimeOut [2] CHOICE {
       duration [0] IMPLICIT INTEGER (-2..86400),
       stopTime [1] IMPLICIT OCTET STRING (SIZE (6))},
    filteringCriteria [3] CHOICE {
       dialledNumber [0] IMPLICIT OCTET STRING (SIZE (??..??)),
       callingLineID [1] IMPLICIT OCTET STRING (SIZE (??..??)),
       serviceKey [2] IMPLICIT INTEGER (0..2147483647),
       addressAndService [30] IMPLICIT SEQUENCE {
          calledAddressValue [0] IMPLICIT OCTET STRING (SIZE (??..??)),
          serviceKey [1] IMPLICIT INTEGER (0..2147483647),
          callingAddressValue [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
          locationNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}},
       startTime [4] IMPLICIT OCTET STRING (SIZE (6)) OPTIONAL,
       extensions [5] IMPLICIT SEQUENCE SIZE (1..??) OF
          SEQUENCE {
             type INTEGER,
             criticality ENUMERATED {
                ignore  (0),
                abort   (1)} DEFAULT ignore ,
             value      [1] ANY DEFINED BY type  } OPTIONAL}

ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16}
 ::= localValue 42

activityTest OPERATION
 ::= localValue 55

analysedInformation OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
```

```
        dpAssignment [1] IMPLICIT ENUMERATED {
            individualLine (0),
            groupBased (1),
            officeBased (2)} OPTIONAL},
    triggerType [2] IMPLICIT ENUMERATED {
        featureActivation (0),
        verticalServiceCode (1),
        customizedAccess (2),
        customizedIntercom (3),
        emergencyService (12),
        aFR (13),
        sharedIOTrunk (14),
        offHookDelay (17),
        channelSetupPRI (18),
        tNoAnswer (25),
        tBusy (26),
        oCalledPartyBusy (27),
        oNoAnswer (29),
        originationAttemptAuthorized (30),
        oAnswer (31),
        oDisconnect (32),
        termAttemptAuthorized (33),
        tAnswer (34),
        tDisconnect (35)} OPTIONAL},
bearerCapability [1] CHOICE {
    bearerCap[0] IMPLICIT OCTET STRING (SIZE (2..??)),
    tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
cGEncountered [8] IMPLICIT ENUMERATED {
    noCGencountered (0),
    manualCGencountered (1),
    scpOverload (2)} OPTIONAL,
locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
terminalType [11] IMPLICIT ENUMERATED {
    unknown     (0),
    dialPulse   (1),
    dtmf        (2),
    isdn        (3),
    isdnNoDtmf  (4),
    spare (16)} OPTIONAL,
extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
    SEQUENCE {
        type        INTEGER,
        criticality ENUMERATED {
            ignore  (0),
            abort   (1)} DEFAULT ignore ,
        value       [1] ANY DEFINED BY type } OPTIONAL,
    chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
dialledDigits [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
callingPartyBusinessGroupID [2] IMPLICIT OCTET STRING OPTIONAL,
callingPartySubaddress [3] IMPLICIT OCTET STRING OPTIONAL,
callingFacilityGroup [4] CHOICE {
    trunkGroupID [0] IMPLICIT INTEGER,
    privateFacilityID [1] IMPLICIT INTEGER,
    huntGroup [2] IMPLICIT OCTET STRING,
    routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
callingFacilityGroupMember [5] IMPLICIT INTEGER OPTIONAL,
originalCalledPartyID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
prefix [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
redirectingPartyID [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
redirectionInformation [9] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
routeList [10] IMPLICIT SEQUENCE SIZE (1..3) OF
```

```
            OCTET STRING (SIZE (??..??)) OPTIONAL,
        travellingClassMark [11] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type        INTEGER,
                criticality ENUMERATED {
                    inore   (0),
                    abort   (1)} DEFAULT ignore ,
                value       [1] ANY DEFINED BY type } OPTIONAL,
        featureCode [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        accessCode [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        carrier [15] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 3

analyseInformation OPERATION
    ARGUMENT
        SEQUENCE {
            destinationRoutingAddress [0] IMPLICIT SEQUENCE SIZE (1..3) OF
                OCTET STRING (SIZE (??..??)),
            alertingPattern [1] IMPLICIT OCTET STRING (SIZE (3)) OPTIONAL,
            iSDNAccessRelatedInformation [2] IMPLICIT OCTET STRING OPTIONAL,
            originalCalledPartyID [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type  } OPTIONAL,
            callingPartyNumber [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [6] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            calledPartyNumber [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            chargeNumber [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            travellingClassMark [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            carrier [10] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 28

applyCharging OPERATION
    ARGUMENT
        SEQUENCE {
            aChBillingChargingCharacteristics [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            partyToCharge [2] CHOICE {
                sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
                receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type } OPTIONAL}
```

```
ERRORS {
-- missingParameter -- localValue 7,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16,
-- unexpectedDataValue -- localValue 15,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12}
 ::= localValue 35

applyChargingReport OPERATION
    ARGUMENT
        OCTET STRING (SIZE (??..??))
ERRORS {
-- missingParameter -- localValue 7,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16,
-- unexpectedDataValue -- localValue 15,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12}
 ::= localValue 36

assistRequestInstructions OPERATION
    ARGUMENT
        SEQUENCE {
            correlationID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            iPAvailable [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPSSPCapabilities [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type    } OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 16

callGap OPERATION
    ARGUMENT
        SEQUENCE {
            gapCriteria [0] CHOICE {
                calledAddressValue [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                gapOnService [2] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647),
                    dpCriteria [1] IMPLICIT ENUMERATED {
                        origAttemptAuthorized (1),
                        collectedInfo (2),
                        analysedInformation (3),
                        routeSelectFailure (4),
                        oCalledPartyBusy (5),
                        oNoAnswer (6),
                        oAnswer (7),
                        oMidCall (8),
                        oDisconnect (9),
                        oAbandon (10),
                        termAttemptAuthorized (12),
                        tBusy (13),
                        tNoAnswer (14),
```

```
            tAnswer (15),
            tMidCall (16),
            tDisconnect (17),
            tAbandon (18)} OPTIONAL},
        calledAddressAndService [29] IMPLICIT SEQUENCE {
            calledAddressValue [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            serviceKey [1] IMPLICIT INTEGER (0..2147483647)},
        callingAddressAndService [30] IMPLICIT SEQUENCE {
            callingAddressValue [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            serviceKey [1] IMPLICIT INTEGER (0..2147483647),
            locationNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}},
    gapIndicators [1] IMPLICIT SEQUENCE {
        duration [0] IMPLICIT INTEGER (-2..86400),
        gapInterval [1] IMPLICIT INTEGER (-1..60000)},
    controlType [2] IMPLICIT ENUMERATED {
        sCPOverloaded (0),
        manuallyInitiated (1),
        destinationOverload (2)} OPTIONAL,
    gapTreatment [3] CHOICE {
        informationToSend [0] CHOICE {
            inbandInfo [0] IMPLICIT SEQUENCE {
                messageID [0] CHOICE {
                    elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                    text [1] IMPLICIT SEQUENCE {
                        messageContent [0] IMPLICIT IA5String (SIZE (??..??)),
                        attributes [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
                    elementaryMessageIDs [29] IMPLICIT SEQUENCE SIZE (1..??) OF
                        INTEGER (0..2147483647),
                    variableMessage [30] IMPLICIT SEQUENCE {
                        elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                        variableParts [1] IMPLICIT SEQUENCE SIZE (1..5) OF
                            CHOICE {
                                integer     [0] IMPLICIT INTEGER (0..2147483647),
                                number      [1] IMPLICIT OCTET STRING (SIZE (??..??)),
                                time        [2] IMPLICIT OCTET STRING (SIZE (2)),
                                date        [3] IMPLICIT OCTET STRING (SIZE (3)),
                                price       [4] IMPLICIT OCTET STRING (SIZE (4))}}},
                numberOfRepetitions [1] IMPLICIT INTEGER (1..127) OPTIONAL,
                duration [2] IMPLICIT INTEGER (0..32767) OPTIONAL,
                interval [3] IMPLICIT INTEGER (0..32767) OPTIONAL},
            tone [1] IMPLICIT SEQUENCE {
                toneID [0] IMPLICIT INTEGER (0..2147483647),
                duration [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
            displayInformation [2] IMPLICIT IA5String (SIZE (??..??))},
        releaseCause [1] IMPLICIT OCTET STRING (SIZE (2..??)),
        both [2] IMPLICIT SEQUENCE {
            informationToSend [0] CHOICE {
                inbandInfo [0] IMPLICIT SEQUENCE {
                    messageID [0] CHOICE {
                        elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                        text [1] IMPLICIT SEQUENCE {
                            messageContent [0] IMPLICIT IA5String (SIZE (??..??)),
                            attributes [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
                        elementaryMessageIDs [29] IMPLICIT SEQUENCE SIZE (1..??) OF
                            INTEGER (0..2147483647),
                        variableMessage [30] IMPLICIT SEQUENCE {
                            elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                            variableParts [1] IMPLICIT SEQUENCE SIZE (1..5) OF
                                CHOICE {
                                    integer     [0] IMPLICIT INTEGER (0..2147483647),
                                    number      [1] IMPLICIT OCTET STRING (SIZE (??..??)),
                                    time        [2] IMPLICIT OCTET STRING (SIZE (2)),
                                    date        [3] IMPLICIT OCTET STRING (SIZE (3)),
                                    price       [4] IMPLICIT OCTET STRING (SIZE (4))}}},
                    numberOfRepetitions [1] IMPLICIT INTEGER (1..127) OPTIONAL,
                    duration [2] IMPLICIT INTEGER (0..32767) OPTIONAL,
                    interval [3] IMPLICIT INTEGER (0..32767) OPTIONAL},
```

```
                tone [1] IMPLICIT SEQUENCE {
                    toneID [0] IMPLICIT INTEGER (0..2147483647),
                    duration [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
                displayInformation [2] IMPLICIT IA5String (SIZE (??..??))},
            releaseCause [1] IMPLICIT OCTET STRING (SIZE (2..??))} OPTIONAL,
        extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type  } OPTIONAL}
 ::= localValue 41

callInformationReport OPERATION
    ARGUMENT
        SEQUENCE {
            requestedInformationList [0] IMPLICIT SEQUENCE SIZE (1..5) OF
                SEQUENCE {
                    requestedInformationType [0] IMPLICIT ENUMERATED {
                        callAttemptElapsedTime (0),
                        callStopTime (1),
                        callConnectedElapsedTime (2),
                        calledAddress (3),
                        releaseCause (30)},
                    requestedInformationValue [1] CHOICE {
                        callAttemptElapsedTimeValue [0] IMPLICIT INTEGER (0..255),
                        callStopTimeValue [1] IMPLICIT OCTET STRING (SIZE (6)),
                        callConnectedElapsedTimeValue [2] IMPLICIT INTEGER (0..2147483647),
                        calledAddressValue [3] IMPLICIT OCTET STRING (SIZE (??..??)),
                        releaseCauseValue [30] IMPLICIT OCTET STRING (SIZE (2..??))}},
            correlationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type  } OPTIONAL}
     ::= localValue 44

callInformationRequest OPERATION
    ARGUMENT
        SEQUENCE {
            requestedInformationTypeList [0] IMPLICIT SEQUENCE SIZE (1..5) OF
                ENUMERATED {
                    callAttemptElapsedTime (0),
                    callStopTime (1),
                    callConnectedElapsedTime (2),
                    calledAddress (3),
                    releaseCause (30)},
            correlationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type  } OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- requestedInfoError -- localValue 10,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 45
```

```
cancel OPERATION
   ARGUMENT
      CHOICE {
         invokeID [0] IMPLICIT INTEGER (-128..127),
         allRequests [1] IMPLICIT NULL}
ERRORS {
-- cancelFailed -- localValue 1,
-- missingParameter -- localValue 7,
-- taskRefused -- localValue 12}
 ::= localValue 53

cancelStatusReportRequest OPERATION
   ARGUMENT
      SEQUENCE {
         resourceID [0] CHOICE {
            lineID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            facilityGroupID [1] CHOICE {
               trunkGroupID [0] IMPLICIT INTEGER,
               privateFacilityID [1] IMPLICIT INTEGER,
               huntGroup [2] IMPLICIT OCTET STRING,
               routeIndex [3] IMPLICIT OCTET STRING},
            facilityGroupMemberID [2] IMPLICIT INTEGER,
            trunkGroupID [3] IMPLICIT INTEGER} OPTIONAL,
         extensions [1] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore      (0),
                  abort       (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type  } OPTIONAL}
ERRORS {
-- cancelFailed -- localValue 1,
-- missingParameter -- localValue 7,
-- taskRefused -- localValue 12}
 ::= localValue 54

collectedInformation OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
               triggerType [2] IMPLICIT ENUMERATED {
                  featureActivation (0),
                  verticalServiceCode (1),
                  customizedAccess (2),
                  customizedIntercom (3),
                  emergencyService (12),
                  aFR (13),
                  sharedIOTrunk (14),
                  offHookDelay (17),
                  channelSetupPRI (18),
                  tNoAnswer (25),
                  tBusy (26),
                  oCalledPartyBusy (27),
                  oNoAnswer (29),
                  originationAttemptAuthorized (30),
                  oAnswer (31),
                  oDisconnect (32),
```

```
                termAttemptAuthorized (33),
                tAnswer (34),
                tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
                bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
                noCGencountered (0),
                manualCGencountered (1),
                scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
                unknown (0),
                dialPulse (1),
                dtmf (2),
                isdn (3),
                isdnNoDtmf (4),
                spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore  (0),
                        abort   (1)} DEFAULT ignore ,
                    value       [1] ANY DEFINED BY type  } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        dialledDigits [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        callingPartyBusinessGroupID [2] IMPLICIT OCTET STRING OPTIONAL,
        callingPartySubaddress [3] IMPLICIT OCTET STRING OPTIONAL,
        callingFacilityGroup [4] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        callingFacilityGroupMember [5] IMPLICIT INTEGER OPTIONAL,
        originalCalledPartyID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        prefix [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectingPartyID [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectionInformation [9] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
        travellingClassMark [10] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [11] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type  } OPTIONAL,
        featureCode [12] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        accessCode [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        carrier [14] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 2
```

```
collectInformation OPERATION
   ARGUMENT
      SEQUENCE {
         alertingPattern [0] IMPLICIT OCTET STRING (SIZE (3)) OPTIONAL,
         numberingPlan [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
         originalCalledPartyID [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         travellingClassMark [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type  } OPTIONAL,
         callingPartyNumber [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         dialledDigits [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 27

connect OPERATION
   ARGUMENT
      SEQUENCE {
         destinationRoutingAddress [0] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)),
         alertingPattern [1] IMPLICIT OCTET STRING (SIZE (3)) OPTIONAL,
         correlationID [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         cutAndPaste [3] IMPLICIT INTEGER (0..22) OPTIONAL,
         forwardingCondition [4] IMPLICIT ENUMERATED {
            busy (0),
            noanswer (1),
            any (2)} OPTIONAL,
         iSDNAccessRelatedInformation [5] IMPLICIT OCTET STRING OPTIONAL,
         originalCalledPartyID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         routeList [7] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
         scfID [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         travellingClassMark [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [10] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type  } OPTIONAL,
         carrier [11] IMPLICIT OCTET STRING OPTIONAL,
         serviceInteractionIndicators [26] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         callingPartyNumber [27] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         callingPartysCategory [28] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
         redirectingPartyID [29] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         redirectionInformation [30] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 20
```

```
connectToResource OPERATION
   ARGUMENT
      SEQUENCE {
         resourceAddress CHOICE {
            ipRoutingAddress [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            legID [1] CHOICE {
               sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
               receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))},
            both [2] IMPLICIT SEQUENCE {
               ipRoutingAddress [0] IMPLICIT OCTET STRING (SIZE (??..??)),
               legID [1] CHOICE {
                  sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
                  receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))}},
            none [3] IMPLICIT NULL},
         extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type  } OPTIONAL,
         serviceInteractionIndicators [30] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}
ERRORS {
--  missingParameter -- localValue 7,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 19

continue OPERATION
 ::= localValue 31

disconnectForwardConnection OPERATION
ERRORS {
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14}
 ::= localValue 18

establishTemporaryConnection OPERATION
   ARGUMENT
      SEQUENCE {
         assistingSSPIPRoutingAddress [0] IMPLICIT OCTET STRING (SIZE (??..??)),
         correlationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         legID      [2] CHOICE {
            sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
            receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
         scfID      [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type  } OPTIONAL,
         carrier    [5] IMPLICIT OCTET STRING OPTIONAL,
         serviceInteractionIndicators [30] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}
ERRORS {
--  eTCFailed -- localValue 3,
--  missingParameter -- localValue 7,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 17
```

```
eventNotificationCharging OPERATION
   ARGUMENT
      SEQUENCE {
         eventTypeCharging [0] IMPLICIT OCTET STRING (SIZE (??..??)),
         eventSpecificInformationCharging [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         legID [2] CHOICE {
            sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
            receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
         extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type  } OPTIONAL,
         monitorMode [30] IMPLICIT ENUMERATED {
            interrupted (0),
            notifyAndContinue (1),
            transparent (2)} DEFAULT notifyAndContinue }
 ::= localValue 26

eventReportBCSM OPERATION
   ARGUMENT
      SEQUENCE {
         eventTypeBCSM [0] IMPLICIT ENUMERATED {
            origAttemptAuthorized (1),
            collectedInfo (2),
            analysedInformation (3),
            routeSelectFailure (4),
            oCalledPartyBusy (5),
            oNoAnswer (6),
            oAnswer (7),
            oMidCall (8),
            oDisconnect (9),
            oAbandon (10),
            termAttemptAuthorized (12),
            tBusy (13),
            tNoAnswer (14),
            tAnswer (15),
            tMidCall (16),
            tDisconnect (17),
            tAbandon (18)},
         bcsmEventCorrelationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         eventSpecificInformationBCSM [2] CHOICE {
            collectedInfoSpecificInfo [0] IMPLICIT SEQUENCE {
               calledPartynumber [0] IMPLICIT OCTET STRING (SIZE (??..??))},
            analyzedInfoSpecificInfo [1] IMPLICIT SEQUENCE {
               calledPartynumber [0] IMPLICIT OCTET STRING (SIZE (??..??))},
            routeSelectFailureSpecificInfo [2] IMPLICIT SEQUENCE {
               failureCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL},
            oCalledPartyBusySpecificInfo [3] IMPLICIT SEQUENCE {
               busyCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL},
            oNoAnswerSpecificInfo [4] IMPLICIT SEQUENCE {},
            oAnswerSpecificInfo [5] IMPLICIT SEQUENCE {},
            oMidCallSpecificInfo [6] IMPLICIT SEQUENCE {
               connectTime [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
            oDisconnectSpecificInfo [7] IMPLICIT SEQUENCE {
               releaseCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
               connectTime [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
            tBusySpecificInfo [8] IMPLICIT SEQUENCE {
               busyCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL},
            tNoAnswerSpecificInfo [9] IMPLICIT SEQUENCE {},
            tAnswerSpecificInfo [10] IMPLICIT SEQUENCE {},
            tMidCallSpecificInfo [11] IMPLICIT SEQUENCE {
               connectTime [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
            tDisconnectSpecificInfo [12] IMPLICIT SEQUENCE {
               releaseCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
               connectTime [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL}} OPTIONAL,
```

```
        legID  [3] CHOICE {
            sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
            receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
        miscCallInfo [4] IMPLICIT SEQUENCE {
            messageType [0] IMPLICIT ENUMERATED {
                request (0),
                notification  (1)},
            dpAssignment [1] IMPLICIT ENUMERATED {
                individualLine (0),
                groupBased (1),
                officeBased (2)} OPTIONAL} DEFAULT {
            messageType request    },
        extensions [5] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL}
    ::= localValue 24

furnishChargingInformation OPERATION
    ARGUMENT
        OCTET STRING (SIZE (??..??))
ERRORS {
--  missingParameter -- localValue 7,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 34

holdCallInNetwork OPERATION
    ARGUMENT
        CHOICE {
            holdcause [0] IMPLICIT OCTET STRING,
            empty [1] IMPLICIT NULL}
ERRORS {
--  missingParameter -- localValue 7,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 21

initialDP OPERATION
    ARGUMENT
        SEQUENCE {
            serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
            dialledDigits [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyBusinessGroupID [4] IMPLICIT OCTET STRING OPTIONAL,
            callingPartysCategory [5] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            callingPartySubaddress [6] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [7] IMPLICIT ENUMERATED {
                noCGencountered (0),
                manualCGencountered (1),
                scpOverload (2)} OPTIONAL,
            iPSSPCapabilities [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            locationNumber [10] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            miscCallInfo [11] IMPLICIT SEQUENCE {
                messageType [0] IMPLICIT ENUMERATED {
                    request (0),
                    notification (1)},
```

```
    dpAssignment [1] IMPLICIT ENUMERATED {
        individualLine (0),
        groupBased (1),
        officeBased (2)} OPTIONAL} OPTIONAL,
originalCalledPartyID [12] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
serviceProfileIdentifier [13] IMPLICIT OCTET STRING OPTIONAL,
    terminalType [14] IMPLICIT ENUMERATED {
    unknown (0),
    dialPulse (1),
    dtmf (2),
    isdn (3),
    isdnNoDtmf (4),
    spare (16)} OPTIONAL,
extensions [15] IMPLICIT SEQUENCE SIZE (1..??) OF
    SEQUENCE {
        type INTEGER,
        criticality ENUMERATED {
            ignore      (0),
            abort       (1)} DEFAULT ignore ,
        value           [1] ANY DEFINED BY type   } OPTIONAL,
triggerType [16] IMPLICIT ENUMERATED {
    featureActivation (0),
    verticalServiceCode (1),
    customizedAccess (2),
    customizedIntercom (3),
    emergencyService (12),
    aFR (13),
    sharedIOTrunk (14),
    offHookDelay (17),
    channelSetupPRI (18),
    tNoAnswer (25),
    tBusy (26),
    oCalledPartyBusy (27),
    oNoAnswer (29),
    originationAttemptAuthorized (30),
    oAnswer (31),
    oDisconnect (32),
    termAttemptAuthorized (33),
    tAnswer (34),
    tDisconnect (35)} OPTIONAL,
highLayerCompatibility [23] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
serviceInteractionIndicators [24] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
additionalCallingPartyNumber [25] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
forwardCallIndicators [26] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
bearerCapability [27] CHOICE {
    bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
    tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
eventTypeBCSM [28] IMPLICIT ENUMERATED {
    origAttemptAuthorized (1),
    collectedInfo (2),
    analysedInformation (3),
    routeSelectFailure (4),
    oCalledPartyBusy (5),
    oNoAnswer (6),
    oAnswer (7),
    oMidCall (8),
    oDisconnect (9),
    oAbandon (10),
    termAttemptAuthorized (12),
    tBusy (13),
    tNoAnswer (14),
    tAnswer (15),
    tMidCall (16),
    tDisconnect (17),
    tAbandon (18)} OPTIONAL,
redirectingPartyID [29] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
redirectionInformation [30] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL}
```

```
    ERRORS {
--  missingCustomerRecord -- localValue 6,
--  missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 0

initiateCallAttempt OPERATION
    ARGUMENT
        SEQUENCE {
            destinationRoutingAddress [0] IMPLICIT SEQUENCE SIZE (1..3) OF
                OCTET STRING (SIZE (??..??)),
            alertingPattern [1] IMPLICIT OCTET STRING (SIZE (3)) OPTIONAL,
            iSDNAccessRelatedInformation [2] IMPLICIT OCTET STRING OPTIONAL,
            travellingClassMark [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type   } OPTIONAL,
            serviceInteractionIndicators [29] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [30] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}
ERRORS {
--  missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 32

    oAnswer OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                    miscCallInfo [1] IMPLICIT SEQUENCE {
                        messageType [0] IMPLICIT ENUMERATED {
                            request (0),
                            notification (1)},
                        dpAssignment [1] IMPLICIT ENUMERATED {
                            individualLine (0),
                            groupBased (1),
                            officeBased (2)} OPTIONAL},
                        triggerType [2] IMPLICIT ENUMERATED {
                            featureActivation (0),
                            verticalServiceCode (1),
                            customizedAccess (2),
                            customizedIntercom (3),
                            emergencyService (12),
                            aFR (13),
                            sharedIOTrunk (14),
                            offHookDelay (17),
                            channelSetupPRI (18),
                            tNoAnswer (25),
                            tBusy (26),
                            oCalledPartyBusy (27),
                            oNoAnswer (29),
```

```
                originationAttemptAuthorized (30),
                oAnswer (31),
                oDisconnect (32),
                termAttemptAuthorized (33),
                tAnswer (34),
                tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
                bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
                noCGencountered (0),
                manualCGencountered (1),
                scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
                unknown (0),
                dialPulse (1),
                dtmf (2),
                isdn (3),
                isdnNoDtmf (4),
                spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore     (0),
                        abort      (1)} DEFAULT ignore ,
                    value          [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        callingPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
        callingPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
        callingFacilityGroup [3] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        callingFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
        originalCalledPartyID [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectingPartyID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectionInformation [7] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
        routeList [8] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
        travellingClassMark [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [10] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore  (0),
                    abort   (1)} DEFAULT ignore ,
                value       [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 7
```

```
oCalledPartyBusy OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification  (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
               triggerType [2] IMPLICIT ENUMERATED {
                  featureActivation (0),
                  verticalServiceCode (1),
                  customizedAccess (2),
                  customizedIntercom (3),
                  emergencyService (12),
                  aFR (13),
                  sharedIOTrunk (14),
                  offHookDelay (17),
                  channelSetupPRI (18),
                  tNoAnswer (25),
                  tBusy (26),
                  oCalledPartyBusy (27),
                  oNoAnswer (29),
                  originationAttemptAuthorized (30),
                  oAnswer (31),
                  oDisconnect (32),
                  termAttemptAuthorized (33),
                  tAnswer (34),
                  tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
               bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
               tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
               noCGencountered (0),
               manualCGencountered (1),
               scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
               unknown (0),
               dialPulse (1),
               dtmf (2),
               isdn (3),
               isdnNoDtmf (4),
               spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
               SEQUENCE {
                  type INTEGER,
                  criticality ENUMERATED {
                     ignore  (0),
                     abort   (1)} DEFAULT ignore ,
                  value       [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
         busyCause [1] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
         callingPartyBusinessGroupID [2] IMPLICIT OCTET STRING OPTIONAL,
         callingPartySubaddress [3] IMPLICIT OCTET STRING OPTIONAL,
         callingFacilityGroup [4] CHOICE {
```

```
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        callingFacilityGroupMember [5] IMPLICIT INTEGER OPTIONAL,
        originalCalledPartyID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        prefix [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectingPartyID [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectionInformation [9] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
        routeList [10] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
        travellingClassMark [11] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL,
        carrier [13] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 5

oDisconnect OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                    miscCallInfo [1] IMPLICIT SEQUENCE {
                        messageType [0] IMPLICIT ENUMERATED {
                            request (0),
                            notification (1)},
                        dpAssignment [1] IMPLICIT ENUMERATED {
                            individualLine (0),
                            groupBased (1),
                            officeBased (2)} OPTIONAL},
                        triggerType [2] IMPLICIT ENUMERATED {
                            featureActivation (0),
                            verticalServiceCode (1),
                            customizedAccess (2),
                            customizedIntercom (3),
                            emergencyService (12),
                            aFR (13),
                            sharedIOTrunk (14),
                            offHookDelay (17),
                            channelSetupPRI (18),
                            tNoAnswer (25),
                            tBusy (26),
                            oCalledPartyBusy (27),
                            oNoAnswer (29),
                            originationAttemptAuthorized (30),
                            oAnswer (31),
                            oDisconnect (32),
                            termAttemptAuthorized (33),
                            tAnswer (34),
                            tDisconnect (35)} OPTIONAL},
```

```
            bearerCapability [1] CHOICE {
                bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                tmr  [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
                noCGencountered (0),
                manualCGencountered (1),
                scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
                unknown (0),
                dialPulse (1),
                dtmf (2),
                isdn (3),
                isdnNoDtmf (4),
                spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore     (0),
                        abort      (1)} DEFAULT ignore ,
                    value [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        callingPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
        callingPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
        callingFacilityGroup [3] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        callingFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
        releaseCause [5] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
        routeList [6] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [7] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore  (0),
                    abort   (1)} DEFAULT ignore ,
                value       [1] ANY DEFINED BY type   } OPTIONAL,
        carrier [8] IMPLICIT OCTET STRING OPTIONAL,
        connectTime [9] IMPLICIT INTEGER (0..2147483647) OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 8

oMidCall   OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
```

```
    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
    miscCallInfo [1] IMPLICIT SEQUENCE {
        messageType [0] IMPLICIT ENUMERATED {
            request (0),
            notification (1)},
        dpAssignment [1] IMPLICIT ENUMERATED {
            individualLine (0),
            groupBased (1),
            officeBased (2)} OPTIONAL},
        triggerType [2] IMPLICIT ENUMERATED {
            featureActivation (0),
            verticalServiceCode (1),
            customizedAccess (2),
            customizedIntercom (3),
            emergencyService (12),
            aFR (13),
            sharedIOTrunk (14),
            offHookDelay (17),
            channelSetupPRI (18),
            tNoAnswer (25),
            tBusy (26),
            oCalledPartyBusy (27),
            oNoAnswer (29),
            originationAttemptAuthorized (30),
            oAnswer (31),
            oDisconnect (32),
            termAttemptAuthorized (33),
            tAnswer (34),
            tDisconnect (35)} OPTIONAL},
    bearerCapability [1] CHOICE {
        bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
        tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
    calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
    iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
    cGEncountered [8] IMPLICIT ENUMERATED {
        noCGencountered (0),
        manualCGencountered (1),
        scpOverload (2)} OPTIONAL,
    locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
    terminalType [11] IMPLICIT ENUMERATED {
        unknown (0),
        dialPulse (1),
        dtmf (2),
        isdn (3),
        isdnNoDtmf (4),
        spare (16)} OPTIONAL,
    extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
        SEQUENCE {
            type INTEGER,
            criticality ENUMERATED {
                ignore     (0),
                abort      (1)} DEFAULT ignore ,
            value   [1] ANY DEFINED BY type   } OPTIONAL,
    chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
calledPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
calledPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
callingPartyBusinessGroupID [3] IMPLICIT OCTET STRING OPTIONAL,
callingPartySubaddress [4] IMPLICIT OCTET STRING OPTIONAL,
featureRequestIndicator [5] IMPLICIT ENUMERATED {
    hold (0),
    retrieve (1),
    featureActivation (2),
    spare1 (3),
```

```
            sparen (127)} OPTIONAL,
        extensions [6] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore  (0),
                    abort   (1)} DEFAULT ignore ,
                value       [1] ANY DEFINED BY type   } OPTIONAL,
        carrier [7] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 14

oNoAnswer OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
                  triggerType [2] IMPLICIT ENUMERATED {
                     featureActivation (0),
                     verticalServiceCode (1),
                     customizedAccess (2),
                     customizedIntercom (3),
                     emergencyService (12),
                     aFR (13),
                     sharedIOTrunk (14),
                     offHookDelay (17),
                     channelSetupPRI (18),
                     tNoAnswer (25),
                     tBusy (26),
                     oCalledPartyBusy (27),
                     oNoAnswer (29),
                     originationAttemptAuthorized (30),
                     oAnswer (31),
                     oDisconnect (32),
                     termAttemptAuthorized (33),
                     tAnswer (34),
                     tDisconnect (35)} OPTIONAL},
               bearerCapability [1] CHOICE {
                  bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                  tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
               calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
               callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
               callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
               iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
               iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
               iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
               cGEncountered [8] IMPLICIT ENUMERATED {
                  noCGencountered (0),
                  manualCGencountered (1),
                  scpOverload (2)} OPTIONAL,
```

```
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
               unknown (0),
               dialPulse (1),
               dtmf (2),
               isdn (3),
               isdnNoDtmf (4),
               spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
               SEQUENCE {
                  type INTEGER,
                  criticality ENUMERATED {
                     ignore     (0),
                     abort      (1)} DEFAULT ignore ,
                  value          [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
         callingPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
         callingPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
         callingFacilityGroup [3] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
         callingFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
         originalCalledPartyID [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         prefix [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         redirectingPartyID [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         redirectionInformation [8] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
         routeList  [9] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
         travellingClassMark [10] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [11] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore  (0),
                  abort   (1)} DEFAULT ignore ,
               value       [1] ANY DEFINED BY type   } OPTIONAL,
         carrier [12] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 6

originationAttemptAuthorized OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
```

```
        triggerType [2] IMPLICIT ENUMERATED {
            featureActivation (0),
            verticalServiceCode (1),
            customizedAccess (2),
            customizedIntercom (3),
            emergencyService (12),
            aFR (13),
            sharedIOTrunk (14),
            offHookDelay (17),
            channelSetupPRI (18),
            tNoAnswer (25),
            tBusy (26),
            oCalledPartyBusy (27),
            oNoAnswer (29),
            originationAttemptAuthorized  (30),
            oAnswer (31),
            oDisconnect (32),
            termAttemptAuthorized  (33),
            tAnswer (34),
            tDisconnect (35)} OPTIONAL},
    bearerCapability [1] CHOICE {
        bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
        tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
    calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
    iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
    cGEncountered [8] IMPLICIT ENUMERATED {
        noCGencountered (0),
        manualCGencountered (1),
        scpOverload (2)} OPTIONAL,
    locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
    terminalType [11] IMPLICIT ENUMERATED {
        unknown (0),
        dialPulse (1),
        dtmf (2),
        isdn (3),
        isdnNoDtmf (4),
        spare (16)} OPTIONAL,
    extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
        SEQUENCE {
            type INTEGER,
            criticality ENUMERATED {
                ignore  (0),
                abort   (1)} DEFAULT ignore ,
            value       [1] ANY DEFINED BY type   } OPTIONAL,
    chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
dialledDigits [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
callingPartyBusinessGroupID [2] IMPLICIT OCTET STRING OPTIONAL,
callingPartySubaddress [3] IMPLICIT OCTET STRING OPTIONAL,
callingFacilityGroup [4] CHOICE {
    trunkGroupID [0] IMPLICIT INTEGER,
    privateFacilityID [1] IMPLICIT INTEGER,
    huntGroup [2] IMPLICIT OCTET STRING,
    routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
callingFacilityGroupMember [5] IMPLICIT INTEGER OPTIONAL,
travellingClassMark [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
extensions [7] IMPLICIT SEQUENCE SIZE (1..??) OF
    SEQUENCE {
        type INTEGER,
        criticality ENUMERATED {
            ignore      (0),
            abort       (1)} DEFAULT ignore ,
        value           [1] ANY DEFINED BY type   } OPTIONAL,
carrier [8] IMPLICIT OCTET STRING OPTIONAL}
```

```
ERRORS {
--  missingCustomerRecord -- localValue 6,
--  missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 1

playAnnouncement OPERATION
   ARGUMENT
      SEQUENCE {
         informationToSend [0] CHOICE {
            inbandInfo [0] IMPLICIT SEQUENCE {
               messageID [0] CHOICE {
                  elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                  text [1] IMPLICIT SEQUENCE {
                     messageContent [0] IMPLICIT IA5String (SIZE (??..??)),
                     attributes [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
                  elementaryMessageIDs [29] IMPLICIT SEQUENCE SIZE (1..??) OF
                     INTEGER (0..2147483647),
                  variableMessage [30] IMPLICIT SEQUENCE {
                     elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                     variableParts [1] IMPLICIT SEQUENCE SIZE (1..5) OF
                        CHOICE {
                           integer [0] IMPLICIT INTEGER (0..2147483647),
                           number [1] IMPLICIT OCTET STRING (SIZE (??..??)),
                           time [2] IMPLICIT OCTET STRING (SIZE (2)),
                           date [3] IMPLICIT OCTET STRING (SIZE (3)),
                           price [4] IMPLICIT OCTET STRING (SIZE (4))}}},
               numberOfRepetitions [1] IMPLICIT INTEGER (1..127) OPTIONAL,
               duration [2] IMPLICIT INTEGER (0..32767) OPTIONAL,
               interval [3] IMPLICIT INTEGER (0..32767) OPTIONAL},
            tone [1] IMPLICIT SEQUENCE {
               toneID [0] IMPLICIT INTEGER (0..2147483647),
               duration [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
            displayInformation [2] IMPLICIT IA5String (SIZE (??..??))},
         disconnectFromIPForbidden [1] IMPLICIT BOOLEAN DEFAULT TRUE,
         requestAnnouncementComplete [2] IMPLICIT BOOLEAN DEFAULT TRUE,
         extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore      (0),
                  abort       (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
--  canceled -- localValue 0,
--  missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16,
--  unavailableResource -- localValue 13}
LINKED {
--  specializedResourceReport -- localValue 49}
 ::= localValue 47

promptAndCollectUserInformation OPERATION
   ARGUMENT
      SEQUENCE {
         collectedInfo [0] CHOICE {
```

```
            collectedDigits [0] IMPLICIT SEQUENCE {
               minimumNbOfDigits [0] IMPLICIT INTEGER (1..127) DEFAULT 1,
               maximumNbOfDigits [1] IMPLICIT INTEGER (1..127),
               endOfReplyDigit [2] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
               cancelDigit [3] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
               startDigit [4] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
               firstDigitTimeOut [5] IMPLICIT INTEGER (1..127) OPTIONAL,
               interDigitTimeOut [6] IMPLICIT INTEGER (1..127) OPTIONAL,
               errorTreatment [7] IMPLICIT ENUMERATED {
                  reportErrorToScf (0),
                  help (1),
                  repeatPrompt (2)} DEFAULT reportErrorToScf ,
               interruptableAnnInd [8] IMPLICIT BOOLEAN DEFAULT TRUE,
               voiceInformation [9] IMPLICIT BOOLEAN DEFAULT FALSE,
               voiceBack [10] IMPLICIT BOOLEAN DEFAULT FALSE},
            iA5Information [1] IMPLICIT BOOLEAN},
         disconnectFromIPForbidden [1] IMPLICIT BOOLEAN DEFAULT TRUE,
         informationToSend [2] CHOICE {
            inbandInfo [0] IMPLICIT SEQUENCE {
               messageID [0] CHOICE {
                  elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                  text [1] IMPLICIT SEQUENCE {
                     messageContent [0] IMPLICIT IA5String   (SIZE (??..??)),
                     attributes [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
                  elementaryMessageIDs [29] IMPLICIT SEQUENCE SIZE (1..??) OF
                     INTEGER (0..2147483647),
                  variableMessage [30] IMPLICIT SEQUENCE {
                     elementaryMessageID [0] IMPLICIT INTEGER (0..2147483647),
                     variableParts [1] IMPLICIT SEQUENCE SIZE (1..5) OF
                        CHOICE {
                           integer [0] IMPLICIT INTEGER (0..2147483647),
                           number [1] IMPLICIT OCTET STRING (SIZE (??..??)),
                           time [2] IMPLICIT OCTET STRING (SIZE (2)),
                           date [3] IMPLICIT OCTET STRING (SIZE (3)),
                           price [4] IMPLICIT OCTET STRING (SIZE (4))}}},
               numberOfRepetitions [1] IMPLICIT INTEGER (1..127) OPTIONAL,
               duration [2] IMPLICIT INTEGER (0..32767) OPTIONAL,
               interval [3] IMPLICIT INTEGER (0..32767) OPTIONAL},
            tone [1] IMPLICIT SEQUENCE {
               toneID [0] IMPLICIT INTEGER (0..2147483647),
               duration [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL},
            displayInformation [2] IMPLICIT IA5String (SIZE (??..??))} OPTIONAL,
         extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore      (0),
                  abort       (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
RESULT
   CHOICE {
      digitsResponse [0] IMPLICIT OCTET STRING (SIZE (??..??)),
      iA5Response [1] IMPLICIT IA5String  }
ERRORS {
-- canceled -- localValue 0,
-- improperCallerResponse -- localValue 4,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unavailableResource -- localValue 13,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 48
```

```
releaseCall OPERATION
   ARGUMENT
      OCTET STRING (SIZE (2..??))
 ::= localValue 22

requestCurrentStatusReport OPERATION
   ARGUMENT
      CHOICE {
         lineID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
         facilityGroupID [1] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING},
         facilityGroupMemberID [2] IMPLICIT INTEGER,
         trunkGroupID [3] IMPLICIT INTEGER}
   RESULT
      SEQUENCE {
         resourceStatus [0] IMPLICIT ENUMERATED {
            busy (0),
            idle (1)},
         resourceID [1] CHOICE {
            lineID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            facilityGroupID [1] CHOICE {
               trunkGroupID [0] IMPLICIT INTEGER,
               privateFacilityID [1] IMPLICIT INTEGER,
               huntGroup [2] IMPLICIT OCTET STRING,
               routeIndex [3] IMPLICIT OCTET STRING},
            facilityGroupMemberID [2] IMPLICIT INTEGER,
            trunkGroupID [3] IMPLICIT INTEGER} OPTIONAL,
         extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16,
-- unknownResource -- localValue 18}
 ::= localValue 37

requestEveryStatusChangeReport OPERATION
   ARGUMENT
      SEQUENCE {
         resourceID [0] CHOICE {
            lineID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            facilityGroupID [1] CHOICE {
               trunkGroupID [0] IMPLICIT INTEGER,
               privateFacilityID [1] IMPLICIT INTEGER,
               huntGroup [2] IMPLICIT OCTET STRING,
               routeIndex [3] IMPLICIT OCTET STRING},
            facilityGroupMemberID [2] IMPLICIT INTEGER,
            trunkGroupID [3] IMPLICIT INTEGER},
         correlationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         monitorDuration [2] IMPLICIT INTEGER (-2..86400) OPTIONAL,
         extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
```

```
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16,
-- unknownResource -- localValue 18}
 ::= localValue 38

requestFirstStatusMatchReport OPERATION
   ARGUMENT
      SEQUENCE {
         resourceID [0] CHOICE {
            lineID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            facilityGroupID [1] CHOICE {
               trunkGroupID [0] IMPLICIT INTEGER,
               privateFacilityID [1] IMPLICIT INTEGER,
               huntGroup [2] IMPLICIT OCTET STRING,
               routeIndex [3] IMPLICIT OCTET STRING},
            facilityGroupMemberID [2] IMPLICIT INTEGER,
            trunkGroupID [3] IMPLICIT INTEGER} OPTIONAL,
         resourceStatus [1] IMPLICIT ENUMERATED {
            busy (0),
            idle (1)} OPTIONAL,
         correlationID [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         monitorDuration [3] IMPLICIT INTEGER (-2..86400) OPTIONAL,
         extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL,
         bearerCapability [5] CHOICE {
            bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
            tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16,
-- unknownResource -- localValue 18}
 ::= localValue 39

requestNotificationChargingEvent OPERATION
   ARGUMENT
      SEQUENCE SIZE (1..??) OF
         SEQUENCE {
            eventTypeCharging [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            monitorMode [1] IMPLICIT ENUMERATED {
               interrupted (0),
               notifyAndContinue (1),
               transparent (2)},
            legID [2] CHOICE {
               sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
               receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
```

```
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 25

requestReportBCSMEvent OPERATION
   ARGUMENT
      SEQUENCE {
         bcsmEvents [0] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               eventTypeBCSM [0] IMPLICIT ENUMERATED {
                  origAttemptAuthorized (1),
                  collectedInfo (2),
                  analysedInformation (3),
                  routeSelectFailure (4),
                  oCalledPartyBusy (5),
                  oNoAnswer (6),
                  oAnswer (7),
                  oMidCall (8),
                  oDisconnect (9),
                  oAbandon (10),
                  termAttemptAuthorized (12),
                  tBusy (13),
                  tNoAnswer (14),
                  tAnswer (15),
                  tMidCall (16),
                  tDisconnect (17),
                  tAbandon (18)},
               monitorMode [1] IMPLICIT ENUMERATED {
                  interrupted (0),
                  notifyAndContinue (1),
                  transparent (2)},
               legID [2] CHOICE {
                  sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
                  receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
               dpSpecificCriteria [30] CHOICE {
                  numberOfDigits [0] IMPLICIT INTEGER (1..255),
                  applicationTimer [1] IMPLICIT INTEGER (0..2047)} OPTIONAL},
         bcsmEventCorrelationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 23

resetTimer OPERATION
   ARGUMENT
      SEQUENCE {
         timerID [0] IMPLICIT ENUMERATED {
            tssf (0)} DEFAULT tssf   ,
         timervalue [1] IMPLICIT INTEGER (0..2147483647),
         extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
```

```
ERRORS {
--  missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 33

routeSelectFailure OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
               triggerType [2] IMPLICIT ENUMERATED {
                  featureActivation (0),
                  verticalServiceCode (1),
                  customizedAccess (2),
                  customizedIntercom (3),
                  emergencyService (12),
                  aFR (13),
                  sharedIOTrunk (14),
                  offHookDelay (17),
                  channelSetupPRI  (18),
                  tNoAnswer (25),
                  tBusy (26),
                  oCalledPartyBusy (27),
                  oNoAnswer (29),
                  originationAttemptAuthorized  (30),
                  oAnswer (31),
                  oDisconnect (32),
                  termAttemptAuthorized  (33),
                  tAnswer (34),
                  tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
               bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
               tmr  [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
               noCGencountered (0),
               manualCGencountered (1),
               scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
               unknown (0),
               dialPulse (1),
               dtmf (2),
               isdn (3),
               isdnNoDtmf (4),
               spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
               SEQUENCE {
                  type INTEGER,
                  criticality ENUMERATED {
```

```
                    ignore  (0),
                    abort   (1)} DEFAULT ignore ,
                value       [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        dialledDigits [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        callingPartyBusinessGroupID [2] IMPLICIT OCTET STRING OPTIONAL,
        callingPartySubaddress [3] IMPLICIT OCTET STRING OPTIONAL,
        callingFacilityGroup [4] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        callingFacilityGroupMember [5] IMPLICIT INTEGER OPTIONAL,
        failureCause [6] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
        originalCalledPartyID [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        prefix [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectingPartyID [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectionInformation [10] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
        routeList [11] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
        travellingClassMark [12] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [13] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL,
        carrier [14] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 4

selectFacility OPERATION
    ARGUMENT
        SEQUENCE {
            alertingPattern [0] IMPLICIT OCTET STRING (SIZE (3)) OPTIONAL,
            destinationNumberRoutingAddress [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [2] IMPLICIT OCTET STRING OPTIONAL,
            calledFacilityGroup [3] CHOICE {
                trunkGroupID [0] IMPLICIT INTEGER,
                privateFacilityID [1] IMPLICIT INTEGER,
                huntGroup  [2] IMPLICIT OCTET STRING,
                routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
            calledFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
            originalCalledPartyID [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            extensions [6] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
```

```
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 30

selectRoute OPERATION
   ARGUMENT
      SEQUENCE {
         destinationRoutingAddress [0] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)),
         alertingPattern [1] IMPLICIT OCTET STRING (SIZE (3)) OPTIONAL,
         correlationID [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         iSDNAccessRelatedInformation [3] IMPLICIT OCTET STRING OPTIONAL,
         originalCalledPartyID [4] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         routeList [5] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
         scfID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         travellingClassMark [7] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [8] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL,
         carrier [9] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 29

sendChargingInformation OPERATION
   ARGUMENT
      SEQUENCE {
         sCIBillingChargingCharacteristics [0] IMPLICIT OCTET STRING (SIZE (??..??)),
         partyToCharge [1] CHOICE {
            sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
            receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))},
         extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore     (0),
                  abort      (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingParameter -- localValue 7,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedParameter -- localValue 16,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unknownLegID -- localValue 17}
 ::= localValue 46

serviceFilteringResponse OPERATION
   ARGUMENT
      SEQUENCE {
         countersValue [0] IMPLICIT SEQUENCE SIZE (0..100) OF
            SEQUENCE {
               counterID  [0] IMPLICIT INTEGER (0..99),
               counterValue [1] IMPLICIT INTEGER (0..2147483647)},
```

```
        filteringCriteria [1] CHOICE {
            dialledNumber [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            callingLineID [1] IMPLICIT OCTET STRING (SIZE (??..??)),
            serviceKey [2] IMPLICIT INTEGER (0..2147483647),
            addressAndService [30] IMPLICIT SEQUENCE {
                calledAddressValue [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                serviceKey [1] IMPLICIT INTEGER (0..2147483647),
                callingAddressValue [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                locationNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}},
        extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL,
        responseCondition [3] IMPLICIT ENUMERATED {
            intermediateResponse (0),
            lastResponse (1)} OPTIONAL}
 ::= localValue 43

specializedResourceReport OPERATION
    ARGUMENT
        NULL
 ::= localValue 49

statusReport OPERATION
    ARGUMENT
        SEQUENCE {
            resourceStatus [0] IMPLICIT ENUMERATED {
                busy (0),
                idle (1)} OPTIONAL,
            correlationID [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            resourceID [2] CHOICE {
                lineID [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                facilityGroupID [1] CHOICE {
                    trunkGroupID [0] IMPLICIT INTEGER,
                    privateFacilityID [1] IMPLICIT INTEGER,
                    huntGroup [2] IMPLICIT OCTET STRING,
                    routeIndex [3] IMPLICIT OCTET STRING},
                facilityGroupMemberID [2] IMPLICIT INTEGER,
                trunkGroupID [3] IMPLICIT INTEGER} OPTIONAL,
            extensions [3] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore      (0),
                        abort       (1)} DEFAULT ignore ,
                    value           [1] ANY DEFINED BY type  } OPTIONAL,
            reportCondition [4] IMPLICIT ENUMERATED {
                statusReport (0),
                timerExpired (1),
                canceled (2)} OPTIONAL}
 ::= localValue 40

tAnswer OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                    miscCallInfo [1] IMPLICIT SEQUENCE {
                        messageType [0] IMPLICIT ENUMERATED {
                            request (0),
                            notification (1)},
                        dpAssignment [1] IMPLICIT ENUMERATED {
                            individualLine (0),
                            groupBased (1),
                            officeBased (2)} OPTIONAL},
```

```
            triggerType [2] IMPLICIT ENUMERATED {
                featureActivation (0),
                verticalServiceCode (1),
                customizedAccess (2),
                customizedIntercom (3),
                emergencyService (12),
                aFR (13),
                sharedIOTrunk (14),
                offHookDelay (17),
                channelSetupPRI (18),
                tNoAnswer (25),
                tBusy (26),
                oCalledPartyBusy (27),
                oNoAnswer (29),
                originationAttemptAuthorized (30),
                oAnswer (31),
                oDisconnect (32),
                termAttemptAuthorized (33),
                tAnswer (34),
                tDisconnect (35)} OPTIONAL},
        bearerCapability [1] CHOICE {
            bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
            tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
        calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
        cGEncountered [8] IMPLICIT ENUMERATED {
            noCGencountered (0),
            manualCGencountered (1),
            scpOverload  (2)} OPTIONAL,
        locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
        terminalType [11] IMPLICIT ENUMERATED {
            unknown (0),
            dialPulse (1),
            dtmf (2),
            isdn (3),
            isdnNoDtmf (4),
            spare (16)} OPTIONAL,
        extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore  (0),
                    abort   (1)} DEFAULT ignore ,
                value       [1] ANY DEFINED BY type   } OPTIONAL,
        chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
    calledPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
    calledPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
    calledFacilityGroup [3] CHOICE {
        trunkGroupID [0] IMPLICIT INTEGER,
        privateFacilityID [1] IMPLICIT INTEGER,
        huntGroup  [2] IMPLICIT OCTET STRING,
        routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
    calledFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
    extensions [5] IMPLICIT SEQUENCE SIZE (1..??) OF
        SEQUENCE {
            type INTEGER,
            criticality ENUMERATED {
                ignore      (0),
                abort       (1)} DEFAULT ignore ,
            value           [1] ANY DEFINED BY type   } OPTIONAL}
```

```
ERRORS {
--  missingCustomerRecord -- localValue 6,
--  missingParameter -- localValue 7,
--  parameterOutOfRange -- localValue 8,
--  systemFailure -- localValue 11,
--  taskRefused -- localValue 12,
--  unexpectedComponentSequence -- localValue 14,
--  unexpectedDataValue -- localValue 15,
--  unexpectedParameter -- localValue 16}
 ::= localValue 12

tBusy OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                    miscCallInfo [1] IMPLICIT SEQUENCE {
                        messageType [0] IMPLICIT ENUMERATED {
                            request (0),
                            notification (1)},
                        dpAssignment [1] IMPLICIT ENUMERATED {
                            individualLine (0),
                            groupBased (1),
                            officeBased (2)} OPTIONAL},
                    triggerType [2] IMPLICIT ENUMERATED {
                        featureActivation (0),
                        verticalServiceCode (1),
                        customizedAccess (2),
                        customizedIntercom (3),
                        emergencyService (12),
                        aFR (13),
                        sharedIOTrunk (14),
                        offHookDelay (17),
                        channelSetupPRI (18),
                        tNoAnswer (25),
                        tBusy (26),
                        oCalledPartyBusy (27),
                        oNoAnswer (29),
                        originationAttemptAuthorized  (30),
                        oAnswer (31),
                        oDisconnect (32),
                        termAttemptAuthorized (33),
                        tAnswer (34),
                        tDisconnect (35)} OPTIONAL},
                bearerCapability [1] CHOICE {
                    bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                    tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
                calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
                iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
                cGEncountered [8] IMPLICIT ENUMERATED {
                    noCGencountered (0),
                    manualCGencountered (1),
                    scpOverload (2)} OPTIONAL,
                locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
                terminalType [11] IMPLICIT ENUMERATED {
                    unknown (0),
                    dialPulse (1),
                    dtmf (2),
                    isdn (3),
                    isdnNoDtmf (4),
                    spare (16)} OPTIONAL,
```

```
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
               SEQUENCE {
                  type INTEGER,
                  criticality ENUMERATED {
                     ignore  (0),
                     abort   (1)} DEFAULT ignore ,
                  value       [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
         busyCause [1] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
         calledPartyBusinessGroupID [2] IMPLICIT OCTET STRING OPTIONAL,
         calledPartySubaddress [3] IMPLICIT OCTET STRING OPTIONAL,
         originalCalledPartyID [4] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         redirectingPartyID [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         redirectionInformation [6] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
         routeList [7] IMPLICIT SEQUENCE SIZE (1..3) OF
            OCTET STRING (SIZE (??..??)) OPTIONAL,
         travellingClassMark [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
         extensions [9] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
               type INTEGER,
               criticality ENUMERATED {
                  ignore      (0),
                  abort       (1)} DEFAULT ignore ,
               value          [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 10

tDisconnect OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
               triggerType [2] IMPLICIT ENUMERATED {
                  featureActivation (0),
                  verticalServiceCode (1),
                  customizedAccess (2),
                  customizedIntercom (3),
                  emergencyService (12),
                  aFR (13),
                  sharedIOTrunk (14),
                  offHookDelay (17),
                  channelSetupPRI (18),
                  tNoAnswer (25),
                  tBusy (26),
                  oCalledPartyBusy (27),
                  oNoAnswer (29),
                  originationAttemptAuthorized (30),
                  oAnswer (31),
                  oDisconnect (32),
                  termAttemptAuthorized (33),
```

```
                tAnswer (34),
                tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
                bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
                noCGencountered (0),
                manualCGencountered (1),
                scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
                unknown (0),
                dialPulse (1),
                dtmf (2),
                isdn (3),
                isdnNoDtmf (4),
                spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore  (0),
                        abort   (1)} DEFAULT ignore ,
                    value       [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        calledPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
        calledPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
        calledFacilityGroup [3] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup  [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        calledFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
        releaseCause [5] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
        extensions [6] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type  } OPTIONAL,
        connectTime [7] IMPLICIT INTEGER (0..2147483647) OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 13

termAttemptAuthorized OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
```

```
        miscCallInfo [1] IMPLICIT SEQUENCE {
            messageType [0] IMPLICIT ENUMERATED {
                request (0),
                notification (1)},
            dpAssignment [1] IMPLICIT ENUMERATED {
                individualLine (0),
                groupBased (1),
                officeBased (2)} OPTIONAL},
        triggerType [2] IMPLICIT ENUMERATED {
            featureActivation (0),
            verticalServiceCode (1),
            customizedAccess (2),
            customizedIntercom (3),
            emergencyService (12),
            aFR (13),
            sharedIOTrunk (14),
            offHookDelay (17),
            channelSetupPRI (18),
            tNoAnswer (25),
            tBusy (26),
            oCalledPartyBusy (27),
            oNoAnswer (29),
            originationAttemptAuthorized (30),
            oAnswer (31),
            oDisconnect (32),
            termAttemptAuthorized (33),
            tAnswer (34),
            tDisconnect (35)} OPTIONAL},
    bearerCapability [1] CHOICE {
        bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
        tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
    calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
    iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
    cGEncountered [8] IMPLICIT ENUMERATED {
        noCGencountered (0),
        manualCGencountered (1),
        scpOverload (2)} OPTIONAL,
    locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
    terminalType [11] IMPLICIT ENUMERATED {
        unknown (0),
        dialPulse (1),
        dtmf (2),
        isdn (3),
        isdnNoDtmf (4),
        spare (16)} OPTIONAL,
    extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
        SEQUENCE {
            type INTEGER,
            criticality ENUMERATED {
                ignore  (0),
                abort   (1)} DEFAULT ignore ,
            value       [1] ANY DEFINED BY type   } OPTIONAL,
    chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
    servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
calledPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
calledPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
callingPartyBusinessGroupID [3] IMPLICIT OCTET STRING OPTIONAL,
originalCalledPartyID [4] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
redirectingPartyID [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
redirectionInformation [6] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
routeList [7] IMPLICIT SEQUENCE SIZE (1..3) OF
    OCTET STRING (SIZE (??..??)) OPTIONAL,
travellingClassMark [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL
extensions [9] IMPLICIT SEQUENCE SIZE (1..??) OF
```

```
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 9

tMidCall OPERATION
   ARGUMENT
      SEQUENCE {
         dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
            serviceAddressInformation [0] IMPLICIT SEQUENCE {
               serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
               miscCallInfo [1] IMPLICIT SEQUENCE {
                  messageType [0] IMPLICIT ENUMERATED {
                     request (0),
                     notification (1)},
                  dpAssignment [1] IMPLICIT ENUMERATED {
                     individualLine (0),
                     groupBased (1),
                     officeBased (2)} OPTIONAL},
               triggerType [2] IMPLICIT ENUMERATED {
                  featureActivation (0),
                  verticalServiceCode (1),
                  customizedAccess (2),
                  customizedIntercom (3),
                  emergencyService (12),
                  aFR (13),
                  sharedIOTrunk (14),
                  offHookDelay (17),
                  channelSetupPRI (18),
                  tNoAnswer (25),
                  tBusy (26),
                  oCalledPartyBusy (27),
                  oNoAnswer (29),
                  originationAttemptAuthorized  (30),
                  oAnswer (31),
                  oDisconnect (32),
                  termAttemptAuthorized (33),
                  tAnswer (34),
                  tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
               bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
               tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
               noCGencountered (0),
               manualCGencountered (1),
               scpOverload  (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
```

```
            terminalType [11] IMPLICIT ENUMERATED {
                unknown (0),
                dialPulse (1),
                dtmf (2),
                isdn (3),
                isdnNoDtmf (4),
                spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore  (0),
                        abort   (1)} DEFAULT ignore ,
                    value       [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        calledPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
        calledPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
        callingPartyBusinessGroupID [3] IMPLICIT OCTET STRING OPTIONAL,
        callingPartySubaddress [4] IMPLICIT OCTET STRING OPTIONAL,
        featureRequestIndicator [5] IMPLICIT ENUMERATED {
            hold (0),
            retrieve (1),
            featureActivation (2),
            spare1 (3),
            sparen (127)} OPTIONAL,
        extensions [6] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL,
        carrier [7] IMPLICIT OCTET STRING OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 15

tNoAnswer OPERATION
    ARGUMENT
        SEQUENCE {
            dpSpecificCommonParameters [0] IMPLICIT SEQUENCE {
                serviceAddressInformation [0] IMPLICIT SEQUENCE {
                    serviceKey [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                    miscCallInfo [1] IMPLICIT SEQUENCE {
                        messageType [0] IMPLICIT ENUMERATED {
                            request (0),
                            notification (1)},
                        dpAssignment [1] IMPLICIT ENUMERATED {
                            individualLine (0),
                            groupBased (1),
                            officeBased (2)} OPTIONAL},
                    triggerType [2] IMPLICIT ENUMERATED {
                        featureActivation (0),
                        verticalServiceCode (1),
                        customizedAccess  (2),
                        customizedIntercom (3),
                        emergencyService (12),
                        aFR (13),
                        sharedIOTrunk (14),
                        offHookDelay (17),
```

```
                channelSetupPRI (18),
                tNoAnswer (25),
                tBusy (26),
                oCalledPartyBusy (27),
                oNoAnswer  (29),
                originationAttemptAuthorized (30),
                oAnswer (31),
                oDisconnect (32),
                termAttemptAuthorized (33),
                tAnswer (34),
                tDisconnect (35)} OPTIONAL},
            bearerCapability [1] CHOICE {
                bearerCap [0] IMPLICIT OCTET STRING (SIZE (2..??)),
                tmr [1] IMPLICIT OCTET STRING (SIZE (1))} OPTIONAL,
            calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartyNumber [3] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            callingPartysCategory [4] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
            iPSSPCapabilities [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iPAvailable [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            iSDNAccessRelatedInformation [7] IMPLICIT OCTET STRING OPTIONAL,
            cGEncountered [8] IMPLICIT ENUMERATED {
                noCGencountered  (0),
                manualCGencountered  (1),
                scpOverload (2)} OPTIONAL,
            locationNumber [9] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            serviceProfileIdentifier [10] IMPLICIT OCTET STRING OPTIONAL,
            terminalType [11] IMPLICIT ENUMERATED {
                unknown (0),
                dialPulse (1),
                dtmf (2),
                isdn (3),
                isdnNoDtmf (4),
                spare (16)} OPTIONAL,
            extensions [12] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {
                    type INTEGER,
                    criticality ENUMERATED {
                        ignore  (0),
                        abort   (1)} DEFAULT ignore ,
                    value       [1] ANY DEFINED BY type   } OPTIONAL,
            chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
            servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
        calledPartyBusinessGroupID [1] IMPLICIT OCTET STRING OPTIONAL,
        calledPartySubaddress [2] IMPLICIT OCTET STRING OPTIONAL,
        calledFacilityGroup [3] CHOICE {
            trunkGroupID [0] IMPLICIT INTEGER,
            privateFacilityID [1] IMPLICIT INTEGER,
            huntGroup [2] IMPLICIT OCTET STRING,
            routeIndex [3] IMPLICIT OCTET STRING} OPTIONAL,
        calledFacilityGroupMember [4] IMPLICIT INTEGER OPTIONAL,
        originalCalledPartyID [5] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectingPartyID [6] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        redirectionInformation [7] IMPLICIT OCTET STRING (SIZE (2)) OPTIONAL,
        travellingClassMark [8] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        extensions [9] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {
                type INTEGER,
                criticality ENUMERATED {
                    ignore      (0),
                    abort       (1)} DEFAULT ignore ,
                value           [1] ANY DEFINED BY type   } OPTIONAL}
ERRORS {
-- missingCustomerRecord -- localValue 6,
-- missingParameter -- localValue 7,
-- parameterOutOfRange -- localValue 8,
-- systemFailure -- localValue 11,
-- taskRefused -- localValue 12,
-- unexpectedComponentSequence -- localValue 14,
```

```
-- unexpectedDataValue -- localValue 15,
-- unexpectedParameter -- localValue 16}
 ::= localValue 11

canceled ERROR
 ::= localValue 0

cancelFailed ERROR
    PARAMETER
        SEQUENCE {
            problem [0] IMPLICIT ENUMERATED {
                unknownOperation (0),
                tooLate (1),
                operationNotCancellable (2)},
            operation [1] IMPLICIT INTEGER (-128..127)}
 ::= localValue 1

eTCFailed ERROR
 ::= localValue 3

improperCallerResponse ERROR
 ::= localValue 4

missingCustomerRecord ERROR
 ::= localValue 6

missingParameter ERROR
 ::= localValue 7

parameterOutOfRange ERROR
 ::= localValue 8

requestedInfoError ERROR
    PARAMETER
        ENUMERATED {
            unknownRequestedInfo (1),
            requestedInfoNotAvailable (2)}
 ::= localValue 10

systemFailure ERROR
    PARAMETER
        unavailableNetworkResource ENUMERATED {
            unavailableResources (0),
            componentFailure (1),
            basicCallProcessingException (2),
            resourceStatusFailure (3),
            endUserFailure (4)}
 ::= localValue 11

taskRefused ERROR
    PARAMETER
        ENUMERATED {
            generic (0),
            unobtainable (1),
            congestion  (2)}
 ::= localValue 12

unavailableResource ERROR
 ::= localValue 13

unexpectedComponentSequence ERROR
 ::= localValue 14

unexpectedDataValue ERROR
 ::= localValue 15

unexpectedParameter ERROR
 ::= localValue 16

unknownLegID ERROR
 ::= localValue 17

unknownResource ERROR
 ::= localValue 18
```