INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.1228

Fascicle 5/5

(09/97)

SERIES Q: SWITCHING AND SIGNALLING

Intelligent Network

Interface Recommendation for intelligent network Capability Set 2: Parts 5, 6 and 7

ITU-T Recommendation Q.1228 – Fascicle 5/5

(Previously CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.849
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	**Q.1200–Q.1999**
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to ITU-T List of Recommendations.

Recommendation Q.1228

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CAPABILITY SET 2

FASCICLE 5

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the 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.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

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

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Page

PART 3

Page

PART 5

PART 6

PART 7

Recommendation Q.1228

INTERFACE RECOMMENDATION FOR INTELLIGENT NETWORK CAPABILITY SET 2

(Geneva, 1997)

PART 5

ANNEX A.1

Introduction to the INAP CS-1 and CS-2 SDL models

A.1.1 Introduction

The SDLs included in this Annex are a normative reference and considered an integral part of this Recommendation.

The IN CS-1 SDL diagrams do not conform precisely with the behaviour of INAP as specified in Recommendation Q.1218, as some changes have been made for convenience in modelling. The behaviour of INAP as specified in this Recommendation is correctly modelled by the combination of the IN CS-1 and IN CS-2 specifications in Annex A.

The INAP IN CS-1 and IN CS-2 models are specified with SDL 92 in an object-oriented way. The IN CS-2 specification inherits the IN CS-1 and only specifies the difference between IN CS-1 and IN CS-2. Consequently, the architectures of IN CS-1 and IN CS-2 are more or less the same.

The SDL model specifies precisely and unambiguously the behaviour and the interworking between the different functional entities: CS, CSA, SSF-FSM, BCSM. The data structures are not completely specified. They are included in this Recommendation and appropriate references are made in the SDL model.

The model provides a platform for service emulation and the development of test cases based on IN services.

Figure A.1-1 shows the IN CS-1/CS-2 information model. There are zero, one or many Call Segments in one Call Segment Association. There is a one-to-one correspondence between Call Segment and Connection Point, and so on. There are two objects of type BCSM, the OriginatingBCSM and the TerminatingBCSM.

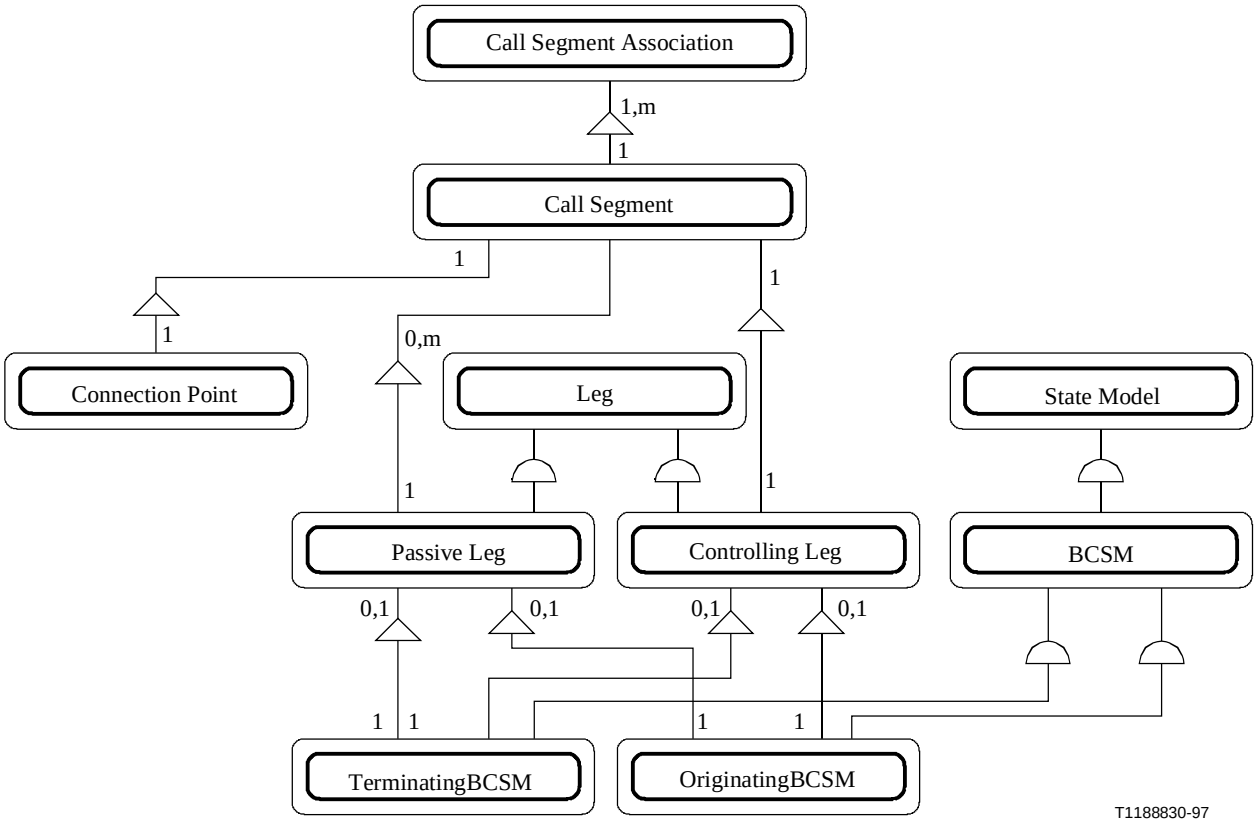


Figure A.1-1/Q.1228 – IN CS-1/CS-2 information model

Figure A.1-2 shows the SDL model of IN CS-1 SSF/CCF. The IN CS-2 equivalent is shown in Figure A.1-5. It inherits most of the items of the IN CS-1 model and refines them. The objects in the information model of Figure A.1-1 map to the SDL model of Figure A.1-2 in the following way.

The Call Segment Association object is a process type in the SDL model. The CallSegmentAssociation manages:

– the creation of CallSegments; and

– the dialogue with the SCF.

The Call Segment object is divided into two process types in the SDL model: CallSegment and SSF-FSM. The process type CallSegment manages the:

– identifiers of the legs (connection view). This data structure models the Connection Point object;

– creation of the O-BCSMs and T-BCSMs;

– creation of the SSF-FSM;

– filtering of detection points;

– processing of connection view oriented IN operations.

The Call Segment process defines connection view states for the CS and it should not be confused with the SSF for CS in 11.5.2 neither with the CVS approach in Recommendation Q.1224. The reason is that in the Q.1224, the CVS approach gives explicit names to combinations of states of associated CSs. In Q.1228 this would lead to an infinite number of CSA states, since the Q.1224 only represents a very small subset of the protocol states. Instead, in the SDLs the states of each CS is given a name (e.g. Stable-2-Party) and the CSA is represented as a set of associated CSs, possibly in different states.

The process type SSF-FSM manages the:

– processing of IN operations;

– handling of detection points (EDPs and TDPs).

In order to centralise the DP processing and thus simplify the model, the SSF-FSM is created even though no TDP may be armed for the call.

This SSF-FSM represented by SDLs is different from the FSM for CS as described in 11.5.2. The SSF-FSM in the SDL has three states and handles user interaction in the monitoring state.

The Connection Point object is modelled by a data structure as described above.

The Leg object is also a data structure within the process type CallSegment. It is identical to the data structures of Passive Leg and Controlling Leg. The data structure contains the status of the leg and a pointer to the BCSM connected to the leg.

Terminating BCSM and Originating BCSM are both objects of the class BCSM in the information model. The SDL model is not exactly constructed this way. Since the O-BCSM and T-BCSM significantly differ, they are modelled in two completely independent process types.

The SDL model contains one additional process type: InterfaceHandler. The InterfaceHandler is the permanent manager of the call control function. When the interpretation of the SDL specification begins, the IH is the only process that exists. Then during the course of a call setup, the IH, after having received the appropriated messages from the environment, creates the call segment association. It also handles the dialogue with the SCF and passes the primitives between the Signalling Control Interface and the CSAs and between the SCF interface and the CSAs.

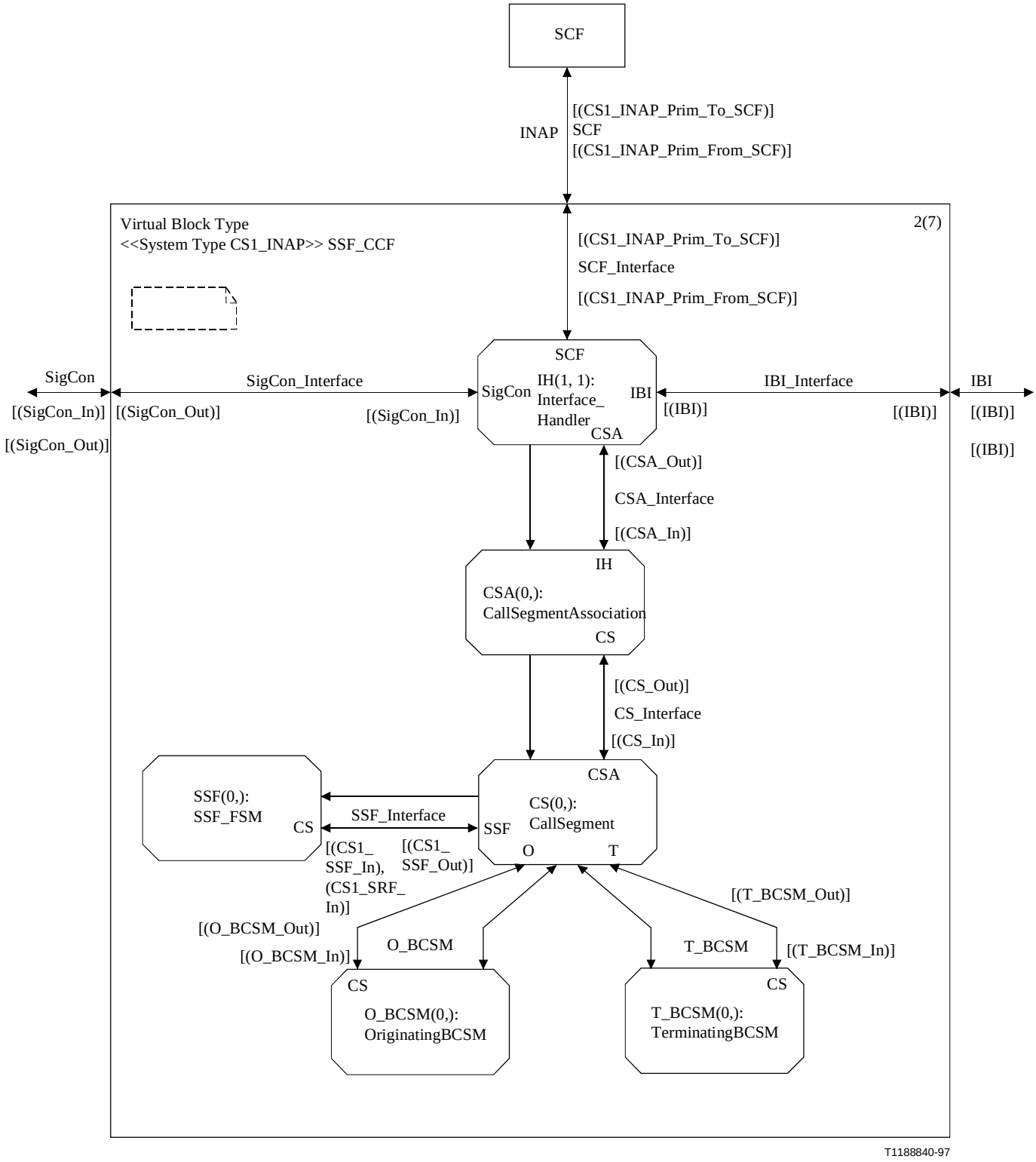


Figure A.1-2/Q.1228 – The INAP IN CS-1 SDL model, half-call view

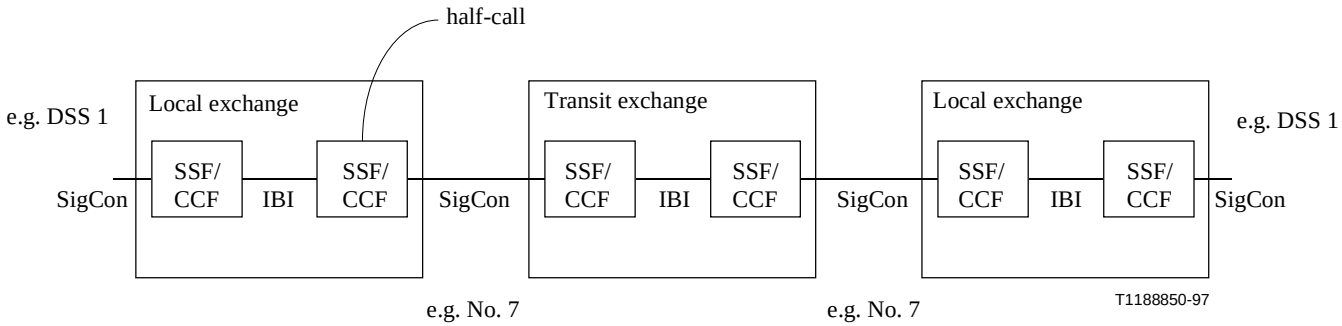


Figure A.1-3/Q.1228 – General example scenario for INAP showing the SSF/CCF only

Figure A.1-3 shows a general example scenario for IN related to the SDL model. The IBI interface is an internal intra BCSM interface between two half calls.

The full SDL model consists of two half calls as indicated in Figure A.1-4. Figure A.1-2 is a refinement of the SSF/CCF block. It shows one half call under control of the SCF, including the service switching and call control functions. In order to operate the model, the Block SSF_CCF needs to be doubled to get two half calls. In this way, the full functionality of the interworking between the O-BCSM and T-BCSM can be simulated.

The model has three interfaces. The SigCon interface could, for example, be a DSS 1 interface. The Messages on SigCon are abstract and have to be mapped to a real protocol, e.g. DSS 1, in a concrete case. IBI is completely internal to a switch. The INAP interface to the SCF is the one with the standardized INAP messages.

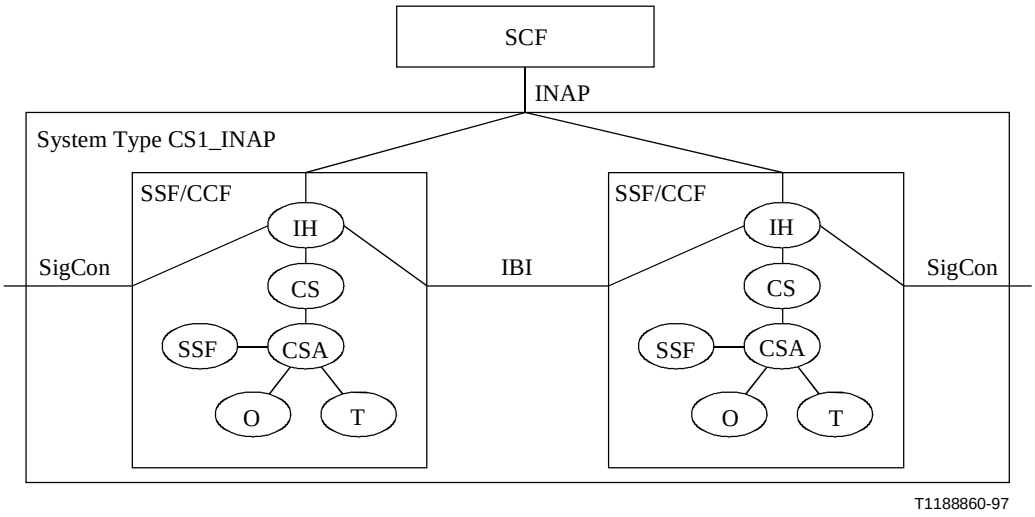


Figure A.1-4/Q.1228 – Two half calls

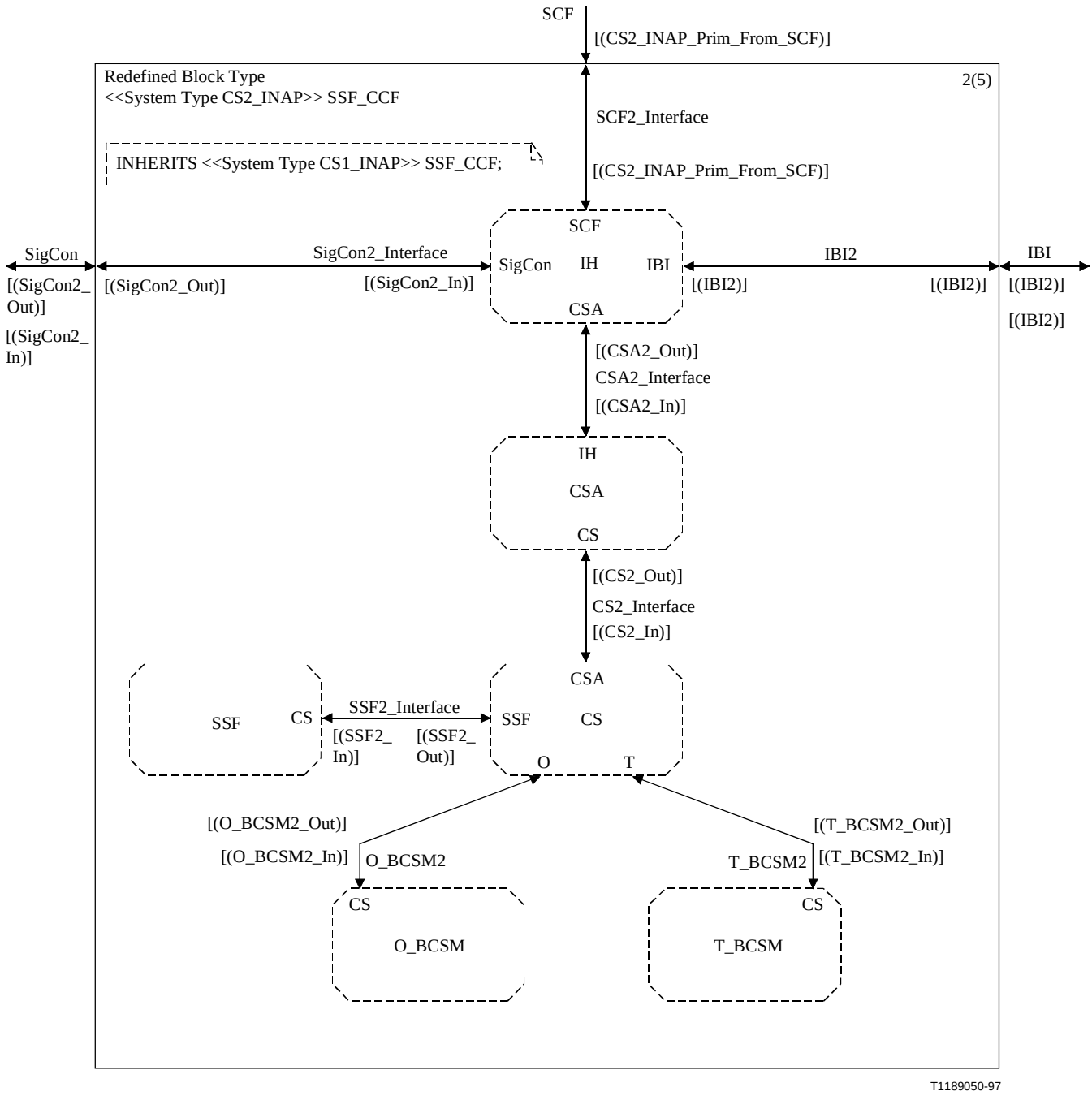


Figure A.1-5/Q.1228 – INAP IN CS-2 model which inherits the IN CS-1

A.1.2 Example for the interworking of the SSF/CCF SDL processes

The following Message Sequence Chart (MSC) shows the interworking between the environment, IH, CSA, CS, SSF and O_BCSM after a SetupInd is received through the SigCon interface. For brevity, only the first few signal exchanges are shown. See Figure A.1-6.

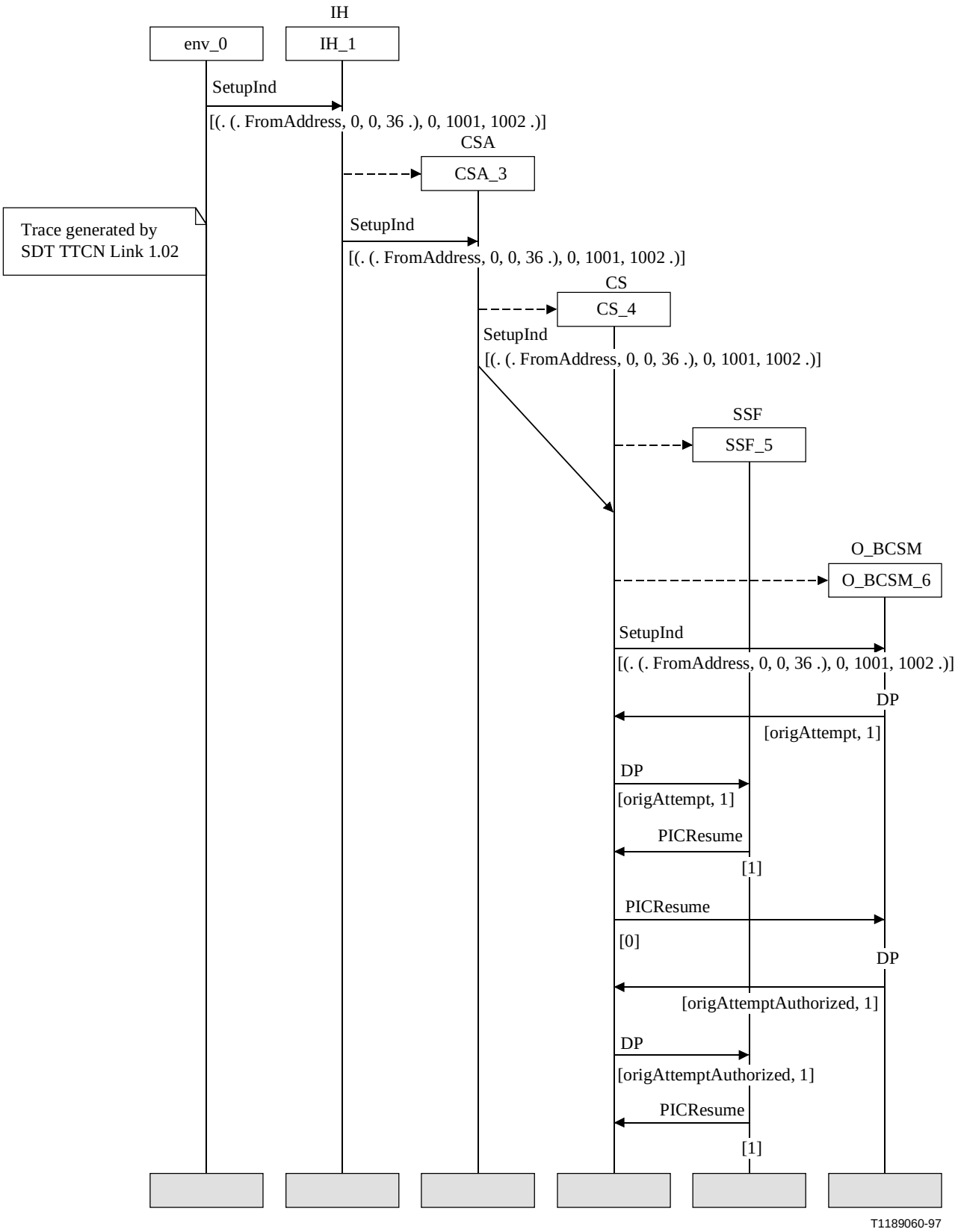


Figure A.1-6/Q.1228 – Message Sequence Chart (MSC) for the interworking of the SSF/CCF SDL processes

A.1.3 Example for the Three-Party Call setup as seen from the environment

The following Message Sequence Chart (MSC) shows an example of the interworking between the SDL model and the environment during a Three-Party Call. The MSC is developed according to the architecture of Figure A.1-4. The internal message exchange of the two half calls CS2_INAP model is not shown in the MSC. There are three local exchanges involved: A, B and C. From the point of view of the MSC they are all environments, together with the SCF. For the sake of clarity these environments are each put on a different process instance axis, each one corresponding to one Point of Control and Observation (PCO). See Figures A.1-7 and A.1-8.

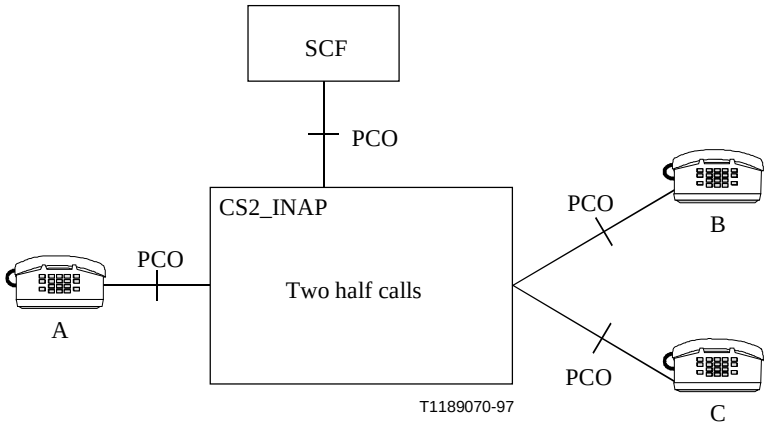


Figure A.1-7/Q.1228 – Three-Party Call Setup – Two half calls

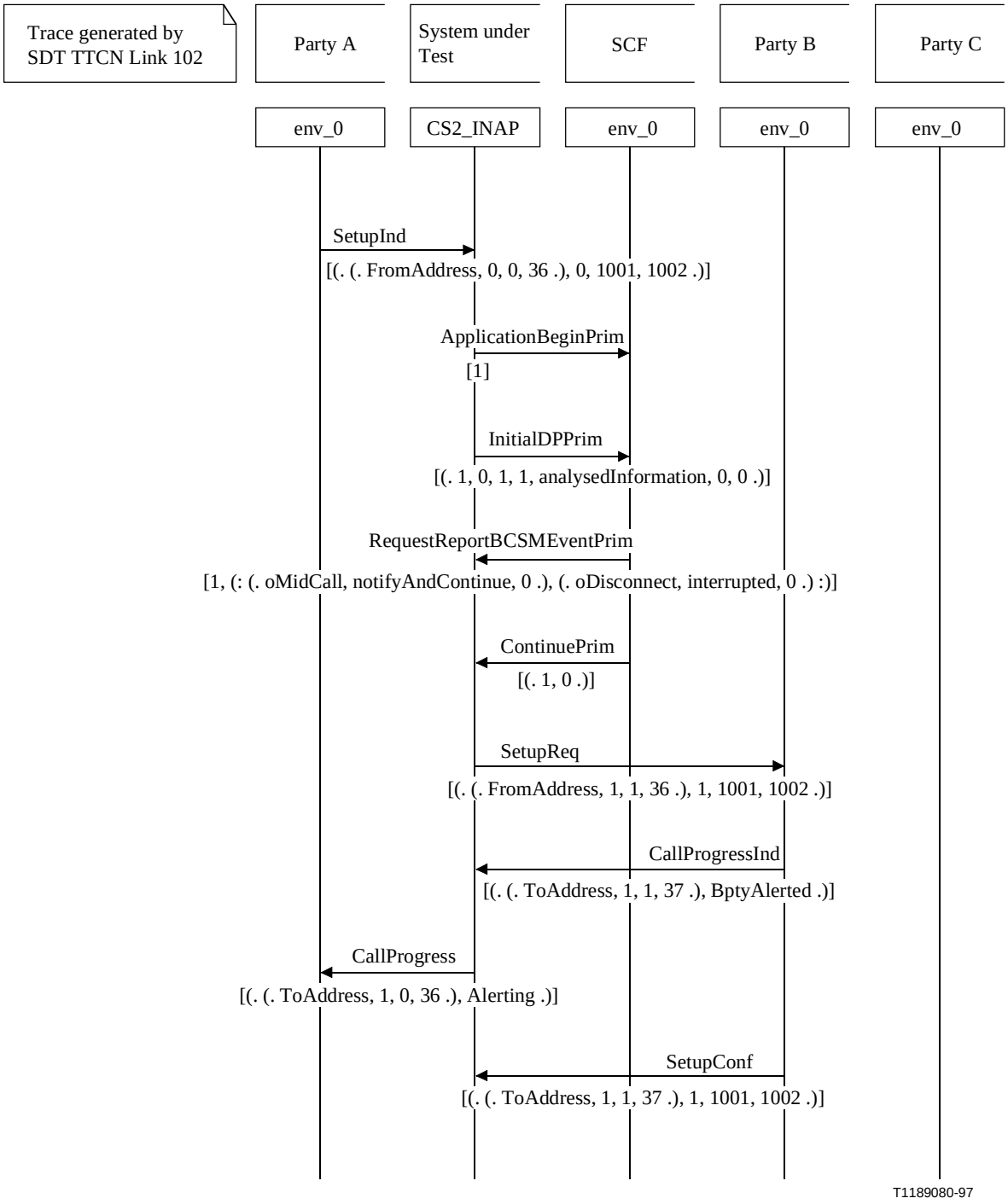


Figure A.1-8/Q.1228 (sheet 1 of 2) – Message Sequence Chart (MSC) – Three-Party Call

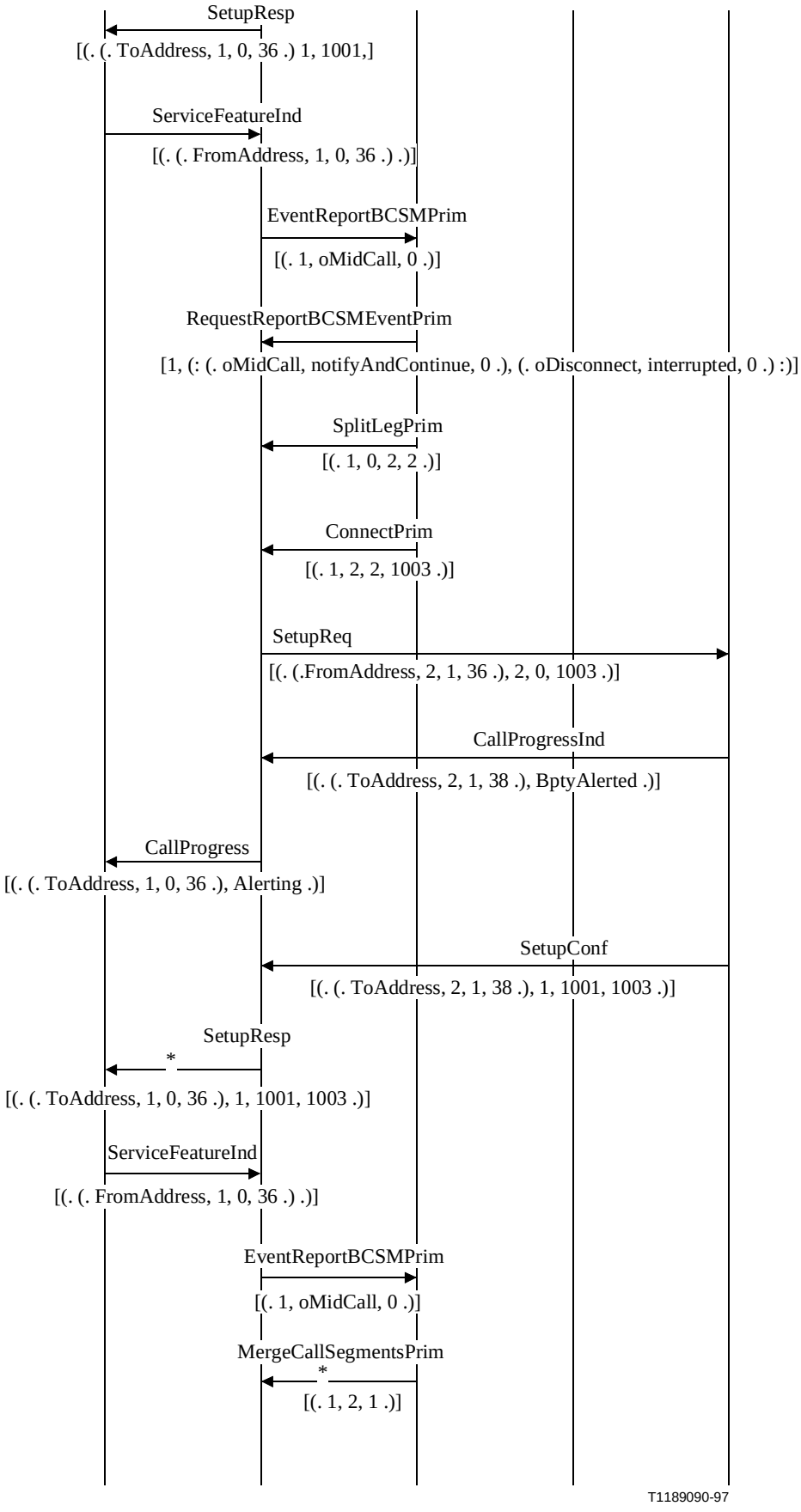


Figure A.1-8/Q.1228 (sheet 2 of 2) – Message Sequence Chart (MSC) – Three-Party Call

ANNEX A.2

Transition diagrams

A.2.1 Call Segment Association transition diagram

A.2.1.1 Definition of states and transitions

The states identified for CSACV are:

Idle/Null: The condition (idle) indicates that no CSACV instance is existing and may be created, this condition occurs when a particular CSACV instance is deleted. The state (Null) represents the condition that a call segment association is created without any call segment and a dialogue is opened with the service logic.

One Segment: This state represents a call segment association containing one call segment.

Multi-Segment: This state represents a call segment association containing multiple call segments.

A.2.1.2 General functional procedures

A.2.1.2.1 Queueing of BCSM events on receipt of an RRBE operation in the CSA

The armed events have to be stored into a queue associated with the CSA instance for those events received in an RRBCSM operation for which no passive leg is created. When an operation Connect or InitiateCallAttempt is subsequently sent, then the armed events shall be retrieved and transferred to the exiting or newly-created CS instance at the moment the leg is created. The armed events have to be stored at the CSA level and not in the initial created CS because it can happen that this CS is deleted and also the associated queue.

A.2.1.3 Finite State Model for Call Segment Association Connection View (CSACV)

The Finite State Model for the Call Segment Association Connection View (CSACV) is given in Figure A.2-1.

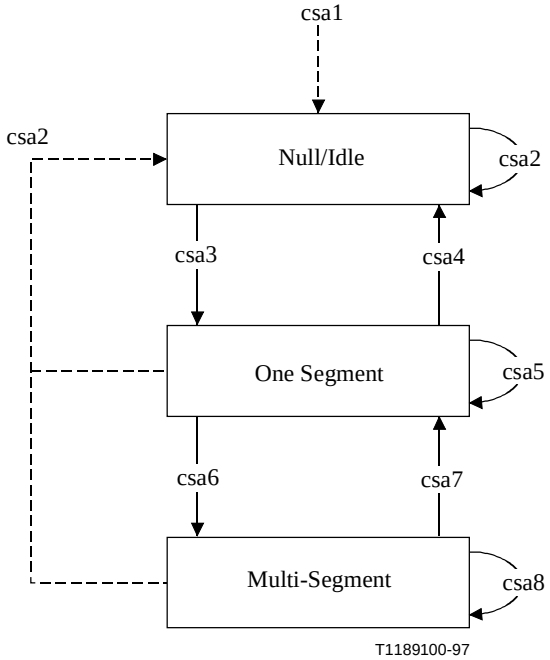


Figure A.2-1/Q.1228 – Call Segment Association Connection View FSM

The CSACV SSF state diagram contains the following events:

csa1: Create Call Segment Association.

csa2: Delete Call segment Association.

csa3: SetupInd, SetupReqInd, InitiateCallAttempt.

csa4: ReleaseCall.

csa5: Reconnect, DisconnectLeg.

csa6: SplitLeg, InitiateCallAttempt.

csa7: ReleaseCall, MergeCallSegments.

csa8: SplitLeg, InitiateCallAttempt, MergeCallSegments, Reconnect, DisconnectLeg, MoveLeg.

A.2.2 Call Segment transition diagram

A.2.2.1 Definition's of states and transitions

The states identified for Call Segment Connection View (CSCV) are:

Idle/Null: The condition "Idle" indicates that no CSCV instance is existing and may be created, this condition occurs when a particular CSCV instance is deleted. The state "Null" represents the condition where call processing is not active and there is no controlling leg or passive leg connected to the connection point.

Originating Setup: This state represents an originating two-party in the setup phase with the controlling leg having the "joined" status and a "pending" passive leg with an associated O_BCSM.

Terminating Setup: This state represents a terminating two-party call in the setup phase with the controlling leg having the "pending" status and a "joined" passive leg with an associated T_BCSM.

Stable 2-Party: This state represents a stable or clearing two-party call, and is either an originating or a terminating call from the perspective of the controlling user with a "joined" controlling leg and a "joined" passive leg with either an O_BCSM or T_BCSM associated with it.

1-Party: This state represents a 1-party call being originated with the controlling leg having the "joined" status and no passive leg. The BCSM is associated with the controlling leg since no passive leg is connected to the CP (Connection Point).

Originating 1-Party Setup: This state represents a 1-party call being originated on behalf of the network with the controlling leg having the "surrogate" status and a "pending" passive leg status to which an O_BCSM is associated.

Terminating 1-Party Setup: This state represents a 1-party call being terminated on behalf of the network with the controlling leg having the "surrogate" status and a "joined" passive leg status to which an T_BCSM is associated.

Stable 1-Party: This state represents a 1-party call originated on behalf of the network; with the controlling leg status in "surrogate" and a "joined" passive leg with either an O_BCSM or T_BCSM associated with it , that is in a stable or clearing phase.

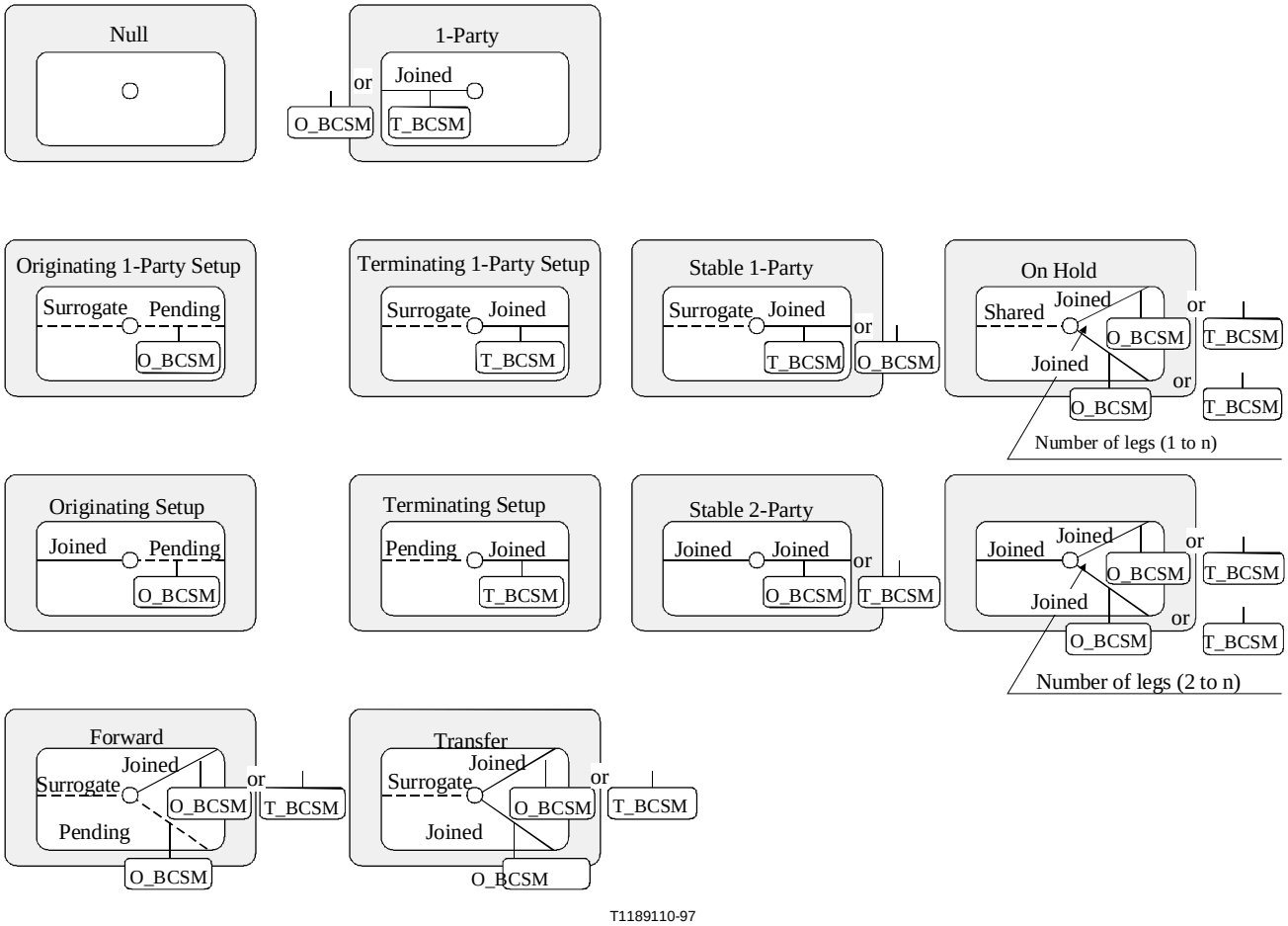
Forward: This state represents a forwarded call. Call processing for the originally calling party passive legs are either in an originating or terminating call setup/stable phase, whereas call processing for the forwarded-to passive leg is in an originating call setup phase. The call segment has a "surrogate" controlling leg with one or two calling party "joined" passive legs with either an associated O_BCSM or T_BCSM, and a "pending" forwarded-to passive leg with an associated O_BCSM.

Transfer: This state represents a transferred call. The call between the involved passive legs is in the stable or clearing phase.

On Hold: This state represents a held call, where the controlling user has put the involved remote parties on hold and is participating in another call.

Stable Multi-Party: This state represents a stable or clearing multi-party call.

The states identified for Call Segment Connection View (CSCV) are given in Figure A.2-2.



T1189110-97

Figure A.2-2/Q.1228 – States of the CSCV

A.2.2.2 General functional procedures for the CSCV

The following general principles shall apply:

A.2.2.2.1 BCSM type indication

An "BCSM type" attribute shall be allocated to a controlling leg at the moment this leg is created, indicating if the controlling leg belongs to an Originating or a Terminating BCSM, i.e. an indication for the type of BCSM in which call triggering occurred.

When a controlling leg is left in a CS, the "BCSM type" attribute shall determine which BCSM shall apply. The conditions of the leg i.e. the armed EDPs, the ApplyChargingReport pending, the EventNotificationCharging pending, and the CallInformationReport pending are also applied to the same leg after creation of an BCSM instance (e.g. after a SplitLeg). The latter entails that if no EDPs and/or reports were armed for the leg in relation to the created BCSM type, then an explicit arming is to be made using, for example, the RequestReportBCSMEvent operation, if needed.

In order to obtain full alignment with the Service Logic (SLP) design in the SCF no mapping of events from O_BCSM to T_BCSM events and vice versa shall occur. This means that the DP at which call processing is to be suspended shall select in the associated BCSM instance the actual state of the call as seen from the legs viewpoint; e.g. if an "active call PIC" exists then the BCSM instance shall be put in MidCall DP of the active BCSM state when the leg is left in a CS after, for example, a DisconnectLeg or SplitLeg operation. The following is noted for the O_BCSM and T_BCSM instances:

- When the O_BCSM instance is put in the O_Mid_Call of the O_Active state then:
  - a transition to the Collect_Information, Analyse_Information or Select_Route is possible by means of sending the appropriate operations from the SCF. This allows to support follow-on calls in accordance with the extended BCSM transitions as described in Recommendation Q.1224.
- When the T_BCSM instance is put in the T_Mid_Call of the T_Active state then:
  - In order to allow to set up an originating call connection to another terminating called party, the ContinueWithArgument followed by the InitiateCallAttempt and the MergeCallSegments operations may be sent from the SCF in order to perform a "transfer call" (see Figure A.2-3). It is noted that the transfer of a call in this case where the controlling leg has the terminating BCSM type attribute is only possible in a serving node exchange (not in a transit exchange).

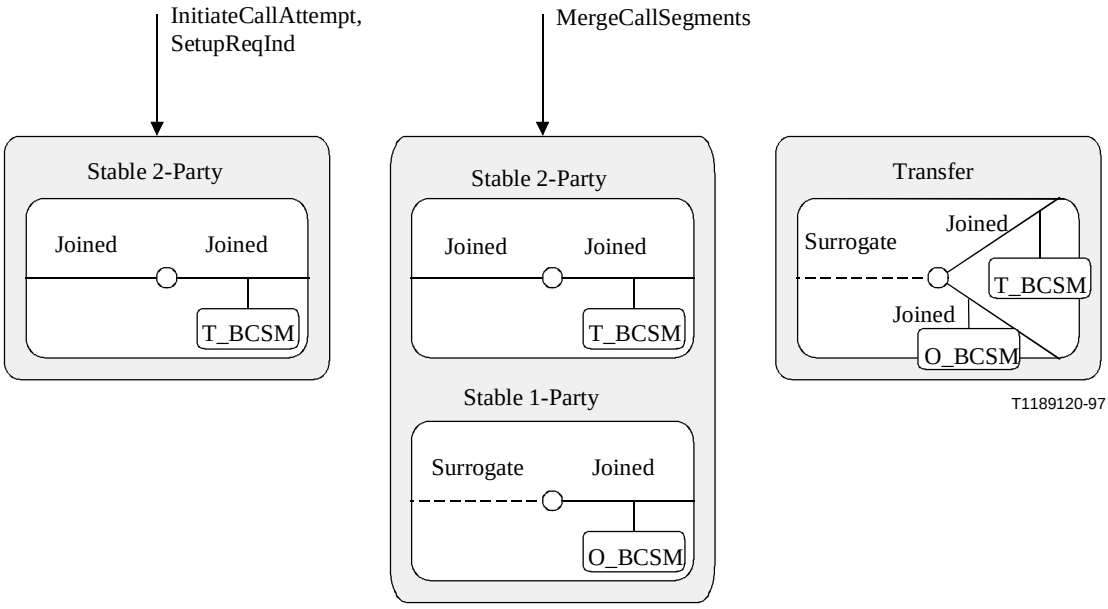


Figure A.2-3/Q.1228 – Transfer Call at T_BCSM

In the above-mentioned cases, the CS shall be put in the 1-Party state with the associated SSF_FSM state in "Waiting For Instructions" for the case where the controlling joined leg is left in a CS due to a SCF operation (DisconnectLeg, SplitLeg, MoveLeg etc.). In order to resume from the O_Mid_Call and T_Mid_Call detection points of the associated BCSM O_Active and T_Active PIC states, a ContinueWithArgument operation can be sent by the service logic.

It should be noted that the problem of "stranding" a "surrogate" controlling leg does not exist as it does for a "Joined" controlling leg, because it is assumed that a "stranded" surrogate leg will be destroyed due to the fact that no voice path is associated with this controlling leg.

A.2.2.2.2 Event handling rules

The following inter-BCSM precedence rules for the reporting to the SCF of events to be signalled for the controlling leg shall apply:

- An event shall only be reported once to the SCF irrelevant of the number of passive legs per CS connected to the controlling leg (via CP).
- A "BCSM type" attribute on the controlling leg indicates if the controlling leg belongs to an Originating or a Terminating BCSM. The "BCSM type" attribute is set to reflect the type of BCSM (originating or terminating) in which the trigger or event occurred at the moment the CS was created.
- If the controlling leg belongs to an Originating BCSM, it is reported as an originating event.
- If the controlling leg belongs to a Terminating BCSM, it is reported as a terminating event.

A.2.2.2.3 Creation of O/T_BCSM based on "BCSM type" attribute

A CS with only a controlling leg (leg status "joined") left is assigned a BCSM instance to supervise the leg. The "BCSM type" attribute set on the controlling leg is used to reflect the type of BCSM (originating or terminating) to be assigned. The corresponding SSF_FSM for the CS shall be put in "Waiting_For_Instructions" state where a leg is left in a CS due to a CPH (e.g. DisconnectLeg) operation and call processing is suspended at the Mid Call DP of the Active state in the associated BCSM.

When the controlling leg becomes connected to the passive leg, no BCSM instance shall be connected to the controlling leg. The BCSM instance which was connected to the controlling leg disappears in case a passive leg is moved (imported) to the Call Segment (e.g. MergeCallSegments operation). If a new passive leg is created (e.g. Connect operation) within the CS, then the existing BCSM instance becomes connected to the passive leg.

An export of a leg due to the execution of a MoveLeg, SplitLeg or MergeCallSegments operation for a passive leg can only be executed when the O_BCSM is at least in the Send_Call PIC and for the T_BCSM in the T_Active PIC. This entails that these operations cannot be executed when the passive leg is in the pending status.

A.2.2.2.4 Release of a Call/Connection

The release of a Call/Connection for a normal call where both call segments are in the "Stable_2_Party" state (with the associated O_BCSM and T_BCSM in the active state) is given in Figure A.2-4.

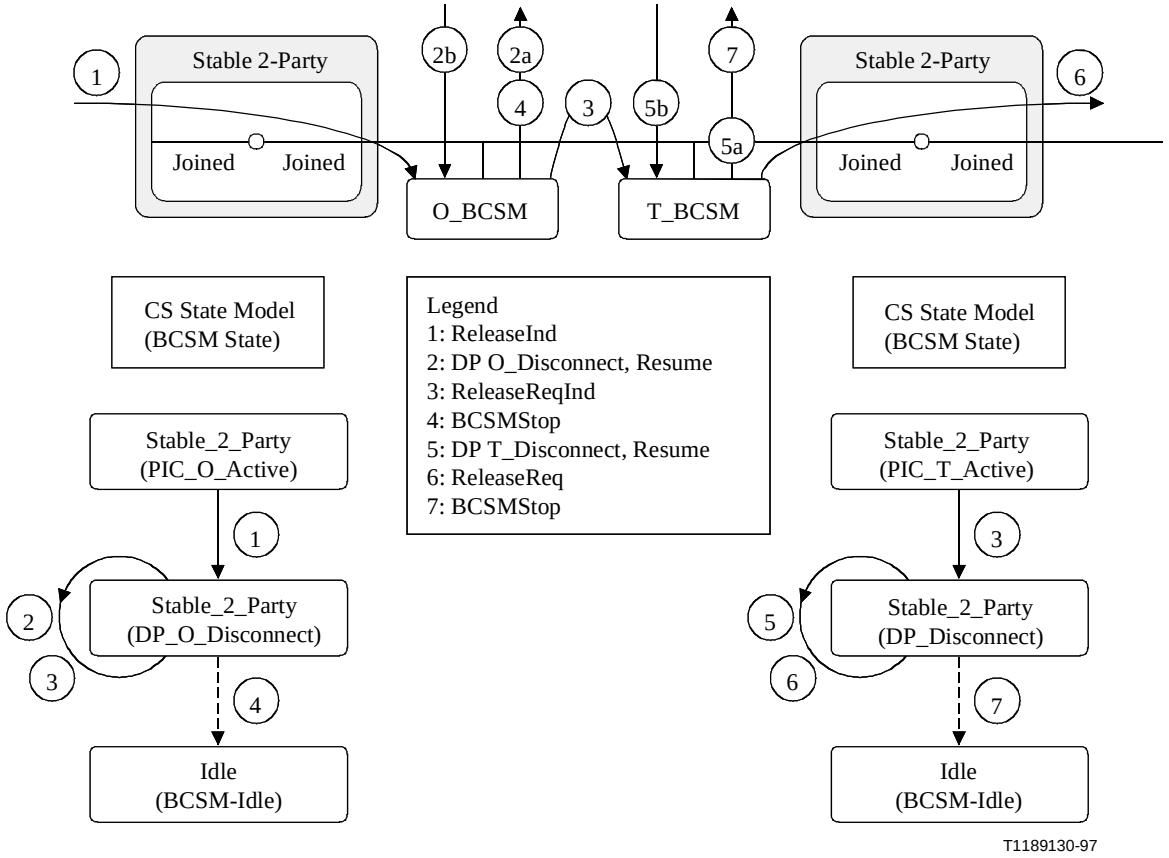


Figure A.2-4/Q.1228 – Release of Call/Connection where call segments are both in "Stable_2_Party"

The release of a Call/Connection for a redirected call where the call segments are in the "Stable_2_Party" state and in the "Forward" states is given in Figure A.2-5.

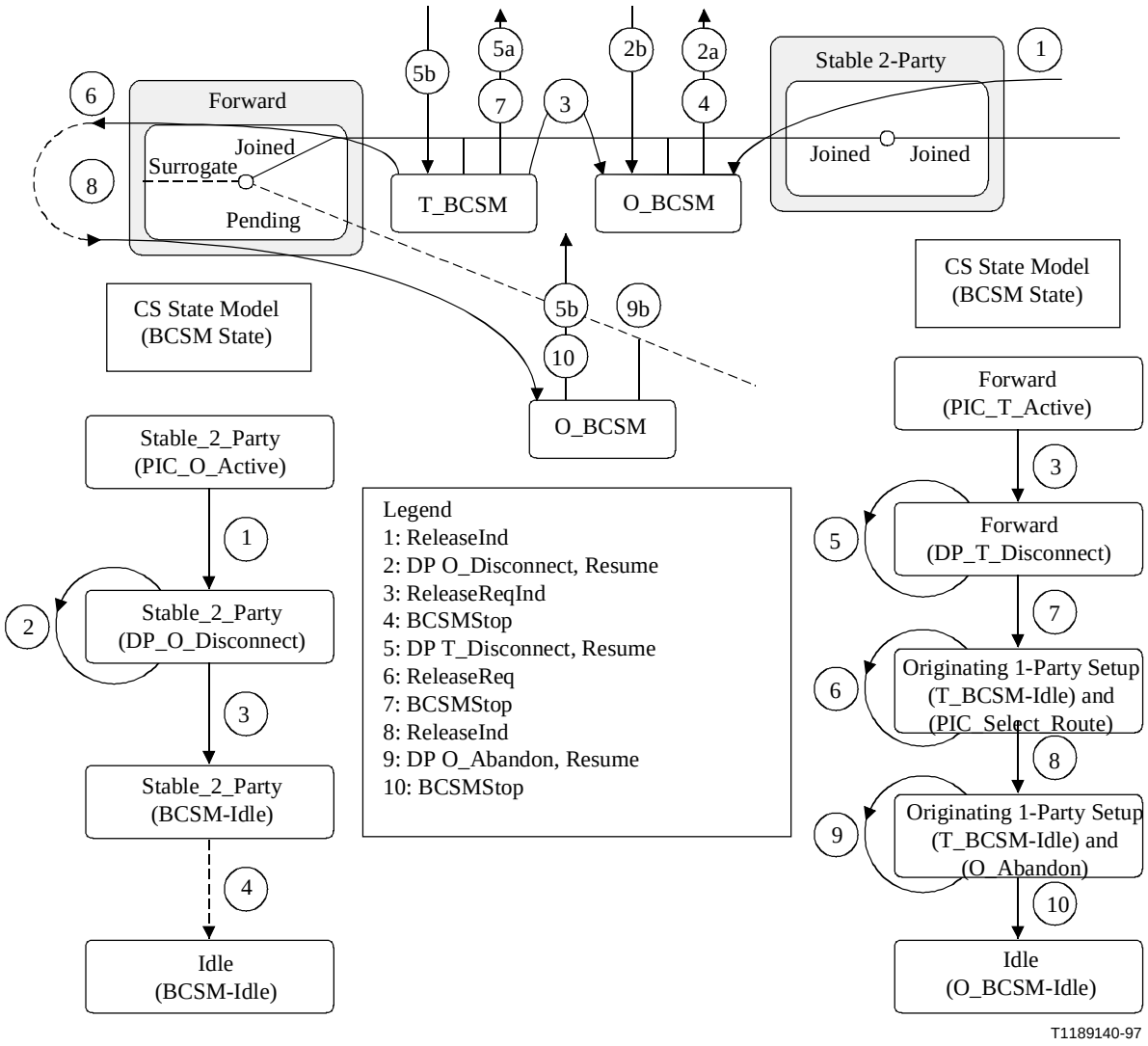


Figure A.2-5/Q.1228 – Release of Call/Connection where the call segments are in the "Stable_2_Party" state and in the "Forward" state

The release of a Call/Connection for a redirected call where the call segments are in the "Stable_2_Party" state and in the "Transfer" states is given in Figures A.2-6a and A.2-6b.

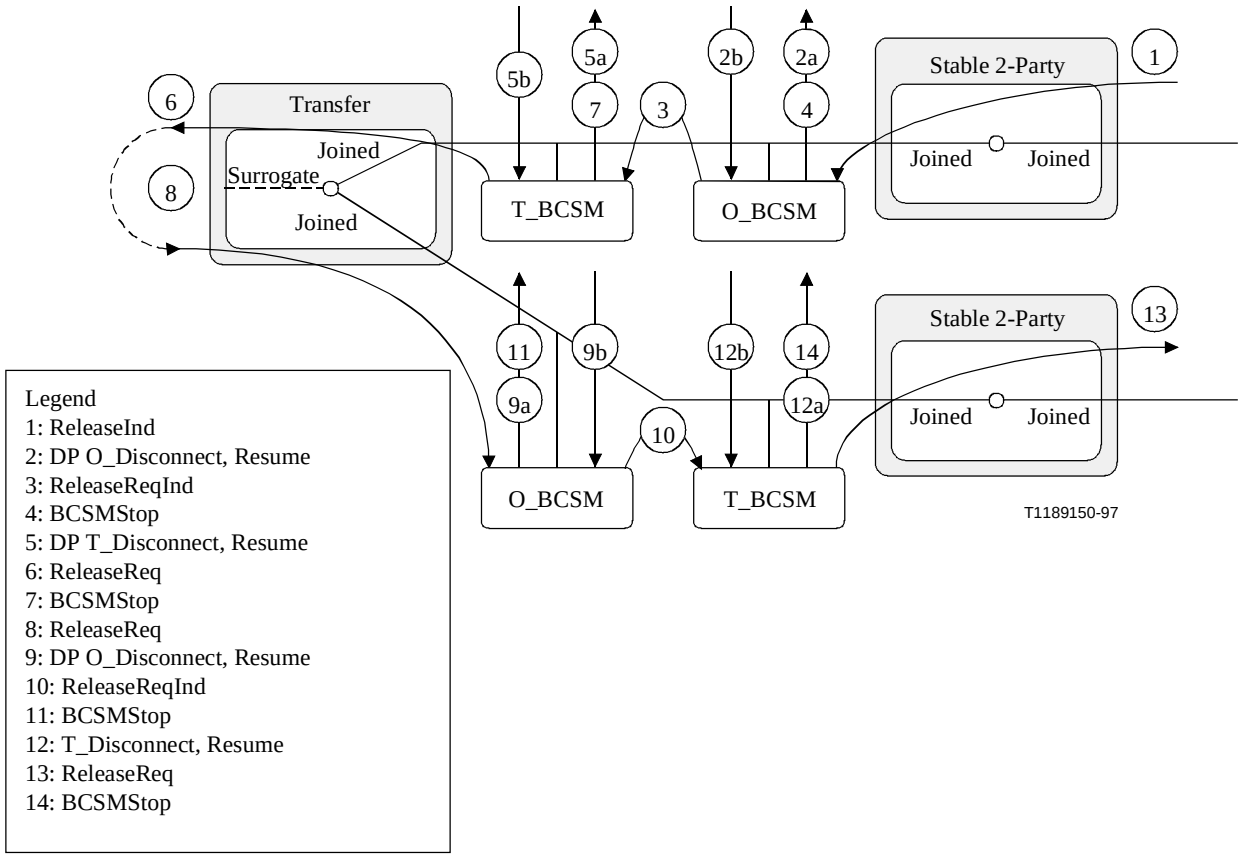


Figure A.2-6a/Q.1228 – Release of Call/Connection where the call segments are in the "Stable_2_Party" state and in the "Transfer" state (information flow)

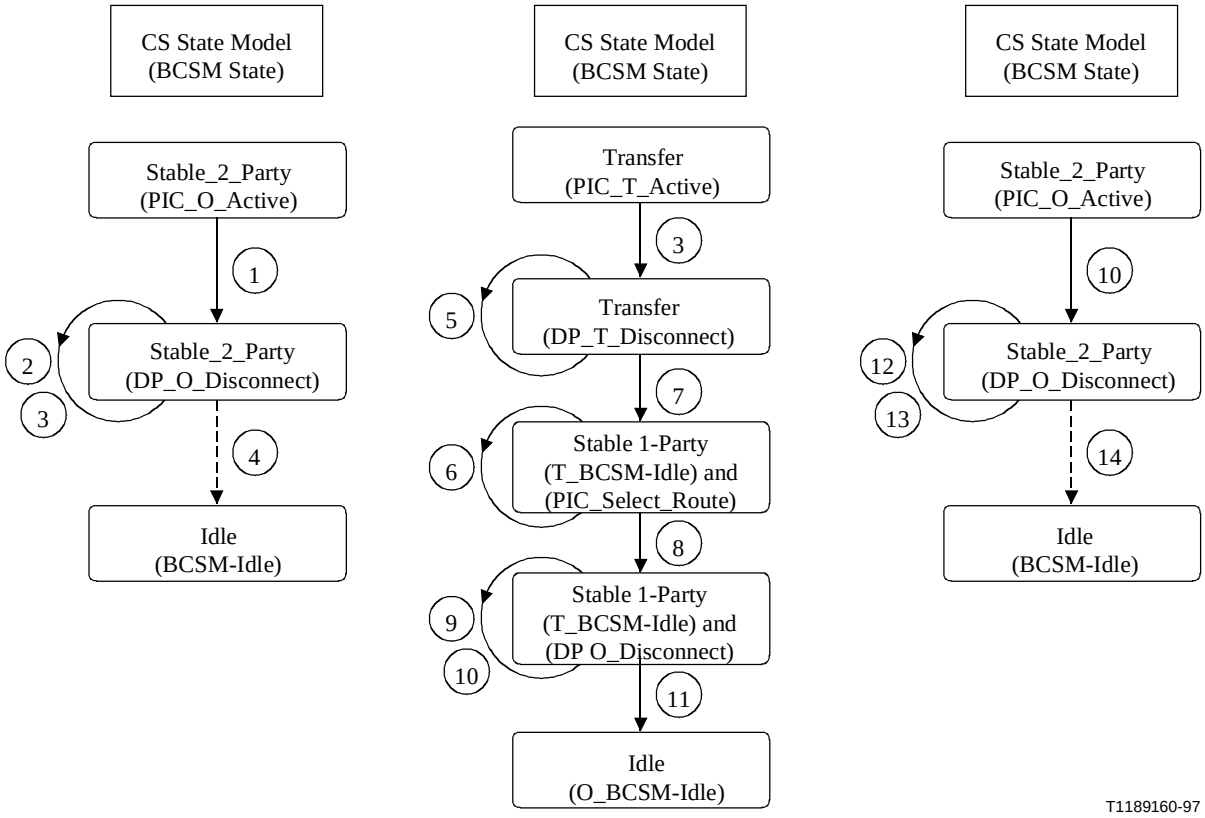


Figure A.2-6b/Q.1228 – Release of Call/Connection where the call segments are in the "Stable_2_Party" state and in the "Transfer" state (state transitions)

A.2.2.2.5 Disconnect Leg operation

A DisconnectLeg operation for a controlling and passive leg shall physically release the specified leg from the connection point towards the remote user. The information flows depicting the release sequences as a result of a DisconnectLeg operation for the controlling (c) and passive legs of a Call/Connection for a normal call where both call segments are in the "Stable_2_Party" state (with the associated O_BCSM and T_BCSM in the active state) are given in Figure A.2-7.

The information flows for the receipt of the following operations are illustrated:

- The following actions occur when a DisconnectLeg operation (for the controlling leg) is received from the service logic:
  - a ReleaseReq signal is sent to the CCAF with a cause value "disconnect controlling leg" in the PIC_O_Disconnect (or PIC_T_Disconnect), and;
  - the BCSM shall transit to the appropriate MidCall_Active PIC sub-state depending on the fact whether a passive leg or not is still associated with the controlling leg.

    NOTE – When a Reconnect Operation is received from the service logic and a remote passive leg is still associated with the controlling leg, then the BCSM shall return to the previous BCSM state (either O_Active, O_Suspended or T_Active) after resumption of the appropriate O_Mid_Call or T_Mid_Call detection (generated as a result of the behaviour for reconnection of a disconnected call).
- The following actions occur when a DisconnectLeg operation (for the passive leg) is received from the service logic:
  - a ReleaseReqInd signal is sent to remote BCSM CCAF with a cause value "disconnect passive leg" in the DP_O_Disconnect (or DP_T_Disconnect), and;

– the BCSM shall transit to the appropriate Disconnect PIC sub-state depending on the fact whether a controlling leg or not is still associated with the passive leg.

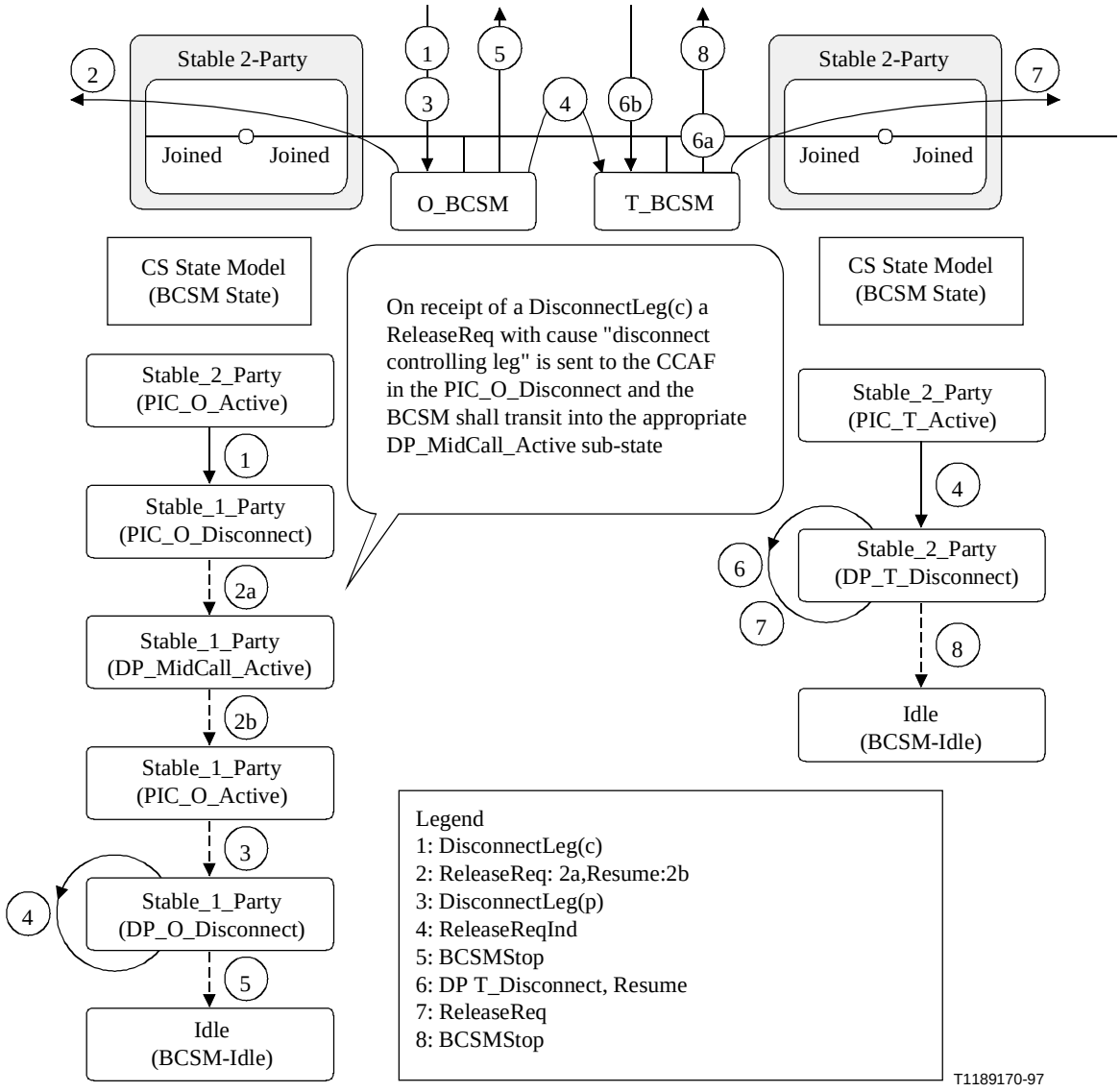


Figure A.2-7/Q.1228 – Disconnect Leg of Call/Connection where call segments are both in "Stable_2_Party"

A.2.2.2.6 User interactions during the SSF-FSM "Monitoring" state

During the "Monitoring" state of the SSF-FSM, it shall be possible to perform user interactions in order to send tones, announcements and display information.

A.2.2.2.7 Leg numbering and associated dp detection rules

i) The definition of the leg identifiers are the following:

- legID = 1 and 2: indicates the party that was present at the moment of the "initialDp" or a DP-specific operation, or the party that was created with the "InitiateCallAttempt" operation. For a CS in originating setup state (with a controlling leg and one passive leg connected to an O_BCSM), the controlling legID will be "1" while the passive leg will be 2. For a CS in terminating setup state (with a controlling leg and one passive leg

connected to an T_BCSM), the controlling legID will be "2" while the passive leg can be "1". It should be noted that legID = 0 can never be used.

- legID = 3 to n: the SCF will assign the leg values.

ii) The detection point arming principle is as follows:

- All events for which filtering applies (abandon, midcall and disconnect) can be armed for the controlling and passive legs depending on what direction (either from the party which is connected to the controlling or passive leg) events have to be captured. As an example, the disconnect DP can be armed for the controlling and the passive leg. In that case if a release request is received from the user, it will be detected by the DisconnectDP armed for the controlling leg, while a release request from the remote parties shall be detected by arming the relevant passive leg DisconnectDP. It should be noted that for the time being, no DPs should be armed for the Midcall and Abandon DPs for the passive legs since only requests can happen from the user connected to the controlling leg.
- All events for which no filtering principles apply shall be armed for the passive leg only.

iii) The following default values for the events to be armed apply for the RequestReportBCSMEvent operation:

- The default value for arming shall be the controlling leg for all events for which the filtering principles apply (abandon, midcall and disconnect).
- The default value for arming shall be the passive leg for all events for which the filtering principles do not apply .

A.2.2.2.8 Call Segment and associated BCSM states for CPH operations

The following principles apply for the Call Segment and associated BCSM states for CPH operations:

- If a CPH operation (SplitLeg , DisconnectLeg, MergeCallSegments, MoveCallSegment or MoveLeg) is received in the Monitoring state, the FSMs for the involved Call Segments shall first transit to the "Waiting for Instruction" state while the associated BCSM instances within the involved Call Segments shall move from the O/T PIC of the corresponding O/T MidCallDP in order to handle subsequent EDP re-arming. It shall be noted that when the BCSM instance is in a DP, it shall stay in this DP after processing of the CPH operation. This involves that only some PIC transitions to MidCall DPs are possible as described in the templates of the operations.
- The receipt of a CPH operation received and/or processed in the state "Waiting for Instructions" shall not cause the change of the state "Waiting for Instructions".
- For these CPH operations which create for the controlling leg a new BCSM, (stranded leg) only this BCSM shall be put into the DP-O/T-MidCall of the active state, the BCSM states of the other involved CSs should transit from the PIC into the corresponding DP state or remain in the DP wait state, the associated CS SSF_FSM for the newly-created BCSM and the other involved CSs shall be put into the "Waiting for Instruction" state so that the EDPs can be re-armed.

All CPH operation sequences shall be finalised by an operation which changes the state into Monitoring (e.g. Continue, ContinueWithArgument, Connect).

A.2.2.2.9 ’Forward’ and ’Transfer’ connection view behaviour principles

The connection view for 'Forward' and 'Transfer' states behaves differently for the case where an operation is received from Terminating Setup (e.g. Call forward service) or Stable_1_Party

(e.g. Meet-Me conference service). Depending on the situation, the signalling events received from one party (one leg) may have to be relayed to the other party. The method of processing the signalling event received from parties is outside the scope of IN CS-2.

A.2.2.2.10 Conventions at the IBI interface

A call flag has to be allocated at the IBI interface for those signals which are bidirectional like ReleaseReqInd, DataReqInd. For the signals which are either from the O_BCSM to T_BCSM (e.g. SetupReqInd) or from T_BCSM to O_BCSM (e.g. ProgressReqInd), no call flag must be inserted.

The following conventions apply for the call flag at the IBI interface:

- The call flag is set to <sender> for the signals generated from the O_BCSM.
- The call flag is set to <receiver> for the signals generated from the T_BCSM.

A.2.2.3 Finite State Model

The transitions for the IN CS-1 Subset of IN CS-2 for the Call Segment Connection View Process Type states are as shown in Figure A.2-8.

The transitions for the IN CS-2 for the Call Segment Connection View Process Type states are as shown in Figure A.2-9.

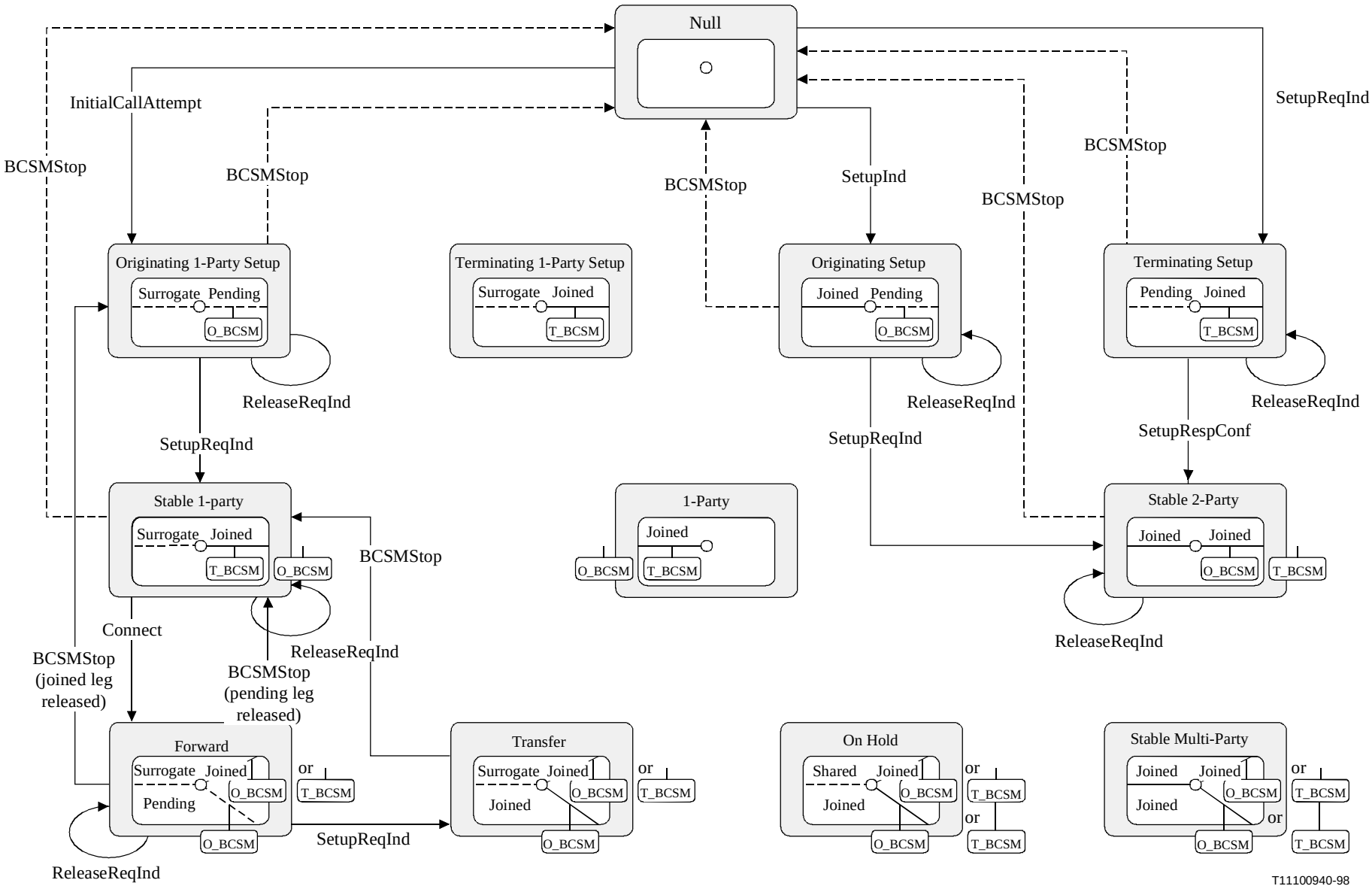


Figure A.2-8/Q.1228 – Finite State Model for the CSCV for the IN CS-1 Subset of IN CS-2

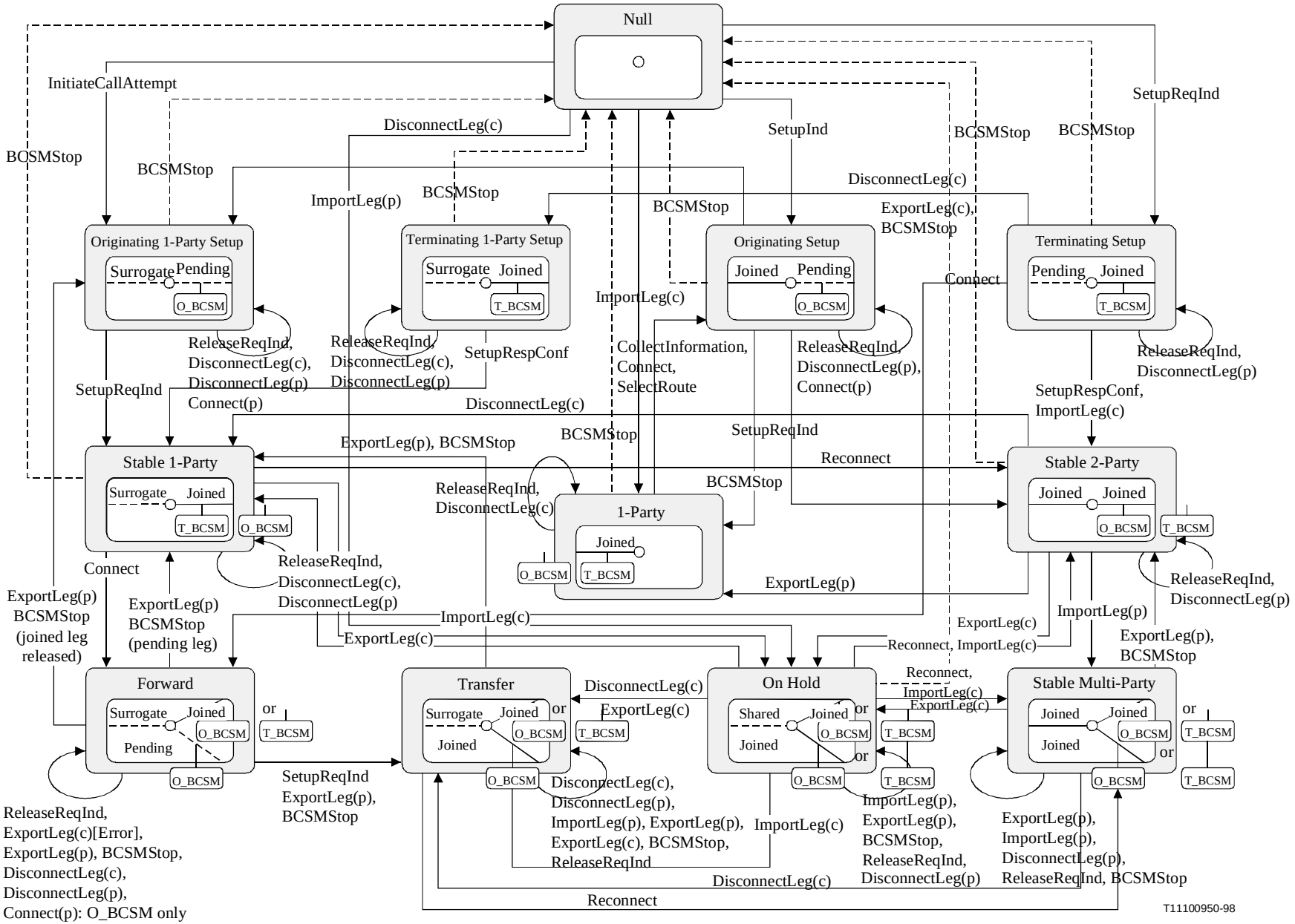


Figure A.2-9/Q.1228 – Finite State Model for CSCV of IN CS-2

ANNEX A.3

SDL Specification of CS-1 SSF/CCF

This Annex is available only in electronic format on the diskettes included in this Fascicle.

PART 6

ANNEX A.4

SDL Specification of CS-2 extensions to SSF/CCF

This Annex is available only in electronic format on the diskettes included in this Fascicle.

PART 7

ANNEX A.5

SDL Specification of CS-2 SRF

This Annex is available only in electronic format on the diskettes included in this Fascicle.

ANNEX A.6

SDL Specification of CS-2 Assist/Hand-off SSF

This Annex is available only in electronic format on the diskettes included in this Fascicle.

ANNEX A.7

SDL Specification of CS-2 CUSF

This Annex is available only in electronic format on the diskettes included in this Fascicle.

ANNEX A.8

SDL Specification of CS-2 SCF

This Annex is available only in electronic format on the diskettes included in this Fascicle.

APPENDIX I

Expanded ASN.1 source

This Appendix contains the expanded ASN.1 source for the common datatypes of Q.1228.

NOTE – The following changes were made to the module IN-CS2-datatypes for the generation of expanded source (only for compiling purposes):

Component: relayedComponent was coded as an OCTET STRING

ExtensionField: was coded as an empty SEQUENCE

TraceItem: the extension marker (...) was removed from the SET type

```
-- Expanded ASN.1 Module 'IN-CS2-datatypes
-- Date: 97-09-23

IN-CS2-datatypes { 0 0 17 1228 modules (0) in-cs2-datatypes (0) version1 (0) }

DEFINITIONS

::=

BEGIN

AccessCode ::= OCTET STRING (SIZE (??..??))

AccountNumber ::= NumericString  (SIZE (1..151))

AChBillingChargingCharacteristics ::= OCTET STRING (SIZE (??..??))

ActionIndicator ::= ENUMERATED {
        activate        (1),
        deactivate  (2),
        retrieve        (3)}

ActionPerformed ::= ENUMERATED {
        activated       (1),
        deactivated  (2),
        alreadyActive  (3),
        alreadyInactive  (4),
        isActive        (5),
        isInactive  (6)}

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

```
AdditionalCallingPartyNumber ::= OCTET STRING (SIZE (??..??))

AlertingPattern ::= OCTET STRING (SIZE (3))

ApplicationTimer ::= INTEGER (0..2047)

AssistingSSPIPRoutingAddress ::= OCTET STRING (SIZE (??..??))

BackwardGVNS ::= OCTET STRING (SIZE (??..??))

BackwardServiceInteractionInd ::= SEQUENCE {
        conferenceTreatmentIndicator [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        callCompletionTreatmentIndicator [2] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL}

BCSMEvent  ::= 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),
                oTermSeized  (19),
                oSuspended  (20),
                tSuspended  (21),
                origAttempt  (22),
                termAttempt  (23),
                oReAnswer   (24),
                tReAnswer   (25),
                facilitySelectedAndAvailable  (26),
                callAccepted  (27)},
        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),
                midCallControlInfo [2] IMPLICIT SEQUENCE SIZE (??..??) OF
                .       SEQUENCE {
                .               midCallInfoType [0] IMPLICIT SEQUENCE {
                .               .       iNServiceControlCodeLow [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                .               .       iNServiceControlCodeHigh [1] IMPLICIT OCTET STRING (SIZE (??..??))
                                        OPTIONAL},
                .               midCallReportType [1] IMPLICIT ENUMERATED {
                .               .       inMonitoringState  (0),
```

```
            .           .       inAnyState  (1)} DEFAULT inMonitoringState }} OPTIONAL}

BCUSMEvent ::= SEQUENCE {
        eventType  [0] IMPLICIT ENUMERATED {
                componentReceived  (127),
                associationReleaseRequested  (126)},
        monitorMode [1] IMPLICIT ENUMERATED {
                interrupted  (0),
                notifyAndContinue  (1),
                transparent  (2)}}

BearerCapabilities ::= BIT STRING {
        speech          (0),
        bc64kbits       (1),
        bc2x64kbits  (2),
        bc384kbits  (3),
        bc1536kbits  (4),
        bc1920kbits  (5),
        multirate       (6),
        restrictedDigitalInfo  (7),
        bc3-1khzAudio  (8),
        bc7khzAudio  (9),
        video           (10)}

BearerCapability ::= CHOICE {
        bearerCap  [0] IMPLICIT OCTET STRING (SIZE (2..??)),
        tmr             [1] IMPLICIT OCTET STRING (SIZE (1))}

BothwayThroughConnectionInd ::= ENUMERATED {
        bothwayPathRequired  (0),
        bothwayPathNotRequired  (1)}

CallConditions ::= CHOICE {
        userAbandon [0] IMPLICIT NULL,
        callFailure [1] IMPLICIT OCTET STRING (SIZE (1)),
        noReply         [2] IMPLICIT INTEGER,
        callRelease [3] IMPLICIT NULL,
        ss-invocation [4] IMPLICIT ENUMERATED {
                callingLineIdentificationRestriction  (1),
                connectedLineIdentificationRestriction  (2),
                callWaiting  (3),
                callHold        (4),
                reverseCharging  (5),
                explicitCallTransfer  (6),
                callCompletionOnBusySubscriber  (7)},
        creditLimitReached [5] IMPLICIT INTEGER,
        callDuration [6] IMPLICIT INTEGER,
        calledNumber [7] CHOICE {
                initialMatch [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                totalMatch [1] IMPLICIT OCTET STRING (SIZE (??..??))},
        answeredCall [8] IMPLICIT NULL}

CalledPartyBusinessGroupID ::= OCTET STRING

CalledPartyNumber ::= OCTET STRING (SIZE (??..??))

CalledPartySubaddress ::= OCTET STRING

CallIdentifier ::= INTEGER (1..2147483647)

CallingPartyBusinessGroupID ::= OCTET STRING
```

```
CallingPartyNumber ::= OCTET STRING (SIZE (??..??))

CallingPartySubaddress ::= OCTET STRING

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

CallProcessingOperationCorrelationID ::= ENUMERATED {
        aLERTing     (1),
        sETUP        (5),
        cONNect      (7),
        dISConnect   (69),
        rELease      (77),
        rELeaseCOMPlete  (90),
        fACility     (98)}

CallRecord ::= SEQUENCE {
        callDuration [0] IMPLICIT INTEGER (–2..86400),
        callingPartyNumber [1] IMPLICIT OCTET STRING (SIZE (??..??)),
        calledPartyNumber [2] IMPLICIT OCTET STRING (SIZE (??..??))}

CallResult ::= OCTET STRING (SIZE (??..??))

CallSegmentID ::= INTEGER (1..??)

initialCallSegment INTEGER ::= 1

CallUnrelatedDpSpecificCommonParameters ::= 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},
        callingPartyNumber [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
        locationNumber [2] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
```

```
        terminalType [3] IMPLICIT ENUMERATED {
                unknown       (0),
                dialPulse     (1),
                dtmf          (2),
                isdn          (3),
                isdnNoDtmf    (4),
                spare         (16)} DEFAULT isdn              ,
        extensions [4] IMPLICIT SEQUENCE SIZE (1..??) OF
                SEQUENCE {} OPTIONAL}

Carrier         ::= OCTET STRING

Cause           ::= OCTET STRING (SIZE (2..??))

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

CGEncountered ::= ENUMERATED {
        noCGencountered  (0),
        manualCGencountered  (1),
        scpOverload  (2)}

ChargeNumber ::= OCTET STRING (SIZE (??..??))

ChargingEvent ::= 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}

ChargingParameters ::= SEQUENCE {
        unitsPerInterval [0] IMPLICIT INTEGER (0..??),
        timePerInterval [1] IMPLICIT INTEGER (0..??),
        scalingFactor [2] IMPLICIT INTEGER (0..??),
        initialUnitIncrement [3] IMPLICIT INTEGER (0..??) OPTIONAL,
        unitsPerDataInterval [4] IMPLICIT INTEGER (0..??) OPTIONAL,
        segmentsPerDataInterval [5] IMPLICIT INTEGER (0..??) OPTIONAL,
        initialTimeInterval [6] IMPLICIT INTEGER (0..??) OPTIONAL}

CollectedDigits ::= 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}

CollectedInfo ::= 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}

Component  ::= CHOICE {
        componentInfo [0] IMPLICIT OCTET STRING (SIZE (1..118)),
        relayedComponent [1] IMPLICIT OCTET STRING}

ComponentCorrelationID ::= INTEGER

ComponentType ::= ENUMERATED {
        any     .  (0),
        invoke   (1),
        rResult   (2),
        rError   (3),
        rReject   (4)}

ConnectedNumberTreatmentInd ::= ENUMERATED {
        noINImpact  (0),
        presentationRestricted  (1),
        presentCalledINNumber  (2)}

Constraints ::= SEQUENCE {
        maximumNumberOfDigits [1] IMPLICIT INTEGER (1..127),
        minimumNumberOfDigits [2] IMPLICIT INTEGER (1..127),
        typeOfRequestedInfo [3] IMPLICIT ENUMERATED {
                numericString  (0),
                characterString  (1),
                iA5String      (2)} DEFAULT numericString ,
        numberOfAllowedRetries [4] IMPLICIT INTEGER (0..127) DEFAULT 0}

ControlConditionByCallParty ::= SEQUENCE {
        endOfMessageSendingDigit [0] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
        replayDigit [1] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL}

ControlType ::= ENUMERATED {
        sCPOverloaded  (0),
        manuallyInitiated  (1),
        destinationOverload  (2)}

CorrelationID ::= OCTET STRING (SIZE (??..??))

CounterAndValue ::= SEQUENCE {
        counterID  [0] IMPLICIT INTEGER (0..??),
        counterValue [1] IMPLICIT INTEGER (0..2147483647)}

CounterID  ::= INTEGER (0..??)

CountersValue ::= SEQUENCE SIZE (0..100) OF
        SEQUENCE {
```

```
                counterID  [0] IMPLICIT INTEGER (0..??),
                counterValue [1] IMPLICIT INTEGER (0..2147483647)}

Credit  ::= CHOICE {
        currency        SEQUENCE {
                currency        PrintableString  (SIZE (3)),
                amount           INTEGER (0..??)},
        units           INTEGER (0..65536)}

CreditUnit ::= INTEGER (0..65536)

CriticalityType ::= ENUMERATED {
        ignore          (0),
        abort           (1)}

CSAID       ::= INTEGER (1..??)

CurrencyID ::= PrintableString  (SIZE (3))

CurrencyValue ::= SEQUENCE {
        currency        PrintableString  (SIZE (3)),
        amount           INTEGER (0..??)}

CutAndPaste ::= INTEGER (0..22)

DateAndTime ::= OCTET STRING (SIZE (6))

DestinationRoutingAddress ::= SEQUENCE SIZE (1..3) OF
        OCTET STRING (SIZE (??..??))

Digits   ::= OCTET STRING (SIZE (??..??))

DisplayInformation ::= IA5String        (SIZE (??..??))

DpSpecificCommonParameters ::= 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 {} OPTIONAL,
chargeNumber [13] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
servingAreaID [14] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
serviceInteractionIndicators [15] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
iNServiceCompatibilityIndication [16] IMPLICIT SEQUENCE SIZE (1..??) OF
        CHOICE {
        .       agreements [0] IMPLICIT OBJECT IDENTIFIER,
        .       networkSpecific [1] IMPLICIT INTEGER (0..2147483647)} OPTIONAL,
serviceInteractionIndicatorsTwo [17] IMPLICIT SEQUENCE {
        forwardServiceInteractionInd [0] IMPLICIT SEQUENCE {
        .       conferenceTreatmentIndicator [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        .       callDiversionTreatmentIndicator [2] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        .       callOfferingTreatmentIndicator [3] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL}
                OPTIONAL,
        backwardServiceInteractionInd [1] IMPLICIT SEQUENCE {
        .       conferenceTreatmentIndicator [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        .       callCompletionTreatmentIndicator [2] IMPLICIT OCTET STRING (SIZE (1))
                OPTIONAL} OPTIONAL,
        bothwayThroughConnectionInd [2] IMPLICIT ENUMERATED {
        .       bothwayPathRequired  (0),
        .       bothwayPathNotRequired  (1)} OPTIONAL,
        suspendTimer [3] IMPLICIT INTEGER (0..120) OPTIONAL,
        connectedNumberTreatmentInd [4] IMPLICIT ENUMERATED {
        .       noINImpact  (0),
        .       presentationRestricted  (1),
        .       presentCalledINNumber  (2)} OPTIONAL,
        suppressCallDiversionNotification [5] IMPLICIT BOOLEAN OPTIONAL,
        suppressCallTransferNotification [6] IMPLICIT BOOLEAN OPTIONAL,
        allowCdINNoPresentationInd [7] IMPLICIT BOOLEAN OPTIONAL,
        userDialogueDurationInd [8] IMPLICIT BOOLEAN DEFAULT TRUE,
        ... } OPTIONAL,
uSIServiceIndicator [18] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
uSIInformation [19] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
forwardGVNS [20] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
```

```
        createdCallSegmentAssociation [21] IMPLICIT INTEGER (1..??) OPTIONAL,
        ... }

DpSpecificCriteria ::= CHOICE {
        numberOfDigits [0] IMPLICIT INTEGER (1..255),
        applicationTimer [1] IMPLICIT INTEGER (0..2047),
        midCallControlInfo [2] IMPLICIT SEQUENCE SIZE (??..??) OF
                SEQUENCE {
                .       midCallInfoType [0] IMPLICIT SEQUENCE {
                .               iNServiceControlCodeLow [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                .               iNServiceControlCodeHigh [1] IMPLICIT OCTET STRING (SIZE (??..??))
                                OPTIONAL},
                .       midCallReportType [1] IMPLICIT ENUMERATED {
                .               inMonitoringState  (0),
                .               inAnyState  (1)} DEFAULT inMonitoringState }}

Duration        ::= INTEGER (-2..86400)

ElementaryMessageID ::= INTEGER (0..2147483647)

Entry           ::= CHOICE {
        agreements [0] IMPLICIT OBJECT IDENTIFIER,
        networkSpecific [1] IMPLICIT INTEGER (0..2147483647)}

ErrorTreatment ::= ENUMERATED {
        reportErrorToScf  (0),
        help              (1),
        repeatPrompt  (2)}

EventSpecificInformationBCSM ::= CHOICE {
        collectedInfoSpecificInfo [0] IMPLICIT SEQUENCE {
                calledPartynumber [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                ... },
        analysedInfoSpecificInfo [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 {
                backwardGVNS [0] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                ... },
        oMidCallSpecificInfo [6] IMPLICIT SEQUENCE {
                connectTime [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                oMidCallInfo [1] IMPLICIT SEQUENCE {
                .       iNServiceControlCode [0] IMPLICIT OCTET STRING (SIZE (??..??))} 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,
                tMidCallInfo [1] IMPLICIT SEQUENCE {
                .       iNServiceControlCode [0] IMPLICIT OCTET STRING (SIZE (??..??))} OPTIONAL,
                ... },
        tDisconnectSpecificInfo [12] IMPLICIT SEQUENCE {
                releaseCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
                connectTime [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
                ... },
        oTermSeizedSpecificInfo [13] IMPLICIT SEQUENCE {
                ... },
        oSuspended [14] IMPLICIT SEQUENCE {
                ... },
        tSuspended [15] IMPLICIT SEQUENCE {
                ... },
        origAttemptAuthorized [16] IMPLICIT SEQUENCE {
                ... },
        oReAnswer  [17] IMPLICIT SEQUENCE {
                ... },
        tReAnswer  [18] IMPLICIT SEQUENCE {
                ... },
        facilitySelectedAndAvailable [19] IMPLICIT SEQUENCE {
                ... },
        callAccepted [20] IMPLICIT SEQUENCE {
                ... },
        oAbandon   [21] IMPLICIT SEQUENCE {
                abandonCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
        ... },
        tAbandon   [22] IMPLICIT SEQUENCE {
                abandonCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
        ... }}EventSpecificInformationCharging ::= OCTET STRING (SIZE (??..??))

EventTypeBCSM ::= ENUMERATED {
        origAttemptAuthorized  (1),
        collectedInfo  (2),
        analysedInformation  (3),
        routeSelectFailure  (4),
        oCalledPartyBusy  (5),
        oNoAnswer   (6),
        oAnswer        (7),
        oMidCall       (8),
        oDisconnect  (9),
        oAbandon     (10),
        termAttemptAuthorized  (12),
        tBusy          (13),
        tNoAnswer   (14),
        tAnswer        (15),
        tMidCall       (16),
        tDisconnect  (17),
        tAbandon       (18),
        oTermSeized  (19),
        oSuspended  (20),
        tSuspended  (21),
        origAttempt  (22),
        termAttempt  (23),
        oReAnswer   (24),
        tReAnswer   (25),
        facilitySelectedAndAvailable  (26),
        callAccepted  (27)}
```

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

EventTypeCharging ::= OCTET STRING (SIZE (??..??))

ExtensionField ::= SEQUENCE {}

FacilityGroup ::= CHOICE {
        trunkGroupID [0] IMPLICIT INTEGER,
        privateFacilityID [1] IMPLICIT INTEGER,
        huntGroup  [2] IMPLICIT OCTET STRING,
        routeIndex [3] IMPLICIT OCTET STRING}

FacilityGroupMember ::= INTEGER

FailureCause ::= OCTET STRING

FCIBillingChargingCharacteristics ::= OCTET STRING (SIZE (??..??))

FeatureCode ::= OCTET STRING (SIZE (??..??))

FeatureRequestIndicator ::= ENUMERATED {
        hold              (0),
        retrieve          (1),
        featureActivation  (2),
        spare1            (3),
        sparen            (127)}

FilteredCallTreatment ::= 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 ::= CHOICE {
```

```
        interval        [0] IMPLICIT INTEGER (1..32000),
        numberOfCalls [1] IMPLICIT INTEGER (0..2147483647)}

FilteringCriteria ::= 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}}

FilteringTimeOut ::= CHOICE {
        duration        [0] IMPLICIT INTEGER (-2..86400),
        stopTime        [1] IMPLICIT OCTET STRING (SIZE (6))}

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

ForwardGVNS ::= OCTET STRING (SIZE (??..??))

ForwardingCondition ::= ENUMERATED {
        busy            (0),
        noanswer        (1),
        any             .       (2)}
ForwardServiceInteractionInd ::= SEQUENCE {
        conferenceTreatmentIndicator [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        callDiversionTreatmentIndicator [2] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
        callOfferingTreatmentIndicator [3] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL}

GapCriteria ::= 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),
                .       oTermSeized  (19),
                .       oSuspended  (20),
                .       tSuspended  (21),
                .       origAttempt  (22),
                .       termAttempt  (23),
                .       oReAnswer    (24),
                .       tReAnswer    (25),
                .       facilitySelectedAndAvailable  (26),
                .       callAccepted  (27)} OPTIONAL},
```

```
        gapAllInTraffic [3] IMPLICIT NULL,
        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}}

GapOnService ::= 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),
                oTermSeized  (19),
                oSuspended  (20),
                tSuspended  (21),
                origAttempt  (22),
                termAttempt  (23),
                oReAnswer  (24),
                tReAnswer  (25),
                facilitySelectedAndAvailable  (26),
                callAccepted  (27)} OPTIONAL}

GapIndicators ::= SEQUENCE {
        duration  [0] IMPLICIT INTEGER (-2..86400),
        gapInterval [1] IMPLICIT INTEGER (-1..60000)}

GapTreatment ::= 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..??))}}

GenericName ::= OCTET STRING (SIZE (??..??))

GenericNumber ::= OCTET STRING (SIZE (??..??))

GenericNumbers ::= SET SIZE (1..??) OF
        OCTET STRING (SIZE (??..??))

HighLayerCompatibilities ::= BIT STRING {
        telephony       (0),
        facsimileGroup2-3   (1),
        facsimileGroup4classeI  (2),
        teletexMixedMode    (3),
        teletexProcessableMode  (4),
        teletexBasicMode    (5),
        syntaxBasedVideotex     (6),
        internationalVideotex   (7),
        telexService    (8),
        messageHandlingSystem   (9),
        osiApplication  (10),
        audioVisual     (11)}

HighLayerCompatibility ::= OCTET STRING (SIZE (2))
```

```
HoldCause  ::= OCTET STRING

InbandInfo ::= 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}

InformationToRecord ::= SEQUENCE {
        messageID  [0] IMPLICIT INTEGER (0..2147483647) OPTIONAL,
        messageDeletionTimeOut [1] IMPLICIT INTEGER (1..3600) OPTIONAL,
        timeToRecord [3] IMPLICIT INTEGER (0..??) OPTIONAL,
        controlDigits [4] IMPLICIT SEQUENCE {
                endOfRecordingDigit [0] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
                cancelDigit [1] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
                replayDigit [2] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
                restartRecordingDigit [3] IMPLICIT OCTET STRING (SIZE (1..2)) OPTIONAL,
                restartAllowed [4] IMPLICIT BOOLEAN DEFAULT FALSE,
                replayAllowed [5] IMPLICIT BOOLEAN DEFAULT FALSE}}

InformationToSend ::= 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 (??..??))}
```

```
InfoToSend ::= CHOICE {
        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))}}},
        toneId  [1] CHOICE {
                local           [0] IMPLICIT INTEGER (0..2147483647),
                global  [1] IMPLICIT OBJECT IDENTIFIER},
        displayInformation [2] IMPLICIT IA5String  (SIZE (??..??))}

InfoType      ::= ENUMERATED {
        numericString  (0),
        characterString  (1),
        iA5String      (2)}

INServiceCompatibilityIndication ::= SEQUENCE SIZE (1..??) OF
        CHOICE {
                agreements [0] IMPLICIT OBJECT IDENTIFIER,
                networkSpecific [1] IMPLICIT INTEGER (0..2147483647)}

INServiceCompatibilityResponse ::= CHOICE {
        agreements [0] IMPLICIT OBJECT IDENTIFIER,
        networkSpecific [1] IMPLICIT INTEGER (0..2147483647)}

Integer4      ::= INTEGER (0..2147483647)

InteractionStrategy ::= ENUMERATED {
        stopOnError  (1),
        bestEffort  (2)}

Interval      ::= INTEGER (-1..60000)

InvokableService ::= ENUMERATED {
        callingLineIdentificationRestriction  (1),
        connectedLineIdentificationRestriction  (2),
        callWaiting  (3),
        callHold       (4),
        reverseCharging  (5),
        explicitCallTransfer  (6),
        callCompletionOnBusySubscriber  (7)}

InvokeID      ::= INTEGER (-128..127)

IPAvailable ::= OCTET STRING (SIZE (??..??))

IPRoutingAddress ::= OCTET STRING (SIZE (??..??))

IPSSPCapabilities ::= OCTET STRING (SIZE (??..??))
```

```
ISDNAccessRelatedInformation ::= OCTET STRING

Language     ::= PrintableString  (SIZE (3))

LegID        ::= CHOICE {
        sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
        receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))}

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

leg1             OCTET STRING (SIZE (1)) ::= '01'H

leg2             OCTET STRING (SIZE (1)) ::= '02'H

LocationNumber ::= OCTET STRING (SIZE (??..??))

MailBoxID  ::= OCTET STRING (SIZE (??..??))

MaximumNumberOfCounters ::= INTEGER (1..100)

Media        ::= ENUMERATED {
        voiceMail     (0),
        faxGroup3     (1),
        faxGroup4     (2)}

Message      ::= ENUMERATED {
        rELease         (77),
        rELeaseCOMPlete  (90),
        fACility        (98)}

MessageID  ::= 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))}}}

MidCallControlInfo ::= SEQUENCE SIZE (??..??) OF
        SEQUENCE {
                midCallInfoType [0] IMPLICIT SEQUENCE {
                .       iNServiceControlCodeLow [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                .       iNServiceControlCodeHigh [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL},
                midCallReportType [1] IMPLICIT ENUMERATED {
                .       inMonitoringState  (0),
                .       inAnyState  (1)} DEFAULT inMonitoringState }

MidCallInfo ::= SEQUENCE {
        iNServiceControlCode [0] IMPLICIT OCTET STRING (SIZE (??..??))}

MidCallInfoType ::= SEQUENCE {
        iNServiceControlCodeLow [0] IMPLICIT OCTET STRING (SIZE (??..??)),
```

```
        iNServiceControlCodeHigh [1] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL}

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

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

Notification ::= ENUMERATED {
        userAbandon  (0),
        callFailure  (1),
        noReply         (2),
        callRelease  (3),
        ssInvocation  (4),
        creditLimitReached  (5),
        callDuration  (6),
        calledNumber  (7),
        answeredCall  (8)}

NotificationInformation ::= CHOICE {
        userAbandonSpecificInfo [0] IMPLICIT SEQUENCE {
                ... },
        callFailureSpecificInfo [1] IMPLICIT SEQUENCE {
                failureCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
                ... },
        noReplySpecificInfo [2] IMPLICIT SEQUENCE {
                ... },
        callReleaseSpecificInfo [3] IMPLICIT SEQUENCE {
                releaseCause [0] IMPLICIT OCTET STRING (SIZE (2..??)) OPTIONAL,
                timeStamp  [1] IMPLICIT OCTET STRING (SIZE (6)) OPTIONAL,
                ... },
        ssInvocationSpecificInfo [4] IMPLICIT SEQUENCE {
                invokedService [0] IMPLICIT ENUMERATED {
                .       callingLineIdentificationRestriction  (1),
                .       connectedLineIdentificationRestriction  (2),
                .       callWaiting  (3),
                .       callHold        (4),
                .       reverseCharging  (5),
                .       explicitCallTransfer  (6),
                .       callCompletionOnBusySubscriber  (7)},
                ... },
        creditLimitReachedSpecificInfo [5] IMPLICIT SEQUENCE {
                timeStamp  [0] IMPLICIT OCTET STRING (SIZE (6)) OPTIONAL,
                ... },
        callDurationSpecificInfo [6] IMPLICIT SEQUENCE {
                timeStamp  [0] IMPLICIT OCTET STRING (SIZE (6)) OPTIONAL,
                ... },
        calledNumberSpecificInfo [7] IMPLICIT SEQUENCE {
                calledNumber [0] IMPLICIT OCTET STRING (SIZE (??..??)) OPTIONAL,
                ... },
        answeredCallSpecificInfo [8] IMPLICIT SEQUENCE {
                timeStamp  [0] IMPLICIT OCTET STRING (SIZE (6)) OPTIONAL,
                ... }}
```

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

NumberMatch ::= CHOICE {
        initialMatch [0] IMPLICIT OCTET STRING (SIZE (??..??)),
        totalMatch [1] IMPLICIT OCTET STRING (SIZE (??..??))}

NumberOfDigits ::= INTEGER (1..255)

OperationCode ::= CHOICE {
        globalCode OBJECT IDENTIFIER,
        local           INTEGER}

OriginalCalledPartyID ::= OCTET STRING (SIZE (??..??))

ProfileIdentifier ::= CHOICE {
        access  [0] IMPLICIT OCTET STRING (SIZE (??..??)),
        group           [1] CHOICE {
                trunkGroupID [0] IMPLICIT INTEGER,
                privateFacilityID [1] IMPLICIT INTEGER,
                huntGroup  [2] IMPLICIT OCTET STRING,
                routeIndex [3] IMPLICIT OCTET STRING}}

Reason          ::= OCTET STRING (SIZE (??..??))

ReceivedInformation ::= SEQUENCE SIZE (??..??) OF
        IA5String

ReceivedStatus ::= ENUMERATED {
        messageComplete  (0),
        messageInterrupted  (1),
        messageTimeOut  (2)}

RecordedMessageID ::= INTEGER (0..2147483647)

RedirectingPartyID ::= OCTET STRING (SIZE (??..??))

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

RegistratorIdentifier ::= OCTET STRING

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

RequestedInformationList ::= 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..??))}}
```

```
RequestedInformationTypeList ::= SEQUENCE SIZE (1..5) OF
        ENUMERATED {
                callAttemptElapsedTime  (0),
                callStopTime  (1),
                callConnectedElapsedTime  (2),
                calledAddress  (3),
                releaseCause  (30)}

RequestedInformation ::= 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..??))}}

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

RequestedInformationValue ::= 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..??))}

RequestedNotifications ::= SET  OF
        CHOICE {
                userAbandon [0] IMPLICIT NULL,
                callFailure [1] IMPLICIT OCTET STRING (SIZE (1)),
                noReply         [2] IMPLICIT INTEGER,
                callRelease [3] IMPLICIT NULL,
                ss-invocation [4] IMPLICIT ENUMERATED {
                .       callingLineIdentificationRestriction  (1),
                .       connectedLineIdentificationRestriction  (2),
                .       callWaiting  (3),
                .       callHold        (4),
                .       reverseCharging  (5),
                .       explicitCallTransfer  (6),
                .       callCompletionOnBusySubscriber  (7)},
                creditLimitReached [5] IMPLICIT INTEGER,
                callDuration [6] IMPLICIT INTEGER,
                calledNumber [7] CHOICE {
                .       initialMatch [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                .       totalMatch [1] IMPLICIT OCTET STRING (SIZE (??..??))},
                answeredCall [8] IMPLICIT NULL}

RequestedType ::= INTEGER (0..127)

RequestedUTSI ::= SEQUENCE {
        uSIServiceIndicator [0] IMPLICIT OCTET STRING (SIZE (??..??)),
```

```
        uSImonitorMode [1] IMPLICIT ENUMERATED {
                monitoringActive  (0),
                monitoringInactive  (1)},
        legID           [2] CHOICE {
                sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
                receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} DEFAULT sendingSideID  : '01'H}

RequestedUTSIList ::= SEQUENCE SIZE (??..??) OF
        SEQUENCE {
                uSIServiceIndicator [0] IMPLICIT OCTET STRING (SIZE (??..??)),
                uSImonitorMode [1] IMPLICIT ENUMERATED {
                .       monitoringActive  (0),
                .       monitoringInactive  (1)},
                legID           [2] CHOICE {
                .       sendingSideID [0] IMPLICIT OCTET STRING (SIZE (1)),
                .       receivingSideID [1] IMPLICIT OCTET STRING (SIZE (1))} DEFAULT sendingSideID :
                        '01'H}

ResourceID ::= 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}

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

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

RouteList  ::= SEQUENCE SIZE (1..3) OF
        OCTET STRING (SIZE (??..??))

RoutingAddress ::= CHOICE {
        routingProhibited [0] IMPLICIT NULL,
        destinationRoutingAddress [1] IMPLICIT SEQUENCE SIZE (1..3) OF
                OCTET STRING (SIZE (??..??))}

ScfAddress ::= OCTET STRING (SIZE (??..??))

ScfID           ::= OCTET STRING (SIZE (??..??))

SCIBillingChargingCharacteristics ::= OCTET STRING (SIZE (??..??))

ServiceAddressInformation ::= 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}

ServiceInteractionIndicators ::= OCTET STRING (SIZE (??..??))

ServiceInteractionIndicatorsTwo ::= SEQUENCE {
        forwardServiceInteractionInd [0] IMPLICIT SEQUENCE {
                conferenceTreatmentIndicator [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
                callDiversionTreatmentIndicator [2] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
                callOfferingTreatmentIndicator [3] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL}
                OPTIONAL,
        backwardServiceInteractionInd [1] IMPLICIT SEQUENCE {
                conferenceTreatmentIndicator [1] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL,
                callCompletionTreatmentIndicator [2] IMPLICIT OCTET STRING (SIZE (1)) OPTIONAL}
                OPTIONAL,
        bothwayThroughConnectionInd [2] IMPLICIT ENUMERATED {
                bothwayPathRequired  (0),
                bothwayPathNotRequired  (1)} OPTIONAL,
        suspendTimer [3] IMPLICIT INTEGER (0..120) OPTIONAL,
        connectedNumberTreatmentInd [4] IMPLICIT ENUMERATED {
                noINImpact  (0),
                presentationRestricted  (1),
                presentCalledINNumber  (2)} OPTIONAL,
        suppressCallDiversionNotification [5] IMPLICIT BOOLEAN OPTIONAL,
        suppressCallTransferNotification [6] IMPLICIT BOOLEAN OPTIONAL,
        allowCdINNoPresentationInd [7] IMPLICIT BOOLEAN OPTIONAL,
        userDialogueDurationInd [8] IMPLICIT BOOLEAN DEFAULT TRUE,
        ... }
ServiceKey ::= INTEGER (0..2147483647)

ServiceProfileIdentifier ::= OCTET STRING

ServingAreaID ::= OCTET STRING (SIZE (??..??))

SFBillingChargingCharacteristics ::= OCTET STRING (SIZE (??..??))

SubscriberId ::= OCTET STRING (SIZE (??..??))

SupplementaryServices ::= BIT STRING {
        callingLineIdentificationPresentation  (1),
        callingLineIdentificationRestriction  (2),
        connectedLineIdentificationPresentation  (3),
        connectedLineIdentificationRestriction  (4),
        callForwardingOnNoReply  (5),
```

```
        callForwardingUnconditional  (6),
        callForwardingOnBusy  (7),
        callForwardingOnNotReachable  (8),
        callWaiting  (9),
        callHold        (10),
        reverseCharging  (11),
        explicitCallTransfer  (12),
        callCompletionOnBusySubscriber  (13),
        adviceOfChargeOnStart  (14),
        adviceOfChargeAtEnd  (15),
        adviceOfChargeDuringCall  (16),
        timeDependentRouting  (17),
        callingPartingDependentRouting  (18),
        outgoingCallBarring  (19),
        incomingCallBarring  (20)}

SuspendTimer ::= INTEGER (0..120)

TargetLineIdentifier ::= CHOICE {
        individual [0] IMPLICIT OCTET STRING (SIZE (??..??)),
        group          [1] CHOICE {
                trunkGroupID [0] IMPLICIT INTEGER,
                privateFacilityID [1] IMPLICIT INTEGER,
                huntGroup  [2] IMPLICIT OCTET STRING,
                routeIndex [3] IMPLICIT OCTET STRING}}

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

TimerID       ::= ENUMERATED {
        tssf           (0)}

TimerValue ::= INTEGER (0..2147483647)

Tone          ::= SEQUENCE {
        toneID         [0] IMPLICIT INTEGER (0..2147483647),
        duration       [1] IMPLICIT INTEGER (0..2147483647) OPTIONAL}

ToneId        ::= CHOICE {
        local          [0] IMPLICIT INTEGER (0..2147483647),
        global   [1] IMPLICIT OBJECT IDENTIFIER}

TraceInformation ::= SEQUENCE  OF
        SET {
                scf            [0] IMPLICIT OCTET STRING (SIZE (??..??))}

TraceItem  ::= SET {
        scf            [0] IMPLICIT OCTET STRING (SIZE (??..??))}

TravellingClassMark ::= OCTET STRING (SIZE (??..??))

TriggerDataIdentifier ::= SEQUENCE {
        triggerID  [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),
            oTermSeized  (19),
            oSuspended  (20),
            tSuspended  (21),
            origAttempt  (22),
            termAttempt  (23),
            oReAnswer   (24),
            tReAnswer   (25),
            facilitySelectedAndAvailable  (26),
            callAccepted  (27)},
        profileIdentifier [1] CHOICE {
            access   [0] IMPLICIT OCTET STRING (SIZE (??..??)),
            group           [1] CHOICE {
            .       trunkGroupID [0] IMPLICIT INTEGER,
            .       privateFacilityID [1] IMPLICIT INTEGER,
            .       huntGroup  [2] IMPLICIT OCTET STRING,
            .       routeIndex [3] IMPLICIT OCTET STRING}},
        extensions [2] IMPLICIT SEQUENCE SIZE (1..??) OF
            SEQUENCE {} OPTIONAL}

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)}

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

```
UserCredit ::= CHOICE {
        currency        SEQUENCE {
                currency        PrintableString  (SIZE (3)),
                amount          INTEGER (0..??)},
        units           INTEGER (0..65536)}

UserInfo        ::= SEQUENCE  OF
        SEQUENCE {
                infoToSend [0] CHOICE {
                .       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))}}},
                .       toneId  [1] CHOICE {
                .               local           [0] IMPLICIT INTEGER (0..2147483647),
                .               global  [1] IMPLICIT OBJECT IDENTIFIER},
                .       displayInformation [2] IMPLICIT IA5String  (SIZE (??..??))},
                constraints [1] IMPLICIT SEQUENCE {
                .       maximumNumberOfDigits [1] IMPLICIT INTEGER (1..127),
                .       minimumNumberOfDigits [2] IMPLICIT INTEGER (1..127),
                .       typeOfRequestedInfo [3] IMPLICIT ENUMERATED {
                .               numericString  (0),
                .               characterString  (1),
                .               iA5String      (2)} DEFAULT numericString ,
                .       numberOfAllowedRetries [4] IMPLICIT INTEGER (0..127) DEFAULT 0},
                errorInfo  [2] CHOICE {
                .       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))}}},
                .       toneId  [1] CHOICE {
                .               local           [0] IMPLICIT INTEGER (0..2147483647),
                .               global  [1] IMPLICIT OBJECT IDENTIFIER},
                .       displayInformation [2] IMPLICIT IA5String  (SIZE (??..??))} OPTIONAL}

UserInformation ::= SEQUENCE {
        infoToSend [0] CHOICE {
```

```
            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))}}},
            toneId  [1] CHOICE {
            .       local           [0] IMPLICIT INTEGER (0..2147483647),
            .       global  [1] IMPLICIT OBJECT IDENTIFIER},
            displayInformation [2] IMPLICIT IA5String  (SIZE (??..??))},
        constraints [1] IMPLICIT SEQUENCE {
            maximumNumberOfDigits [1] IMPLICIT INTEGER (1..127),
            minimumNumberOfDigits [2] IMPLICIT INTEGER (1..127),
            typeOfRequestedInfo [3] IMPLICIT ENUMERATED {
            .       numericString  (0),
            .       characterString  (1),
            .       iA5String      (2)} DEFAULT numericString ,
            numberOfAllowedRetries [4] IMPLICIT INTEGER (0..127) DEFAULT 0},
        errorInfo  [2] CHOICE {
            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))}}},
            toneId  [1] CHOICE {
            .       local           [0] IMPLICIT INTEGER (0..2147483647),
            .       global  [1] IMPLICIT OBJECT IDENTIFIER},
            displayInformation [2] IMPLICIT IA5String  (SIZE (??..??))} OPTIONAL}

UserInteractionModes ::= BIT STRING {
        voiceMessage  (0),
        tone          (1),
        display       (2)}

USIInformation ::= OCTET STRING (SIZE (??..??))

USIMonitorMode ::= ENUMERATED {
        monitoringActive  (0),
        monitoringInactive  (1)}
```

USIServiceIndicator ::= OCTET STRING (SIZE (??..??))

VariablePart ::= 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))}

highLayerCompatibilityLength INTEGER ::= 2

minAChBillingChargingLength INTEGER ::= ??

maxAChBillingChargingLength INTEGER ::= ??

minAttributesLength INTEGER ::= ??

maxAttributesLength INTEGER ::= ??

minBackwardGVNSLength INTEGER ::= ??

maxBackwardGVNSLength INTEGER ::= ??

maxBearerCapabilityLength INTEGER ::= ??

minCalledPartyNumberLength INTEGER ::= ??

maxCalledPartyNumberLength INTEGER ::= ??

minCallingPartyNumberLength INTEGER ::= ??

maxCallingPartyNumberLength INTEGER ::= ??

minCallResultLength INTEGER ::= ??

maxCallResultLength INTEGER ::= ??

minCauseLength INTEGER ::= 2

maxCauseLength INTEGER ::= ??

minDigitsLength INTEGER ::= ??

maxDigitsLength INTEGER ::= ??

minDisplayInformationLength INTEGER ::= ??

maxDisplayInformationLength INTEGER ::= ??

minEventSpecificInformationChargingLength INTEGER ::= ??

maxEventSpecificInformationChargingLength INTEGER ::= ??

minEventTypeChargingLength INTEGER ::= ??

maxEventTypeChargingLength INTEGER ::= ??

minFCIBillingChargingLength INTEGER ::= ??

maxFCIBillingChargingLength INTEGER ::= ??

minForwardGVNSLength INTEGER ::= ??

maxForwardGVNSLength INTEGER ::= ??

minGenericNameLength INTEGER ::= ??

maxGenericNameLength INTEGER ::= ??

minGenericNumberLength INTEGER ::= ??

maxGenericNumberLength INTEGER ::= ??

maxInitialTimeInterval INTEGER ::= ??

maxINServiceCompatibilityIndLength INTEGER ::= ??

minIPAvailableLength INTEGER ::= ??

maxIPAvailableLength INTEGER ::= ??

minIPSSPCapabilitiesLength INTEGER ::= ??

maxIPSSPCapabilitiesLength INTEGER ::= ??

minLocationNumberLength INTEGER ::= ??

maxLocationNumberLength INTEGER ::= ??

minMailBoxIDLength INTEGER ::= ??

maxMailBoxIDLength INTEGER ::= ??

minMessageContentLength INTEGER ::= ??

maxMessageContentLength INTEGER ::= ??

minMidCallControlInfoNum INTEGER ::= ??

maxMidCallControlInfoNum INTEGER ::= ??

minOriginalCalledPartyIDLength INTEGER ::= ??

maxOriginalCalledPartyIDLength INTEGER ::= ??

minReasonLength INTEGER ::= ??

maxReasonLength INTEGER ::= ??

maxRecordedMessageUnits INTEGER ::= ??

maxRecordingTime INTEGER ::= ??

minRedirectingPartyIDLength INTEGER ::= ??

maxRedirectingPartyIDLength INTEGER ::= ??

minRouteListLength INTEGER ::= ??

maxRouteListLength INTEGER ::= ??

minScfIDLength INTEGER ::= ??

maxScfIDLength INTEGER ::= ??

minSCIBillingChargingLength INTEGER ::= ??

maxSCIBillingChargingLength INTEGER ::= ??

minServiceInteractionIndicatorsLength INTEGER ::= ??

maxServiceInteractionIndicatorsLength INTEGER ::= ??

minSFBillingChargingLength INTEGER ::= ??

maxSFBillingChargingLength INTEGER ::= ??

minUSIInformationLength INTEGER ::= ??

maxUSIInformationLength INTEGER ::= ??

minUSIServiceIndicatorLength INTEGER ::= ??

maxUSIServiceIndicatorLength INTEGER ::= ??

numOfBCSMEvents INTEGER ::= ??

numOfBCUSMEvents INTEGER ::= ??

numOfChargingEvents INTEGER ::= ??

numOfCounters INTEGER ::= 100

numOfCSAs INTEGER ::= ??

numOfCSs INTEGER ::= ??

numOfExtensions INTEGER ::= ??

numOfInfoItems INTEGER ::= 5

numOfGenericNumbers INTEGER ::= ??

numOfLegs INTEGER ::= ??

numOfMessageIDs INTEGER ::= ??

numOfRecordedMessageIDs INTEGER ::= ??

maxCreditUnit INTEGER ::= 65536

maxInitialUnitIncrement INTEGER ::= ??

maxScalingFactor INTEGER ::= ??

maxTimePerInterval INTEGER ::= ??

maxUnitsPerDataInterval INTEGER ::= ??

maxUnitsPerInterval INTEGER ::= ??

maxSegmentsPerDataInterval INTEGER ::= ??

ub-maxUserCredit INTEGER ::= ??

maxAmount INTEGER ::= ??

numOfInServiceCompatibilityIndLength INTEGER ::= ??

minReceivedInformationLength INTEGER ::= ??

maxReceivedInformationLength INTEGER ::= ??

minRequestedUTSINum INTEGER ::= ??

maxRequestedUTSINum INTEGER ::= ??

minScfAddressLength INTEGER ::= ??

maxScfAddressLength INTEGER ::= ??

ub-nbCall INTEGER ::= ??

END

APPENDIX II

Data modelling

II.1 Introduction

II.1.1 Purpose and scope

This Appendix describes an object modelling which may be used for Inter-network Service Profile Transfer (ISPT).

II.1.2 Assumptions

II.1.2.1 Mobility service and subscriber identifiers

II.1.2.1.1 Mobility subscriber unique identifier

A mobility service subscriber must have a unique numeric identifier to distinguish it from other subscribers of the same mobility service provider. Subscriber identifiers are locally significant. That is, service providers do not need to know the unique identifiers of subscribers to other service providers.

A maximum length should be agreed upon; however, within that boundary the length of a subscriber identifier is a local matter.

II.1.2.1.2 Mobility service provider prefix

Each mobility service provider must have a unique, numeric identifier (prefix). Service provider prefixes are globally significant. That is, cooperating mobility service providers must know each others numeric identifiers.

A maximum length should be agreed upon.

Using a service provider's identifier as a prefix to each local subscriber's unique identifier would allow all mobility subscribers to be uniquely and unambiguously identified.

II.1.2.1.3 Mobility service provider sub-prefix

If a mobility service provider needs to be able to logically group its subscribers, for example if multiple DSAs are being used, a service provider sub-prefix may be used. Service provider sub-prefixes are locally significant. That is, cooperating mobility service providers need not know each others sub-prefix identifiers.

A maximum length should be agreed upon; however, within that boundary the length of a service provider's sub-prefix is a local matter.

Use of additional sub-prefixes is not defined in this Appendix. It is not envisioned that this will be necessary, however, it would not be difficult to define and implement.

II.1.2.1.4 Encoding

The means whereby a roaming user's unique identifier, service provider prefix and sub-prefix is disclosed to a visited network is beyond the scope of this Appendix. For the purposes of this Appendix, some type of encoding, such as object identifier, length and value is assumed so that a user's unique identifier, service provider prefix and sub-prefix can be individually recognized.

II.1.2.2 Disclosure of profile entries

It is not desirable to require that the entire list of subscriber entries for a service provider be made available to the other service providers. Only the profiles of roaming users should be disclosed.

II.2 Directory Information Tree (DIT) schema

II.2.1 X.500 DIT

The X.500 DIT shown in Figure II.1 illustrates the DIT schema described in this Appendix.

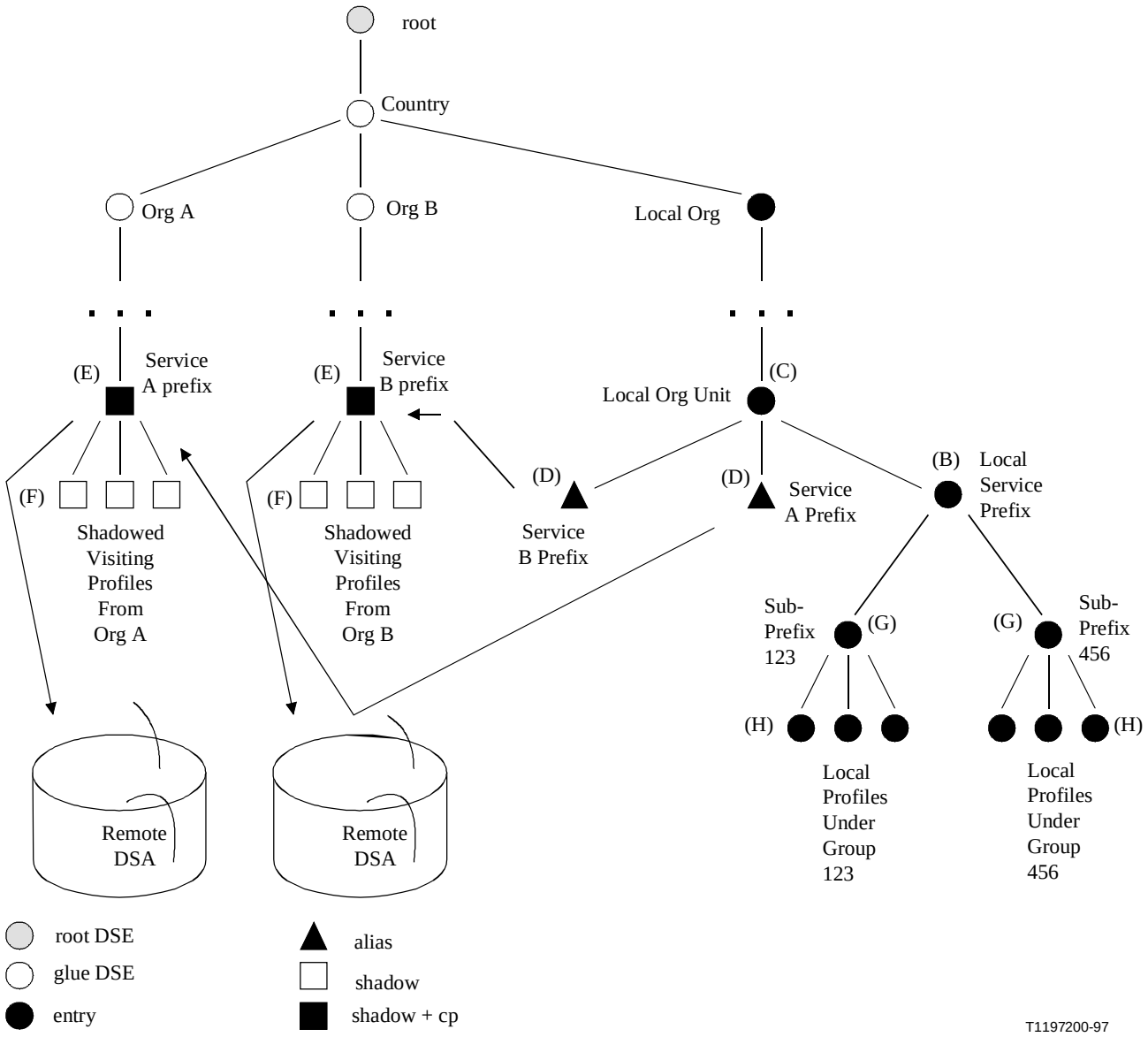


Figure II.1/Q.1228 – Example mobility service provider's local DIT

II.2.1.1 Location of local profiles

A service provider's prefix will be used to name the node [Refer to the node labelled (B) in Figure II.1] in the service provider's local DIT under which all local subscriber profiles are stored. The service provider prefix entry will contain an object class whose value is inMobilityServicePrefix.

User profiles may be further grouped by service provider sub-prefixes. If a sub-prefix is used, it will be realized as an additional relative distinguished name, and node [nodes labelled as (G)] in the DIT, between a user's unique identifier and its service provider's prefix. The service provider sub-prefix entries, if applicable, will contain an object class whose value is inMobilityServiceSubPrefix. The sub-prefix entries will be named using the service provider sub-prefix identifiers.

Local profile entries [nodes labelled as (H)] will contain an object class whose value is inMobilityUserProfile. Local subscriber entries will be named using their unique mobility service identifier.

II.2.1.2 Location of remote subscriber profiles

The location of a service provider's subscriber profiles must be made available to the other cooperating service providers. Entries [nodes labelled (E)] superior to profiles stored on remote DSAs must be shadowed on the local DSA. As long as this shadow DSE entry has incomplete subordinate knowledge, the visited network can generate a continuation reference from the supplierKnowledge attribute found in this entry. The continuation reference will point to the roaming user's home DSA where the actual profiles are located.

II.2.1.3 Location of locally visiting subscriber profiles

Locally visiting subscribers will have their profiles entries [node labelled (F)] shadowed under the node described above [nodes labelled (E)], which corresponds to their home service provider.

II.2.1.4 Reducing message flow

When a user roams into another network, the visited network must modify the roaming user's profile on the user's home network.This will cause the user's profile to be transferred to the visited network.

In order to modify the user's profile, the visited network must determine the roaming user's DN. To reduce message flow, it is possible that this distinguished name may be resolvable without using DAP search operations. The mapping from subscriber identifiers and directory numbers onto DNs will be facilitated by the use of fixed length numbering plans.

As per the assumptions given earlier, a roaming user's unique identifier and prefix (and sub-prefix if applicable) are revealed to the visited network. These can be used as component RDNs in the roaming user's DN. A preliminary DN can be generated by appending these RDNs to the DN of the node superior to the home network's prefix entry[node (B)].

Consider alias entries [nodes labelled (D)] for each cooperating service provider, located adjacent to the node under which a local service subscriber's profiles are stored [node (B)]. Also assume that these alias entries are named using service provider prefix numbers and are used to point to its corresponding node [nodes labelled (E)] located elsewhere in the local DIT.

When used in a DAP operation, the roaming user's preliminary DN would be dereferenced to point to the node in the DIT which is a shadow of the roaming user's home network prefix. As long as this shadow DSE entry has incomplete subordinate knowledge, the visited network can generate a continuation reference from the supplierKnowledge attribute found in this entry. The continuation reference will point to the roaming user's home DSA where the actual profiles are located. This operation will then be chained to the roaming user's home DSA.

II.2.2 Object classes

II.2.2.1 inMobilityUserProfile

The inMobilityUserProfile object class has been defined for storing profile information. The following ASN.1 definition may be used as a starting point for describing the inMobilityUserProfile object class:

```
inMobilityUserProfile     OBJECT-CLASS   ::=     {
        SUBCLASS OF       { top}
        MUST CONTAIN      { inMobilityID |
                                  inMobilityPIN |
                                        <other mandatory attributes}
        MAY CONTAIN       { <optional attributes>}
        ID                                Id-oc-inMobilityUserProfile}
```

The inMobilityID attribute is the distinguished attribute.

Entries of this type are only to be located under an entry of type inMobilityServiceProvider or inMobilitySubscriberGroup.

II.2.2.2 inMobilityServiceProvider

This object class has been created to define a node under which objects of type inMobilityUserProfile and inMobilitySubscriberGroup can be stored. The following ASN.1 definition describes the inMobilityServiceProvider object class:

```
inMobilityServiceProvider       OBJECT-CLASS   ::=     {
        SUBCLASS OF     { top}
        MUST CONTAIN    { inMobilityPrefix | <other mandatory attributes>}
        MAY CONTAIN     { <optional attributes>}
        ID                              Id-oc-inMobilityServiceProvider}
```

The inMobilityPrefix attribute is the distinguished attribute.

Entries of this type may contain a Non-Specific Subordinate Reference (NSSR) pointing to the profile entries, or sub-prefix entries if applicable, stored on a remote DSA.

Entries subordinate to this object class must be of type inMobilityUserProfile or inMobilitySubscriberGroup.

II.2.2.3 inMobilitySubscriberGroup

This object class has been created to define a node under which objects of type inMobilityUserProfile can be stored. The following ASN.1 definition describes the inMobilityServiceProvider object class:

```
inMobilitySubscriberGroup       OBJECT-CLASS   ::=     {
        SUBCLASS OF             { top}
        MUST CONTAIN            { inMobilitySubPrefix | <other mandatory attributes>}
        MAY CONTAIN             { <optional attributes>}
        ID                              Id-oc-inMobilitySubscriberGroup}
```

The inMobilitySubPrefix attribute is the distinguished attribute.

Entries subordinate to this object class must be of type inMobilityUserProfile.

II.2.3 Attribute types

II.2.3.1 inMobilityID

This attribute is used to uniquely distinguish between mobility users of a particular service provider or subscriber group.

The ASN.1 definition for inMobilityID is as follows:

```
inMobilityID                                    ATTRIBUTE ::= {
        WITH SYNTAX                             Digits (SIZE(lb-inMobilityID..ub-inMobilityID))
        EQUALITY MATCHING RULE          octetStringMatch
        ID                                      id-at-inMobilityID}
```

II.2.3.2 inMobilityPIN

This attribute is used to store a mobility user's PIN number.

```
inMobilityPIN                   ATTRIBUTE ::= {
        WITH SYNTAX     userPassword (SIZE lbinMobilityPIN..ubinMobilityPIN)
        ID                      id-at-inMobilityPIN}
```

II.2.3.3 inMobilityPrefix

This attribute uniquely distinguishes between mobility service providers.

```
inMobilityPrefix                    ATTRIBUTE ::= {
    WITH SYNTAX                     Digits (SIZE(lb-inMobilityPrefix..ub-inMobilityPrefix))
    EQUALITY MATCHING RULE          octetStringMatch
    ID                              id-at-inMobilityPrefix}
```

II.2.3.4 inMobilitySubPrefix

This attribute is used to group mobility subscribers.

```
inMobilitySubPrefix                 ATTRIBUTE ::= {
    WITH SYNTAX                     Digits (SIZE(lb-inMobilitySubPrefix..
                                                 ub-inMobilitySubPrefix))
    EQUALITY MATCHING RULE          octetStringMatch
    ID                                  id-at-inMobilitySubPrefix}
```

II.2.4 DIT structure definition

II.2.4.1 Name forms

A name form specifies the attribute that is to be used as the RDN for a specified object class.

II.2.4.1.1 inMobilityUserProfileNameForm

The following name form definition states that inMobilityID is the permitted distinguished attribute for the object class inMobilityUserProfile.

```
inMobilityUserProfileNameForm       NAME-FORM ::= {
    WITH ATTRIBUTES                 inMobilityID
    ID                              id-nf-inMobilityUserProfileNameForm}
```

II.2.4.1.2 inMobilityServiceProviderNameForm

The following name form definition states that inMobilityPrefix is the permitted distinguished attribute for the object class inMobilityServiceProvider.

```
inMobilityServiceProviderNameForm   NAME-FORM ::= {
    NAMES                           inMobilityServiceProvider
    WITH ATTRIBUTES                 inMobilityPrefix
    ID                              id-nf-inMobilityServiceProviderNameForm}
```

II.2.4.1.3 inMobilitySubscriberGroup

The following name form definition states that inMobilitySubPrefix is the permitted distinguished attribute for the object class inMobilitySubscriberGroup.

```
inMobilitySubscriberGroupNameForm   NAME-FORM ::= {
    NAMES                           inMobilitySubscriberGroup
    WITH ATTRIBUTES                 inMobilitySubPrefix
    ID                              id-nf-inMobilitySubscriberGroupNameForm}
```

II.2.4.2 Structure rules

Structure rules specify permitted subordinate and superior entries in a DIT. The following structure rules which are illustrated in Figure II.2 can be used as a basis for defining the structure rules required for mobility service:

sr1 **STRUCTURE-RULE ::= {**
NAME-FORM countryNameForm
ID 1}

sr2 **STRUCTURE-RULE ::= {**
NAME-FORM orgNameForm
SUPERIOR RULES sr1
ID 1}

• • •

sr<n> **STRUCTURE-RULE ::= {**
NAME-FORM inMobilityServiceProviderNameForm
SUPERIOR RULES sr<n-1>
ID <n>}

sr<n+1> **STRUCTURE-RULE ::= {**
NAME-FORM inMobilitySubscriberGroupNameForm
SUPERIOR RULES sr<n>
ID <n+1>}

sr<n+2> **STRUCTURE-RULE ::= {**
NAME-FORM inMobilityUserProfileNameForm
SUPERIOR RULES sr<n>, sr<n+1>
ID <n+2>}

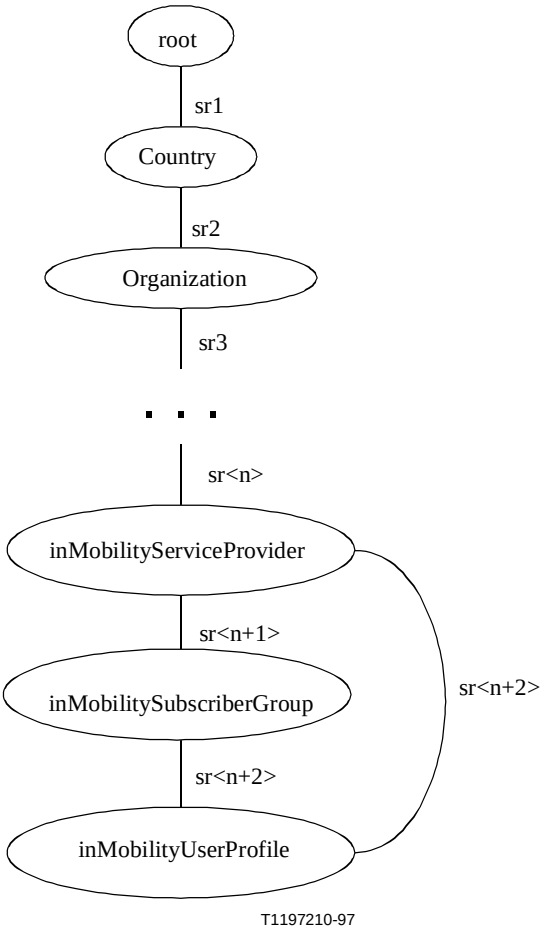


Figure II.2/Q.1228 – Structure rules

II.2.4.3 Object identifier assignments

The following object identifier assignments can be used as a starting point for identifying Mobility objects in Recommendation X.500.

id-at-inMobilityID **OBJECT IDENTIFIER ::= {id-at-inMobility 0}**
id-at-inMobilityPIN **OBJECT IDENTIFIER ::= {id-at-inMobility 1}**
id-at-inMobilityPrefix **OBJECT IDENTIFIER ::= {id-at-inMobility 2}**
id-at-inMobilitySubPrefix **OBJECT IDENTIFIER ::= {id-at-inMobility 3}**
id-oc-inMobilityUserProfile **OBJECT IDENTIFIER ::= {id-oc-inMobility 0}**
id-oc-inMobilityServiceProvider **OBJECT IDENTIFIER ::= {id-oc-inMobility 1}**
id-oc-inMobilitySubscriberGroup **OBJECT IDENTIFIER ::= {id-oc-inMobility 2}**
id-nf-inMobilityUserProfileNameForm **OBJECT IDENTIFIER ::= {id-nf-inMobility 0}**
id-nf-inMobilityServiceProviderNameForm **OBJECT IDENTIFIER ::= {id-nf-inMobility 1}**
id-nf-inMobilitySubscriberGroupNameForm **OBJECT IDENTIFIER ::= {id-nf-inMobility 2}**

APPENDIX III

Examples of SPKM algorithms for IN CS-2

III.1 General

A number of algorithm types are employed in SPKM. Each type, along with its purpose and a set of specific examples, is described in this Appendix. In order to ensure at least a minimum level of interoperability among various implementations of SPKM for IN CS-2, one of the integrity algorithms is specified as MANDATORY; all remaining examples (and any other algorithms) may optionally be supported by a given SPKM implementation, and this Appendix therefore specifies certain algorithms as RECOMMENDED.

SPKM makes use of the terms "initiator" and "target". For IN CS-2, "target" may also be referred to as "consumer".

III.2 Integrity Algorithm (I-ALG)

This algorithm is used to ensure that a message has not been altered in any way after being constructed by the legitimate sender. Depending on the algorithm used, the application of this algorithm may also provide authenticity and support non-repudiation for the message.

III.2.1 Example–1

```
md5WithRSAEncryption OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1)pkcs-1(1) 4
        }
```

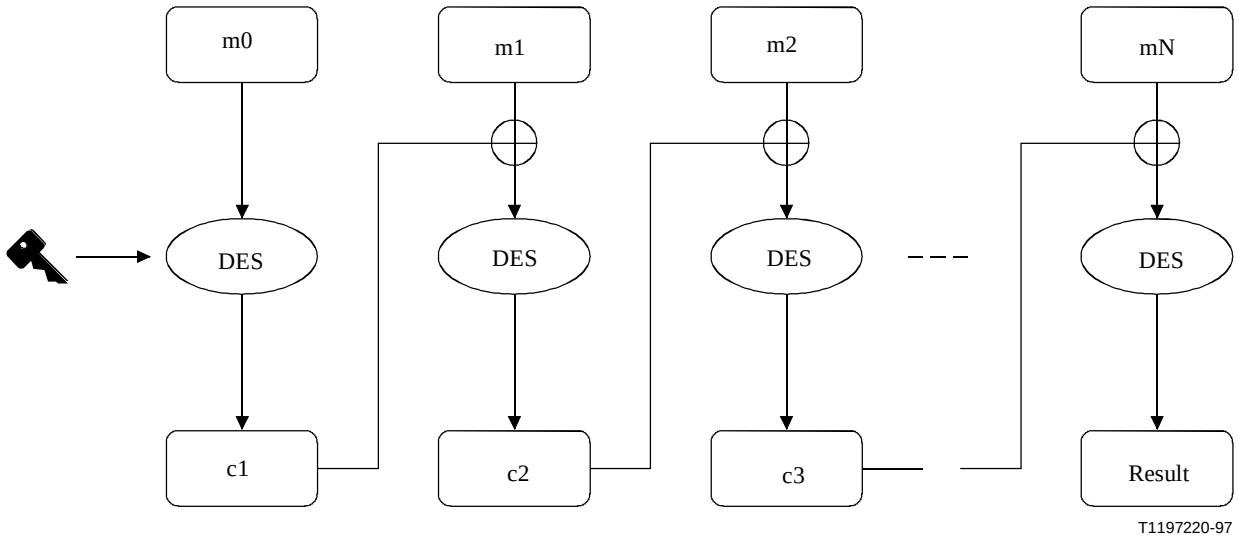
This algorithm (MANDATORY) provides data integrity and authenticity and supports non-repudiation by computing an RSA signature on the MD5 hash of that data.

Note that since this is the only integrity/authenticity algorithm specified to be mandatory at this time, for interoperability reasons it is also stipulated that md5WithRSA be the algorithm used to sign all context establishment tokens which are signed rather than MACed. In future versions of this Appendix, alternate or additional algorithms may be specified to be mandatory and so this stipulation on the context establishment tokens may be removed.

III.2.2 Example–2

DES-MAC OBJECT IDENTIFIER ::= {
iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 10
} *-- carries length in bits of the MAC as an INTEGER parameter, constrained to*
-- multiples of eight from 16 to 64

This algorithm (RECOMMENDED) provides integrity by computing a DES MAC on that data as illustrated in Figure III.1.



Plane text M=m1II m2IIm3II--II mN,
II: concatination
m0: Initial value (e.g. all "0")

Figure III.1/Q.1228 – DES-MAC mechanism

III.2.3 Example–3

md5-DES-CBC OBJECT IDENTIFIER ::= {
iso(1) identified-organization(3) dod(6) internet(1)
security(5) integrity(3) md5-DES-CBC(1)
}

This algorithm provides data integrity by encrypting, using DES-CBC, the "confounded" MD5 hash of that data. This will typically be faster in practice than computing a DES MAC unless the input data is extremely short (e.g. a few bytes). Note that without the confounder, the strength of this integrity mechanism is (at most) equal to the strength of DES under a known-plaintext attack.

III.2.4 Example–4

sum64-DES-CBC OBJECT IDENTIFIER ::= {
iso(1) identified-organization(3) dod(6) internet(1)
security(5) integrity(3) sum64-DES-CBC(2)
}

This algorithm provides data integrity by encrypting, using DES-CBC, the concatenation of the confounded data and the sum of all the input data blocks (the sum computed using addition modulo 2**64 – 1). Thus, in this algorithm, encryption is a requirement for the integrity to be secure.

III.3 Confidentiality Algorithm (C-ALG)

This symmetric algorithm is used to generate the encrypted data.

III.3.1 Example–1

```
DES-CBC OBJECT IDENTIFIER ::= {
        iso(1) identified-organization(3) oiw(14) secsig(3)  algorithm(2) 7
        }       -- carries IV (OCTET STRING) as a parameter; this (optional) parameter is
                -- unused in SPKM due to the use of confounding
```

This algorithm is RECOMMENDED.

III.4 Key Establishment Algorithm (K-ALG)

This algorithm is used to establish a symmetric key for use by both the initiator and the target over the established (security) context. The keys used for C-ALG and any keyed I-ALGs (for example, DES-MAC) are derived from this context key. Key establishment is done within the X.509 authentication exchange and so the resulting shared symmetric key is authenticated.

III.4.1 Example–1

```
RSAEncryption OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1
        }
```

In this algorithm (MANDATORY), the context key is generated by the initiator, encrypted with the RSA public key of the target, and sent to the target. The target need not respond to the initiator for the key to be established.

III.4.2 Example–2

```
id-rsa-key-transport OBJECT IDENTIFIER ::= {
        iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 22
        }
```

Similar to RSAEncryption, but source authenticating information is also encrypted with the target's RSA public key.

III.4.3 Example–3

```
dhKeyAgreement OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) pkcs-3(3) 1
        }
```

In this algorithm, the context key is generated jointly by the initiator and the target using the Diffie-Hellman key establishment algorithm. The target must therefore respond to the initiator for the key to be established.

III.5 One-Way Function (O-ALG) for Subkey Derivation Algorithm

Having established a context key using the negotiated K-ALG, both initiator and target must be able to derive a set of subkeys for the various C-ALGs and keyed I-ALGs supported over the context. Let the (ordered) list of agreed C-ALGs be numbered consecutively, so that the first algorithm (the "default") is numbered "0", the next is numbered "1", and so on. Let the numbering for the (ordered) list of agreed I-ALGs be identical. Finally, let the context key be a binary string of arbitrary length "M", subject to the following constraint: $L \leq M \leq U$ (where the lower limit "L" is the bit length of

the longest key needed by any agreed C-ALG or keyed I-ALG, and the upper limit "U" is the largest bit size which will fit within the K-ALG parameters).

For example, if DES and two-key-triple-DES are the negotiated confidentiality algorithms and DES-MAC is the negotiated keyed integrity algorithm (note that digital signatures do not use a context key), then the context key must be at least 112 bits long. If 512-bit RSAEncryption is the K-ALG in use, then the originator can randomly generate a context key of any greater length up to 424 bits. The target can determine the length which was chosen by removing the padding bytes during the RSA decryption operation. On the other hand, if dhKeyAgreement is the K-ALG in use, then the context key is the result of the Diffie-Hellman computation (with the exception of the high-order byte, which is discarded for security reasons), so that its length is that of the Diffie-Hellman modulus, p, minus 8 bits.

The derivation algorithm for a k-bit subkey is specified as follows:

rightmost_k_bits (OWF(context_key || x || n || s || context_key))

where:

- "x" is the ASCII character "C" (0x43) if the subkey is for a confidentiality algorithm, or the ASCII character "I" (0x49) if the subkey is for a keyed integrity algorithm;
- "n" is the number of the algorithm in the appropriate agreed list for the context [the ASCII character "0" (0x30), "1" (0x31), and so on];
- "s" is the "stage" of processing -- always the ASCII character "0" (0x30), unless "k" is greater than the output size of OWF, in which case the OWF is computed repeatedly with increasing ASCII values of "stage" (each OWF output being concatenated to the end of previous OWF outputs), until "k" bits have been generated;
- "||" is the concatenation operation; and
- "OWF" is any appropriate One-Way Function.

III.5.1 Example–1

```
MD5 OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 5
        }
```

This algorithm is MANDATORY.

```
SHA OBJECT IDENTIFIER ::= {
        iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 18
        }
```

It is recognized that existing hash functions may not satisfy all required properties of OWFs. This is the reason for allowing negotiation of the O-ALG OWF during the context establishment process, since in this way future improvements in OWF design can easily be accommodated. For example, in some environments a preferred OWF technique might be an encryption algorithm which encrypts the input specified above using the context_key as the encryption key.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages